目录

[（一） 实验一：认识DBMS 1](#_Toc39869514)

[1、 安装数据库管理系统 1](#_Toc39869515)

[2、 开启MySQL80服务 1](#_Toc39869516)

[3、 使用图形化界面Navicat Premium 2](#_Toc39869517)

[（二） 实验二：交互式SQL 3](#_Toc39869518)

[1、 新建StudentCourse数据库和学生、课程、选修表 3](#_Toc39869519)

[2、 表中数据的插入 4](#_Toc39869520)

[3、 修改表 6](#_Toc39869521)

[4、 删除表 7](#_Toc39869522)

[5、 索引的创建和删除 7](#_Toc39869523)

[6、 表的查询 8](#_Toc39869524)

[7、 表中数据的删除 13](#_Toc39869525)

[8、 表中数据的插入和修改 13](#_Toc39869526)

[9、 视图 14](#_Toc39869527)

[10、 遇到的问题 15](#_Toc39869528)

[（三） 实验三：数据控制（安全性部分） 16](#_Toc39869529)

[1、 创建用户 16](#_Toc39869530)

[2、 给用户授权 16](#_Toc39869531)

[3、 验证授权是否生效 19](#_Toc39869532)

[4、 授权回收和验证 21](#_Toc39869533)

[5、 角色创建和授权 22](#_Toc39869534)

[6、 出现的问题 23](#_Toc39869535)

[（四） 实验四：数据控制（完整性部分） 24](#_Toc39869536)

[1、 建表和三类完整性 24](#_Toc39869537)

[2、 CHECK语句（MYSQL实际有很大问题，可用ENUM或者触发器替代） 25](#_Toc39869538)

[3、 CONSTRAIN子句 25](#_Toc39869539)

[4、 触发器 27](#_Toc39869540)

[5、 遇到的问题 29](#_Toc39869541)

## 实验一：认识DBMS

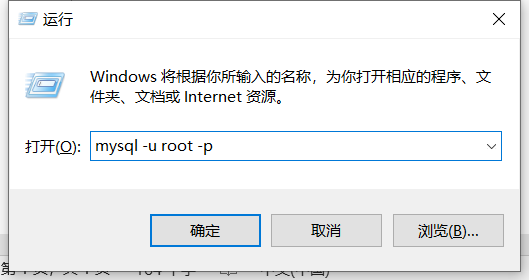
题目：认识DBMS 姓名：袁逸夫 日期：2020/4/28 实验环境：MySQL

### 安装数据库管理系统

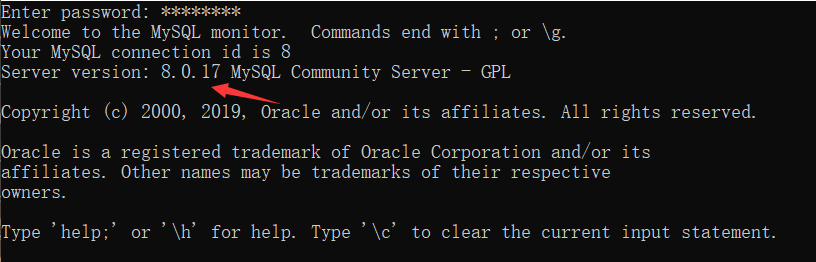
本实验中安装的数据库管理系统为MySQL，相应的图形化界面工具为Navicat Premiun12。

经过解压安装的过程后，我们需要配置系统变量和设置密码。

安装好之后，我们可以通过cmd命令行方式以root用户登录MySQL。使用快捷键【Win+R】打开运行界面，输入【mysql -u root -p】然后点击【确定】。

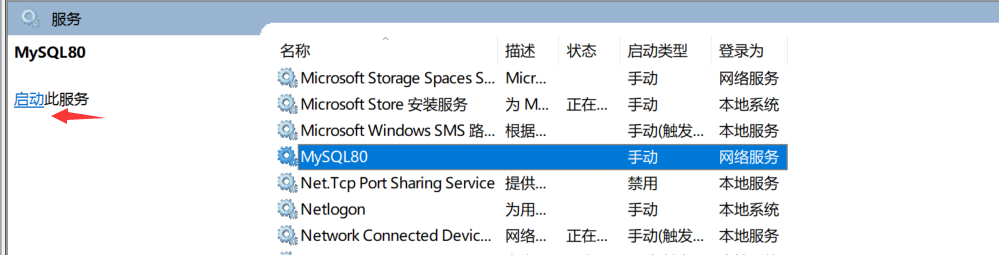


查看MySQL版本为8.0.17。



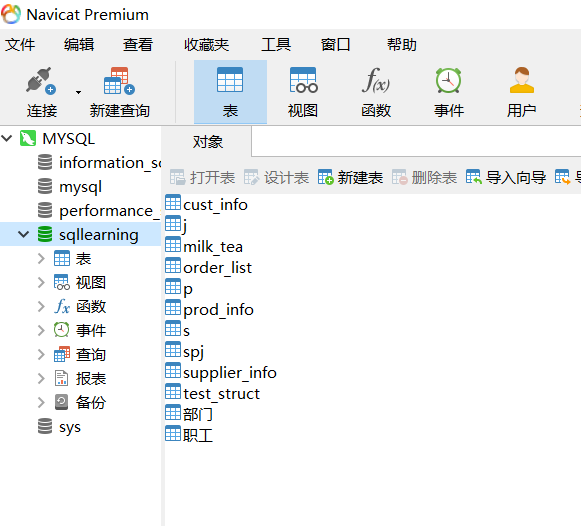
### 开启MySQL80服务

遇到的问题：使用MYSQL之前需要在此电脑的管理应用程序中开启MySQL80服务，否则会出现找不到mysql的错误！



### 使用图形化界面Navicat Premium

左侧菜单栏即创建的各种数据库，可以单击查看数据库中的各种表，并进行查询，修改等操作。

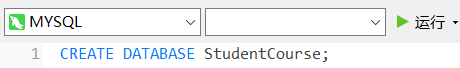


## 实验二：交互式SQL

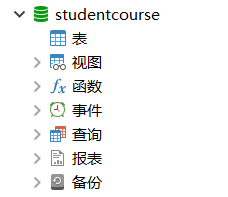
题目：交互式SQL 姓名：袁逸夫 日期：2020/4/29 实验环境：MySQL

### 新建StudentCourse数据库和学生、课程、选修表

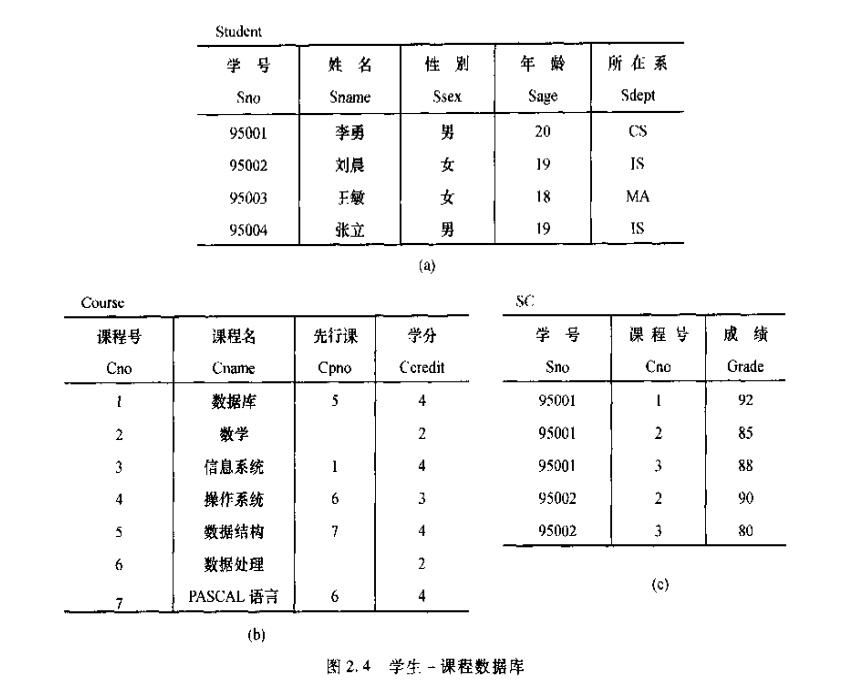
我们使用课本上的学生-课程数据库来进行实验。第一步我们需要在MySQL中新建该数据库。



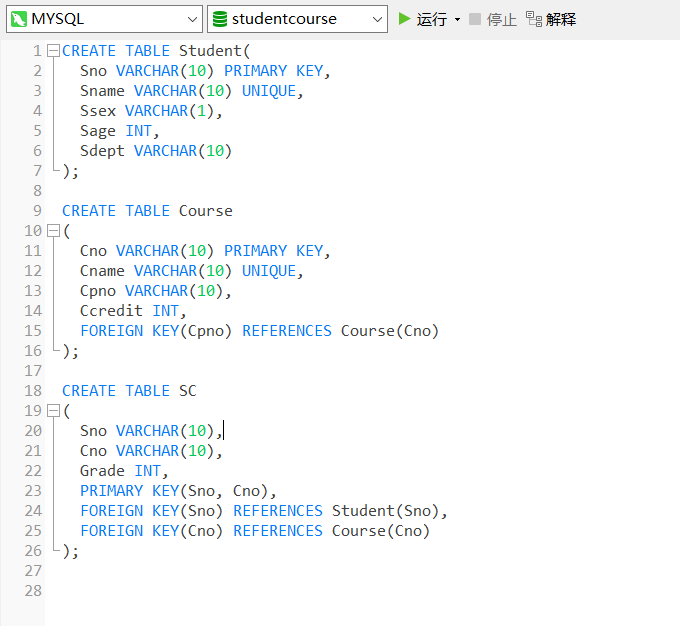
运行结果如下（新建了StudentCourse数据库）：



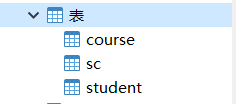
数据库的具体表信息如下图所示，我们在StudentCourse数据库下依次建立学生、选修、课程三张表。



建立表的结果如下：

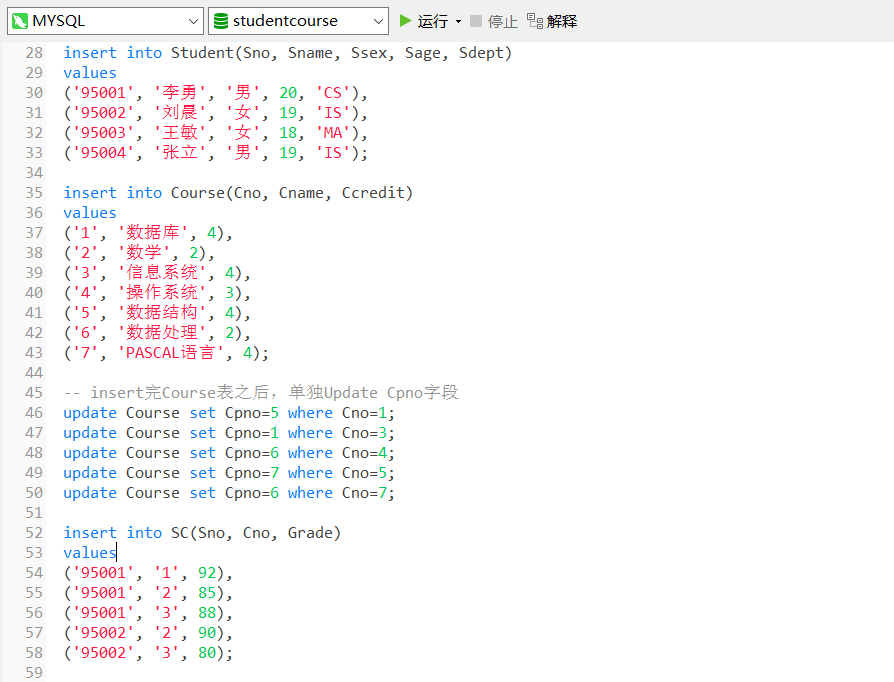


在Student表中，我们使用Sno作为主键，并且Sname字段使用唯一索引。在Course表中，我们使用Cno作为主键，使用Cpno字段作为外码，参考Course表中的Cno字段，在SC表中使用Sno和Cno字段作为主键，同时分别作为外码参考Student表和Course表。建立表结果如下：

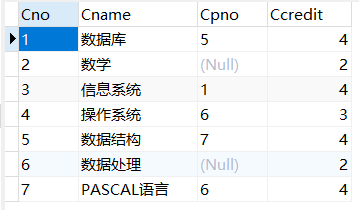


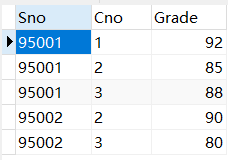
### 表中数据的插入

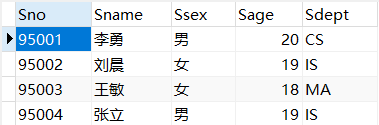
我们主要使用insert语句进行数据的插入，但是在Course表中插入数据不能直接使用insert，因为这样可能会产生外键冲突，Cpno字段的值和参考的Cno的值不同，产生报错，所以我们采用先把Cpno字段信息置为NULL，先插入其他信息，再单独更新该列。具体操作代码如下图：



操作结果如下：

Course表

SC表

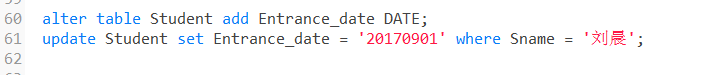
Student表

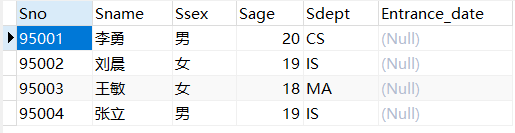
### 修改表

我们使用ALTER TABLE语句进行表中字段的修改

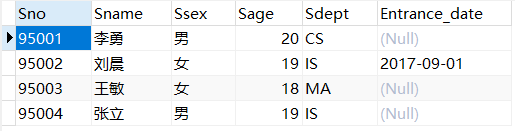
1. 在表中添加列

在Student表中添加入学时间字段“Entrance\_date”，定义数据类型为Date类型。添加后表如下，增加了一个新字段Entrance\_date。





我们在该字段中更新一条记录看看效果，例如刘晨的入学时间为2017/09/01。

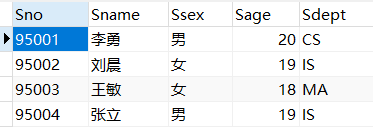


1. 在表中删除列

这里我们删除刚才创建的Entrance\_date字段



删除结果如下：



1. 改变表中列的数据类型

这里我们修改Ssex的数据类型，从VARCHAR（1）改为CHAR（10）。由于运行环境为MySQL，这里使用到了modify语法，而非课本上的alter…alter，但是实现功能一致。



修改前



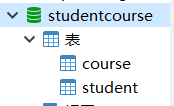
修改后

### 删除表

删除SC表，执行语句：



运行后，studentcourse数据库中的SC表被删除。



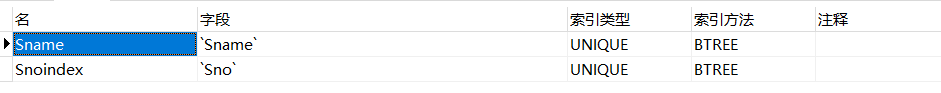
### 索引的创建和删除

（1）索引的创建

在Student表的Sno字段中新建唯一索引Snoindex，执行语句为：



执行结果为：



（2）索引的删除

删除上面创建的索引，执行如下语句：



执行结果即删除了Sno的索引

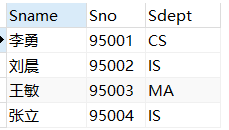


### 表的查询

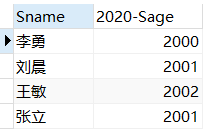
#### （1）单表查询

单表查询即只涉及到一个表的查询。

* 1. 



* 1. 



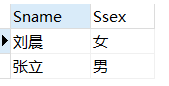
* 1. 



* 1. 



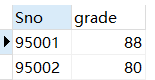
* 1. 



* 1. 

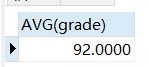


* 1. Order by的用法



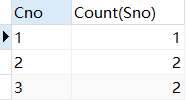
* 1. 聚集函数的用法





* 1. Group by的用法





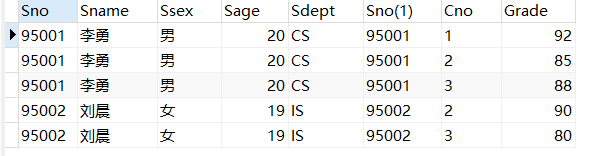
* 1. 



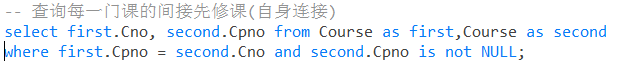
#### 连接查询

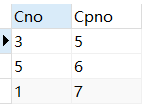
若一个查询同时涉及两个以上的表，则称之为连接查询。

* 1. 

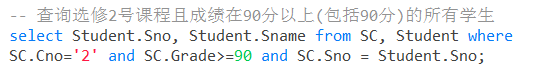


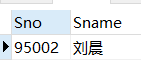
* 1. 自身连接（一个表与其自己进行连接）





* 1. 复合条件连接

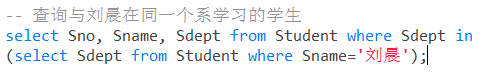




#### 嵌套查询

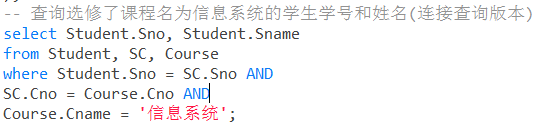
一个SELECT-FROM-WHERE构成一个查询块，一个查询快嵌套在另一个查询块成为嵌套查询。

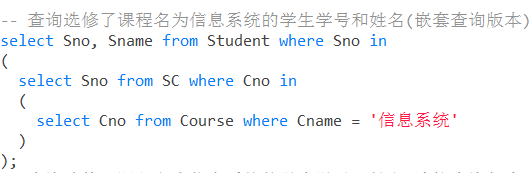
* 1. 带有IN谓词的子查询

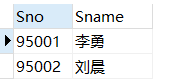




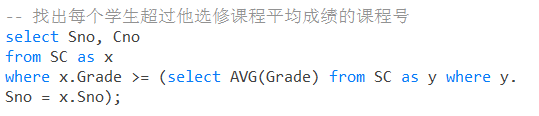
* 1. 嵌套查询和连接查询有些可以互相替代，拿下面问题举例

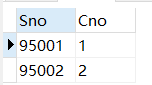




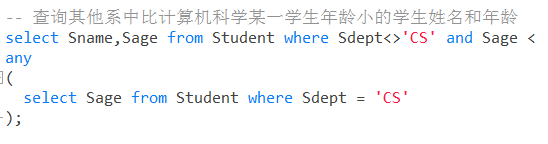
两种方法结果一致。

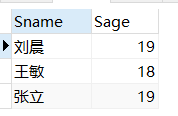
* 1. 带有比较运算符的相关子查询





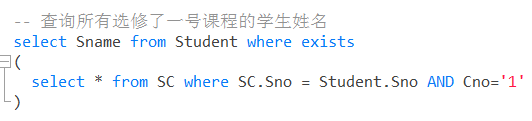
* 1. 带有ANY或ALL谓词的子查询

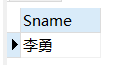




* 1. 带有EXIST谓词的子查询

带有EXIST谓词的子查询不返回任何数据，只产生逻辑真值/假值

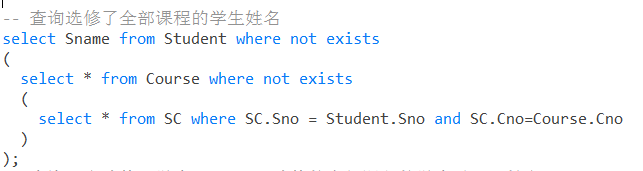




* 1. 查询选修了全部课程的学生姓名

经过等价转换，选修了全部课程即为查询这样的学生，没有一门课程是他不选修的

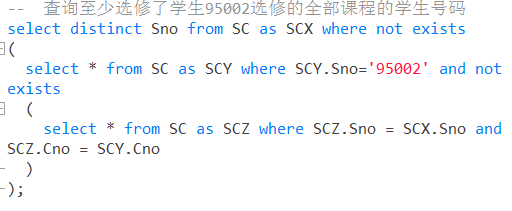






* 1. 查询至少选修了学生95002选修的全部课程的学生号码和姓名

换算表达为：不存在这样的课程y，学生95002选修了该课程，而学生x却没有选

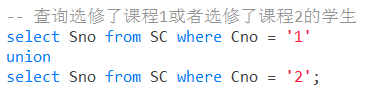




#### 集合查询

多个select语句可以进行集合操作，集合操作包括UNION,INTERSECT,EXCEPT

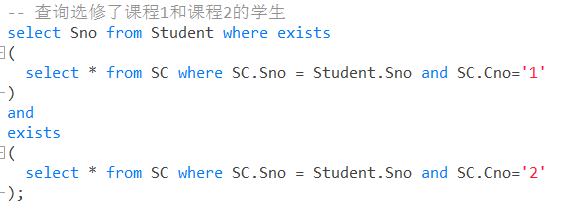
* 1. 查询选修了课程1或者选修了课程2的学生





* 1. 查询即选修课程1又选课程2的学生

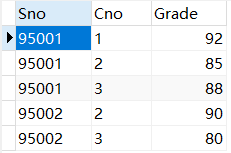
（在MYSQL环境中，没有intersect语法，所以我们使用存在量词exist来模拟intersect）



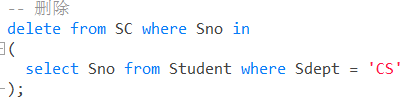


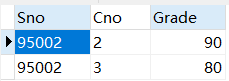
### 表中数据的删除

删除计算机科学系所有学生的选课记录，原有记录如下，其中95001为计算机科学系学生。



执行如下语句：



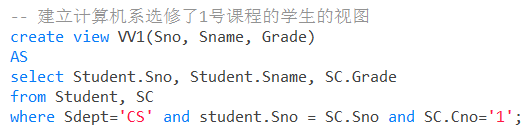


### 表中数据的插入和修改

在2.1节建立表的时候已经完成了对表中数据的插入和修改，故此处省略。

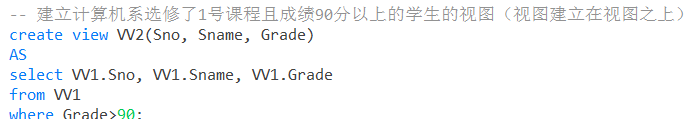
### 视图

#### 视图的定义（创建和删除）





在视图上创建视图



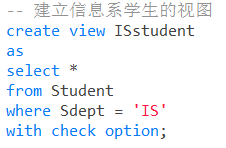


删除视图VV1和VV2，使用CASCADE可以一并删除级联的视图



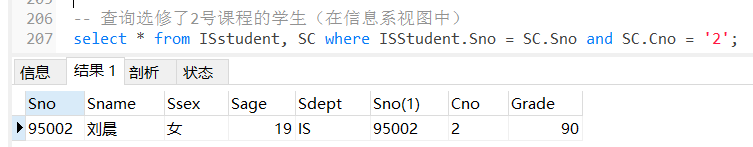
#### 视图的查询

先建立一个信息系学生的视图，以便查询和更新，这里使用了with check option语句，保证更新、插入、删除只对信息系学生生效。

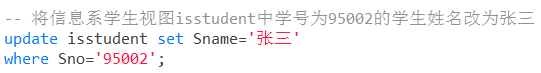




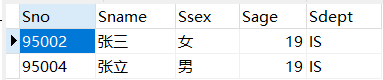
查询选修了2号课程的学生



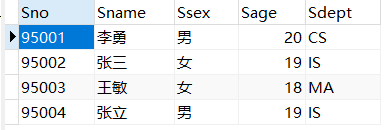
#### 视图的更新



视图中的更新



Student表中相应字段更新



### 遇到的问题

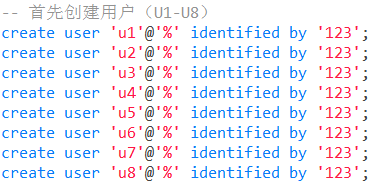
在该实验中遇到的问题主要是使用全称量词查询和离散数学的结合比较难以理解，经过多次练习和查阅资料最终解决了。另外MySQL环境也有一些和书本上不一致的地方，例如更改表中字段的数据类型不用alter…alter而是用alter…modify，例如集合运算没有交集操作关键词intersect，都已经在实验中较好的解决。

## 实验三：数据控制（安全性部分）

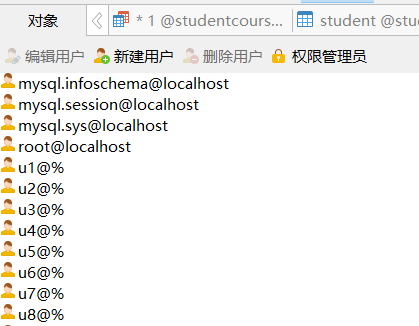
题目：数据控制 姓名：袁逸夫 日期：2020/5\* 实验环境：MySQL

### 创建用户

创建了u1-u8等8个用户，并设置密码为123，注意，因为在本实验中不要求对用户ip地址访问的限定，所以一律使用‘%’标志符，来标识任何主机都可以登录该用户进行访问。



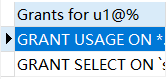
创建用户成功。

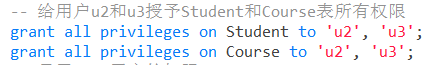


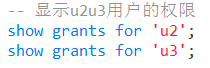
### 给用户授权



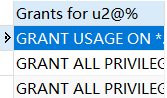


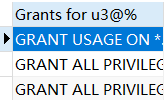




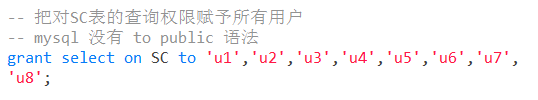


结果如下：

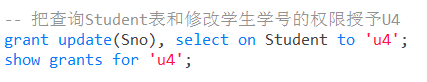


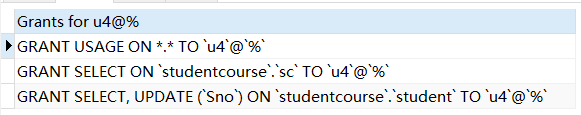


把对SC表的查询权限赋予所有用户，因为mysql没有to public语法，所以我们对每个用户分别授权。

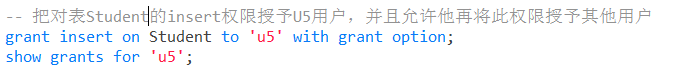


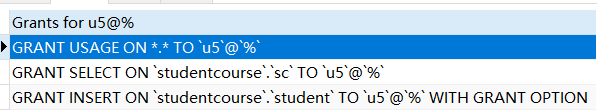
把对Student表的查询和修改学生学号权限赋予u4，并显示u4权限。





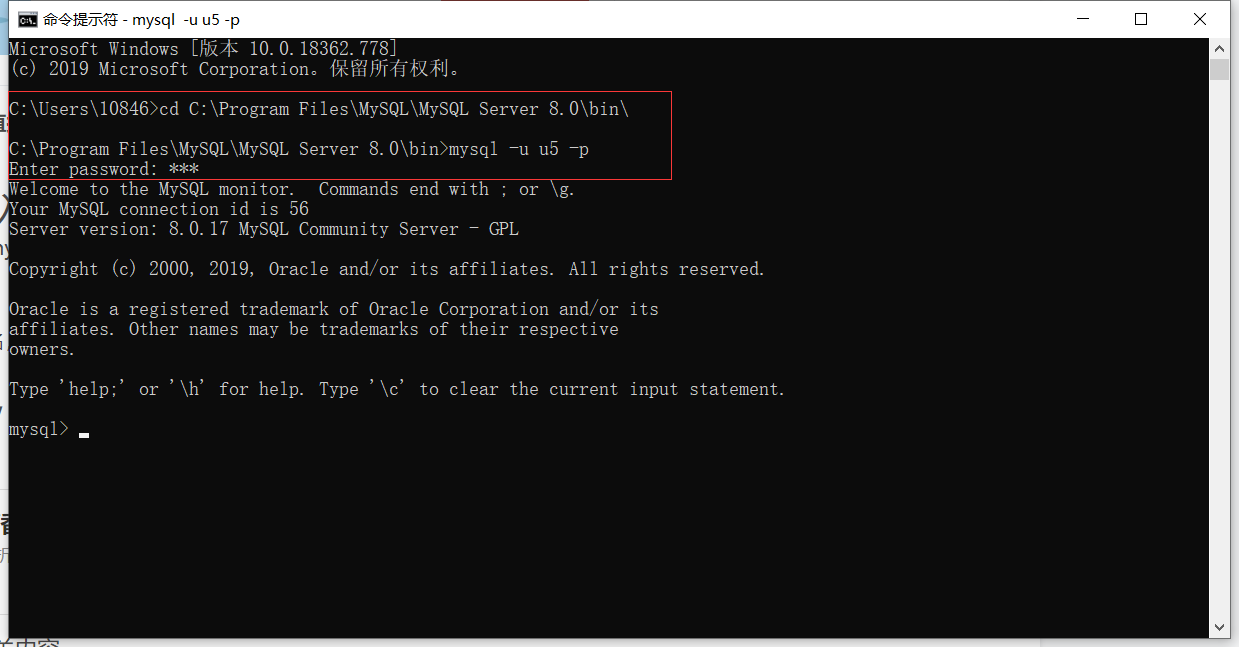
把对Student表的insert权限赋予u5，并且使用with grant option命令允许u5将该权限赋予其他用户。



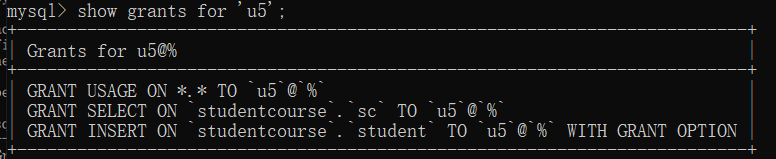


我们为了验证u5是否能授权给其他用户，使用mysql的命令行登录用户u5（因为在navicat图形界面里，默认是root，不方便切换用户，因此下面操作转到命令行）。

如红框所示，用户u5登录成功。



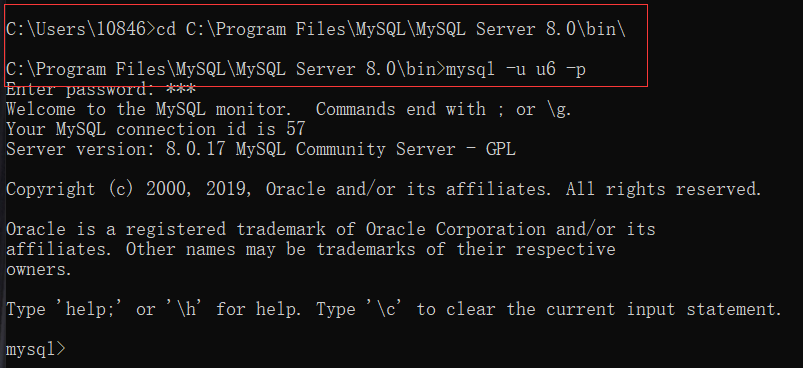
显示用户‘u5’的权限

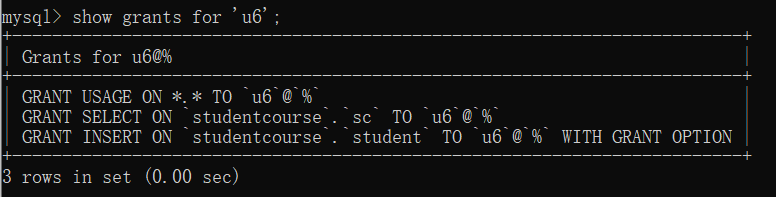


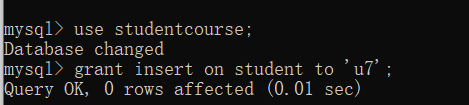
将对student表的insert权限赋予给u6用户，并且赋予u6将此权限赋予其他用户的权限。



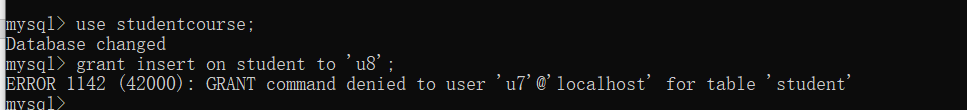
之后登录用户U6测试是否拥有该权限





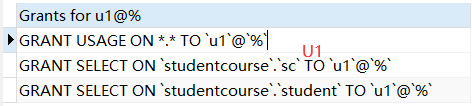


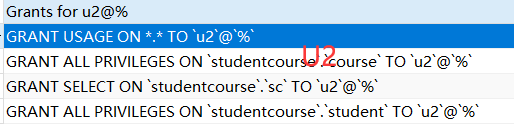
将insert权限赋予用户’u7’，并且不允许将权限赋予其他用户，我们登录用户’u7’，果然无法授予其他用户权限。

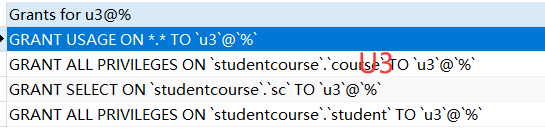


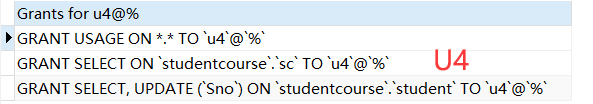
### 验证授权是否生效

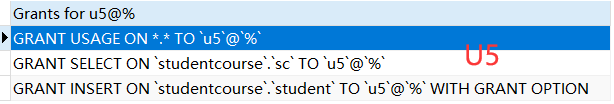
为了避免混乱，我们首先查看u1-u8所有用户的权限。

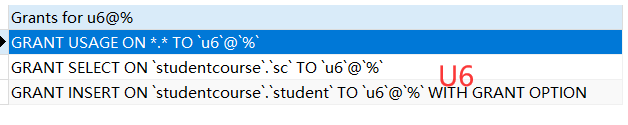


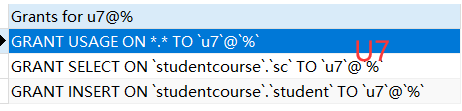


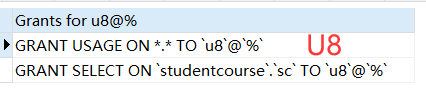




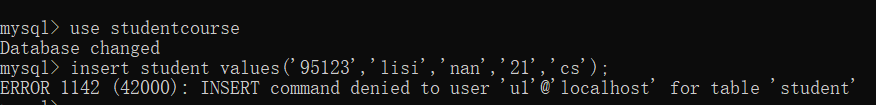






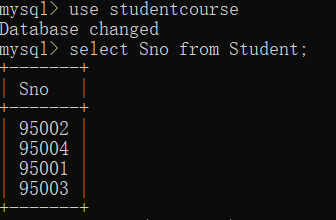


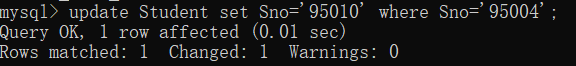
* 1. 登录用户u1，尝试对Student表进行insert操作（实际上u1只有select权限）



显示没有insert权限，验证成功。

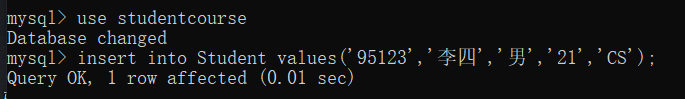
* 1. 登录用户U4，尝试查询和更新Student表中Sno列，把90001改为90010

查询结果



进行修改，修改成功。注意因为Student的Sno是主键，可能会影响了其他表，所以要在不影响其他表的依赖性的前提下进行修改。因为95004学号在其他表中并未使用，所以可以改成95010，如果使用了，需要相应的进行级联（CSACADE）修改。

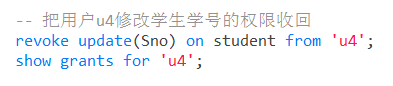
* 1. 登录用户U5，在Student表中插入新数据(‘95123’,’李四’,’男’,’21’，’CS’)

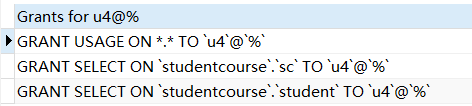


插入成功。

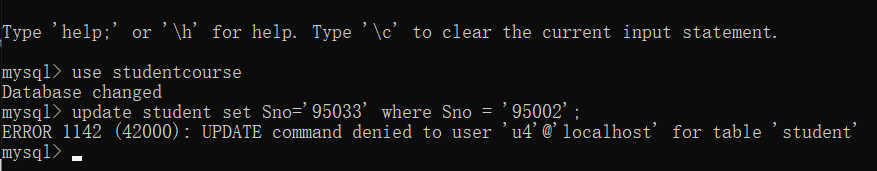
实验结果和之前列出的每个用户的应有权限一致，说明授予的权限均正常生效。

### 授权回收和验证



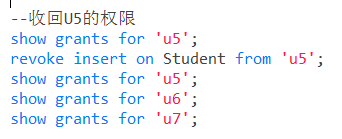


下面验证update Sno的权限是否收回，我们在命令行登录用户U4验证是否权限正确收回。

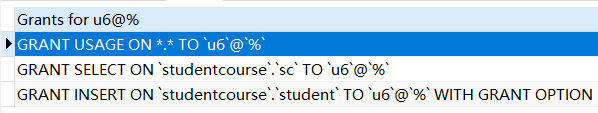


无法正常进行update操作，权限正常收回，验证成功。

由于MySQL没有级联回收的语法，所以收回U5的权限后，U6和U7的相应权限不会被收回。

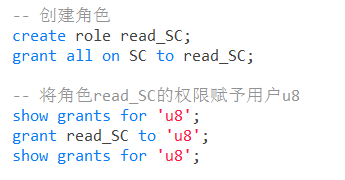


可以看出U6用户的select权限依然正常。

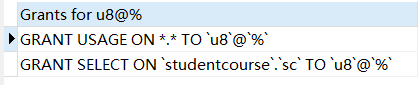


### 角色创建和授权

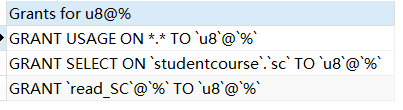
创建角色read\_SC,拥有访问SC表的所有权限，接下来把read\_SC角色的权限赋予用户u8。



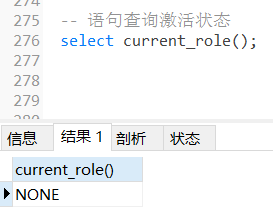
赋予权限之前：



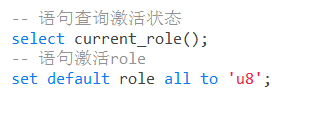
赋予权限之后：



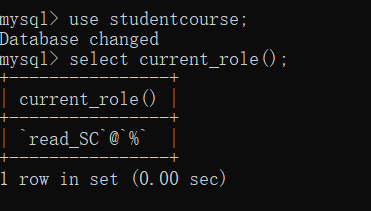
注意，虽然角色的权限已经赋予到用户u8，但是授予的角色需要在会话中处于活动状态，权限才是可用的，我们在这里使用select current\_role();语句来查询激活情况。



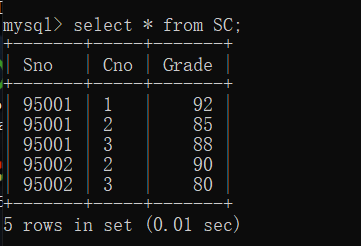
显然，角色read\_SC尚未激活，我们使用set default role激活角色。



进行查询，激活成功。



下面验证用户u8的功能，是否能够对表SC进行正常操作。

能够使用select功能，授权成功。

### 出现的问题

本次实验遇到的问题主要是MySQL中的语法定义和书本上讲的有一些细微的差别，例如MySQL没有级联的权限回收，u6通过u5获得了权限后，只能进行u5u6权限的依次收回。另外，在图形界面navicat切换用户也不方便，涉及到不同用户的功能验证需要用到MySQL的命令行界面。经过摸索和查阅资料，对于数据控制这块功能基本掌握。

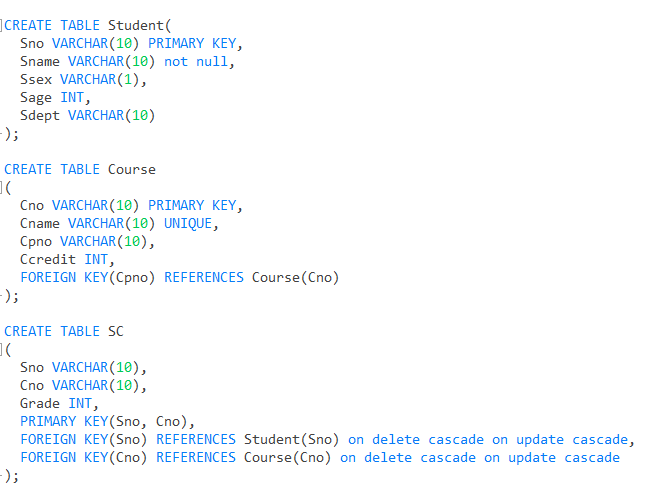
## 实验四：数据控制（完整性部分）

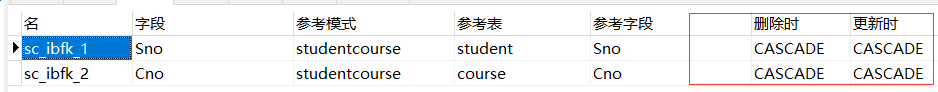
题目：认识DBMS 姓名：袁逸夫 日期：2020/5/4 实验环境：MySQL

### 建表和三类完整性

本次实验同样适用Student，SC，Course表进行操作，但是由于前面进行的一些修改和添加的操作以及需要添加新的约束，为了避免不必要的冲突报错，我们删除了之前的表，重新建立三个表并插入数据，同时针对性的赋予了一些新的完整性约束。

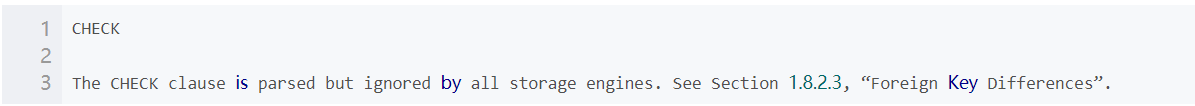
**新的完整性约束包括SC表中关于外键Sno和Cno的更新级联和删除级联，同时三个表的主键体现实体完整性，外键体现参照完整性，用户自定义完整性有Cname的unique，Sname的not null。完整代码如下：**

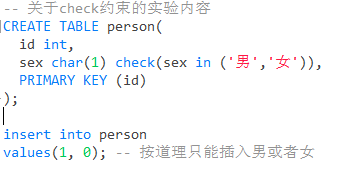


外键属性如下：

### Check语句（MYSQL实际有很大问题，用ENUM或者触发器替代）

MySQL所有的存储引擎实际都不支持CHECK约束，虽然在MySQL中可以写CHECK约束，但是会忽略它的作用，因此CHECK并不起作用。有官方文档为证：

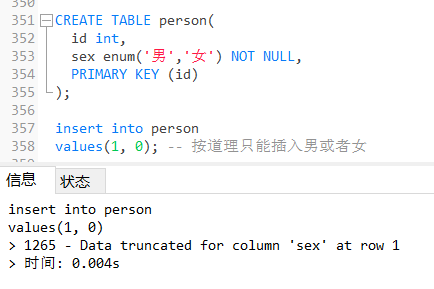
因此实现对数据约束有两种办法：**使用ENUM类型或者使用触发器**，离散的用enum，连续的用触发器，触发器在第4节详细介绍，这里介绍ENUM类型。



测试check发现没有成功约束，错误的结果可以插入。（可能是SQL版本不同，同样的代码经过不同的SQL版本又发现产生了成功的约束，总之这个CHECK方法不靠谱）。



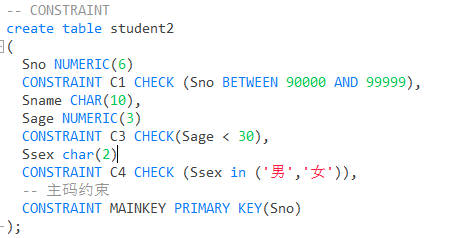
下面用enum类型重新约束这个性别限定于男女，重新建表：

这时插入非男，女的性别即报错，可以达成check的功能。

### CONSTRAIN子句

CONSTRAIN子句可以通过为完整性约束条件命名从而灵活地增加、删除一个完整性约束条件。

例如：建立学生登记表Student，要求学号在90000-99999之间，姓名不能取空值，年龄小于30，性别只能是男/女。



这里适用了CONSTRAINT和CHECK结合的约束方式，我们来测试一下CHECK的约束是否生效。

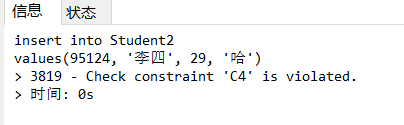
正确例子1：





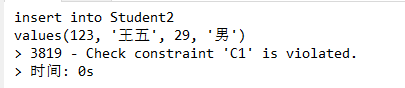
错误例子1：





错误例子2：





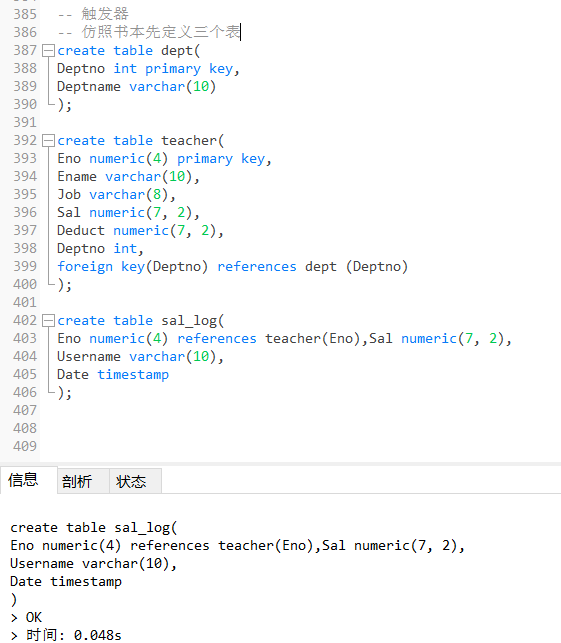
结合正确错误例子，结果生效，可行，说明在MYSQL中单独用CHECK可能出现各种问题，但是CONSTRAINT结合使用可行。

下面是删除约束条件C3的语句：

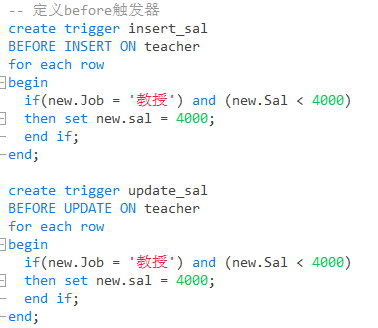


### 触发器

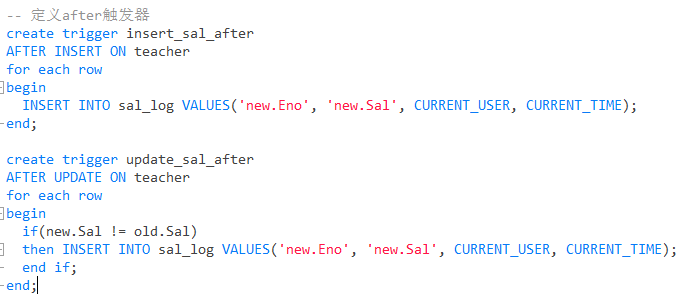
* + 1. 为了方便实验，我们先使用教科书上资料定义三个表dept，teacher，sal\_log。



* + 1. 定义before触发器，功能是教授的工资不得低于4000，如果低于4000，则自动改为4000。（和教科书不同的是，mysql不能直接在一个触发器中设置两个出发，比如update or insert，会报错，必须update设置一个触发器，insert设置一个触发器）

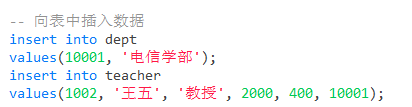


* + 1. 定义两个after触发器，也是同理，这个触发器的作用是当教师表Teacher的工资发生变化后就自动在工资变化表sal\_log增加一条相应记录。



* + 1. 下面进行触发器的激活试验，并探究同一个表上多个触发器的执行顺序。由于之前创建的三个表都是空表，所以需要先进行插入数据。

执行语句为：



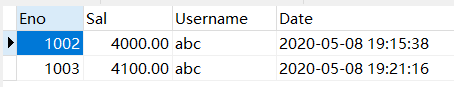
**则分别执行了before的insert触发器和after的insert触发器**，之前王五教授的工资<4000，自动变为4000，并且在sal表中新增一条工资变动记录。





接下来测试update的触发器效果，先建立一条记录

4100>4000，这条记录的工资不会变化，但是会在sal里面添加一条记录



之后测试update的触发器效果

首先执行update before的触发器，new.Sal变为4000



然后执行update after，把变化记录到Sal\_log



至此执行完毕。

### 遇到的问题

本次实验遇到的问题一是CHECK语法在mysql中似乎不可靠（有时候生效有时候不生效），查阅相关资料结合了官方文档，发现check语法确实不可靠，但是可以通过触发器，enum数据类型和constraint实现相应的功能。另一个需要注意的就是触发器和update的结合，我一度以为update before不管怎么样都在update前面，不会影响到update，实际操作才发现实际上是把update中的Sal值给修改了，所以能起到正确的约束作用，本次实验收获良多！