数据库原理 第三次实验报告

09019106 牟倪

目录

[数据库原理 第三次实验报告 1](#_Toc89350097)

[一、实验目的 2](#_Toc89350098)

[二、实验内容 2](#_Toc89350099)

[2.1 完整性约束的内容 2](#_Toc89350100)

[2.2 考察域完整性约束 2](#_Toc89350101)

[2.2.1 尝试1：数据类型不同 3](#_Toc89350102)

[2.2.2 尝试2：数据值不在约束范围内 3](#_Toc89350103)

[2.2.3 尝试3：数据值不在约束范围内 4](#_Toc89350104)

[2.2.4 尝试4：数据值不在约束范围内 4](#_Toc89350105)

[2.3 考察实体完整性约束 4](#_Toc89350106)

[2.3.1 尝试5：插入主键=NULL的记录 4](#_Toc89350107)

[2.3.2 尝试6：将记录的主键修改为NULL 5](#_Toc89350108)

[2.3.3 尝试7：将记录的主键修改为NULL 5](#_Toc89350109)

[2.3.4 尝试8：将多主键记录的其中一个主键修改为NULL 6](#_Toc89350110)

[2.3.5 尝试9：插入主键重复的记录 6](#_Toc89350111)

[2.3.6 尝试10：将记录的主键修改为重复值 7](#_Toc89350112)

[2.4 考察引用完整性约束 7](#_Toc89350113)

[2.4.1 尝试11：插入荡空的记录 8](#_Toc89350114)

[2.4.2 尝试12：级联删除 8](#_Toc89350115)

[2.4.3 尝试13：级联更新 9](#_Toc89350116)

[2.4.4 尝试14：级联删除失败 10](#_Toc89350117)

[2.4.5 尝试15：级联更新失败 11](#_Toc89350118)

[2.4.4 尝试16：多表级联删除 11](#_Toc89350119)

[三、实验总结 13](#_Toc89350120)

# 一、实验目的

* 了解三条完整性约束的内容。
* 在Access中进行相关实践，考察Access是否遵循三条完整性约束。
* 熟练掌握Access中的INSERT、UPDATE、DELETE操作。

# 二、实验内容

## 2.1 完整性约束的内容

* 域完整性约束（Domain Integrity Constraint）：一条记录中所有属性的数据类型和取值范围，必须符合data schema的事先约定。比如说，定义绩点为0到4.8之间的浮点数，则绩点不能为字符串、不能为负数、也不能超过4.8。
* 实体完整性约束（Entity Integrity Constraint）：一条记录里，被定义为的主键（primary key）的属性不能有重复、不能为NULL。
* 引用完整性约束（Referential Integrity Constraint）：如果relation A中某一属性为另一relation B的外键（foreign key），则每一relation B中的记录都应该能对应上一个A中的记录。也可以这样理解，在关联表里，只能将存在的事物相互关联，只能引用存在的事物，所以叫引用完整性约束。
  + 比如说，存在“我在计算机专业”这样一条记录，则不能在数据库中删去“我”或“计算机专业”，也不能改变“我”或“计算机专业”记录的主键，否则“我在计算机专业”将没有可对应的记录。
  + 我们也可以通过级联更新/删除的方式维护引用完整性约束。比如说，修改“计算机专业”为“计软智学院”，则关联表的内容也相应更新，“我在计算机专业”更新为“我在计软智学院”；删除“计算机专业”的记录，则关联表的内容响应删除，“我在计算机专业”将被删除。

## 2.2 考察域完整性约束

对于给出的大学信息表，我们细化Student表的属性数据类型/取值范围：

* 将sid的类型限制为数字，取值范围限制为大于零，代码为[sid]>0；
* 将sex的类型限制为文本，取值范围限制为’f’或’m’，代码为[sex]='f' Or [sex]='m'；
* 将gpa的类型限制为数字，取值范围在0与4之间，代码为[gpa]>=0 And [gpa]<=4；

接下来，我们尝试插入不符合域完整性约束的记录，将原有记录修改成不符合域完整性约束的样子。

### 2.2.1 尝试1：数据类型不同

尝试插入 (sid, sname, sex, age, year, gpa) 为 (’hello’, ’uran’, ’f’, 1000, 3, 4) 的记录，其中sid=’hello’违反了域完整性约束。

SQL语句（见Insert1）：

1. INSERT INTO Student
2. VALUES ('hello', 'uran', 'f', 1000, 3, 4);

运行结果：



插入记录失败。

### 2.2.2 尝试2：数据值不在约束范围内

尝试插入 (sid, sname, sex, age, year, gpa) 为 (-3, ’uran’, ’f’, 1000, 3, 4) 的记录，其中sid=-3违反了域完整性约束。

SQL语句（见Insert2）：

1. INSERT INTO Student
2. VALUES (-3, 'uran', 'f', 1000, 3, 4);

运行结果：



插入记录失败。

### 2.2.3 尝试3：数据值不在约束范围内

尝试插入 (sid, sname, sex, age, year, gpa) 为 (1000, ’uran’, ’great’, 1000, 3, 4) 的记录，其中sex=’great’违反了域完整性约束。

SQL语句（见Insert3）：

1. INSERT INTO Student
2. VALUES (1000, 'uran', 'great', 1000, 3, 4);

运行结果：



插入记录失败。（有趣的是，目前所有的报错信息都一样）

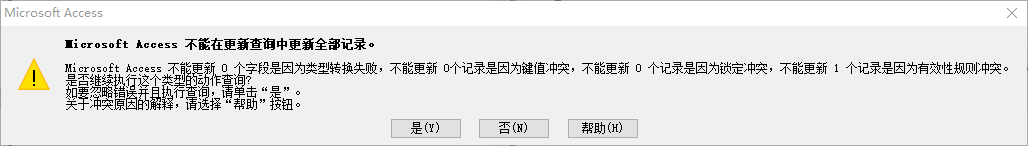
### 2.2.4 尝试4：数据值不在约束范围内

尝试将sid=1的学生信息记录中gpa修改为gpa=8， gpa=8违反了域完整性约束。

SQL语句（见Update1）：

1. UPDATE Student SET gpa=8
2. WHERE sid=1;

运行结果：



修改记录失败。（UPDATE的报错信息和INSERT略有不同）

## 2.3 考察实体完整性约束

我们尝试插入主键重复的记录，尝试把原有记录的主键修改为重复的值；尝试插入主键为NULL的记录，尝试把原有记录的主键修改为NULL。

### 2.3.1 尝试5：插入主键=NULL的记录

尝试插入 (sid, sname, sex, age, year, gpa) 为 (NULL, ’uran’, ’f’, 1000, 3, 4) 的记录，其中sid=NULL违反了实体完整性约束。

SQL语句（见Insert4）：

1. INSERT INTO Student
2. VALUES (NULL, 'uran', 'f', 1000, 3, 4);

运行结果：



插入记录失败。（有趣的是，报错信息和违反域完整性约束的INSERT相同）

### 2.3.2 尝试6：将记录的主键修改为NULL

尝试将sid=1的学生信息记录中sid修改为sid=NULL， sid=NULL违反了实体完整性约束。

SQL语句（见Update2）：

1. UPDATE Student SET sid = NULL
2. WHERE sid=1;

运行结果：



修改记录失败。（有趣的是，报错信息和违反域完整性约束的UPDATE相同）

### 2.3.3 尝试7：将记录的主键修改为NULL

尝试修改具有多个主键的relation。这里，我们来考察Course表，它有cno和dname两个主键。

尝试将cno=302、dname=’Computer Sciences’的记录，修改为cno=NULL、dname=NULL，其中cno=NULL、dname=NULL违反了实体完整性约束。

SQL语句（见Update3）：

1. UPDATE Course SET cno = NULL, dname=NULL
2. WHERE cno = 302 AND dname='Computer Sciences';

运行结果：



修改记录失败。

### 2.3.4 尝试8：将多主键记录的其中一个主键修改为NULL

我们尝试仅将其中一个主键修改为NULL。

尝试将cno=302、dname=’Computer Sciences’的记录，修改为cno=NULL，其中cno=NULL违反了实体完整性约束。

SQL语句（见Update4）：

1. UPDATE Course SET cno = NULL
2. WHERE cno = 302 AND dname='Computer Sciences'

运行结果：



修改记录同样失败。可以看出，所有的主键都不能为NULL。

### 2.3.5 尝试9：插入主键重复的记录

尝试插入 (sid, sname, sex, age, year, gpa) 为 (1, ’uran’, ’f’, 1000, 3, 4) 的记录，其中sid=1是重复主键，违反了实体完整性约束。

SQL语句（见Insert6）：

1. INSERT INTO Student
2. VALUES (1, 'uran', 'f', 1000, 3, 4);

运行结果：



插入记录失败。

### 2.3.6 尝试10：将记录的主键修改为重复值

尝试将sid=1的学生信息记录中sid修改为sid=2， sid=2是重复主键，违反了实体完整性约束。

SQL语句（见Update6）：

1. UPDATE Student SET sid = NULL
2. WHERE sid=1;

运行结果：

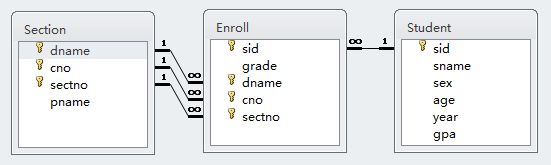


修改记录失败。

## 2.4 考察引用完整性约束

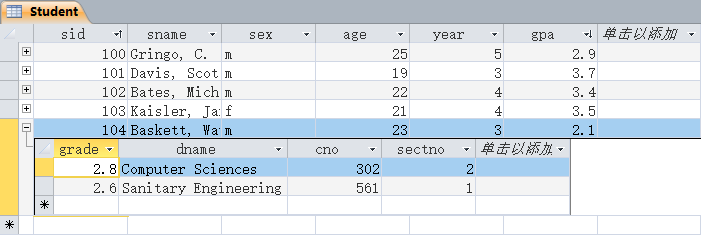
我们拿Enroll表作为考察对象。Enroll表记录了学生选课的信息，有4个主键sid、dname、cno、sectno，其中sid与学生信息（Student表）关联，dname、cno、sectno与具体课程信息（Section表）关联。

建立关联如下：



对每一个关联，都勾选了“实施参照完整性”（即引用完整性）、“级联更新相关字段”、“级联删除相关记录”，关系类型均为一对多（符合实际情况，一个学生可以选多门课，一门课可以有多个学生报名）。

建立关系后，发现access为数据表提供了新的功能：对一条具体记录（如一个Student记录）进行展开，可以看到在关联表中的关联记录（Enroll表中该学生的选课信息），如图所示。



### 2.4.1 尝试11：插入荡空的记录

尝试向Enroll表中插入 (sid, grade, dname, cno, sectno) 为 (1000, 4, ’Computer Sciences’, 302, 1) 的选课记录，其中学生信息（sid=1000）不存在，违反了引用完整性约束。

SQL语句（见Insert5）：

1. INSERT INTO Enroll
2. VALUES (1000, 4, 'Computer Sciences', 302, 1);

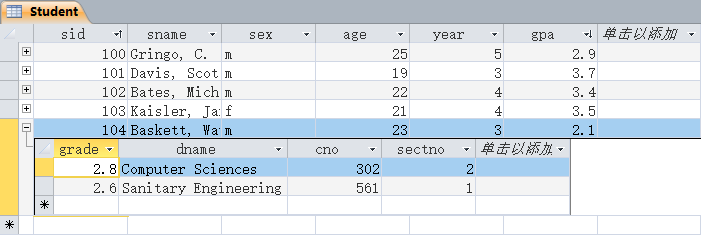
运行结果：



插入记录失败。（貌似所有INSERT报错都一样，不知道为什么）

### 2.4.2 尝试12：级联删除

如图所示，sid=104的学生选了两门课。

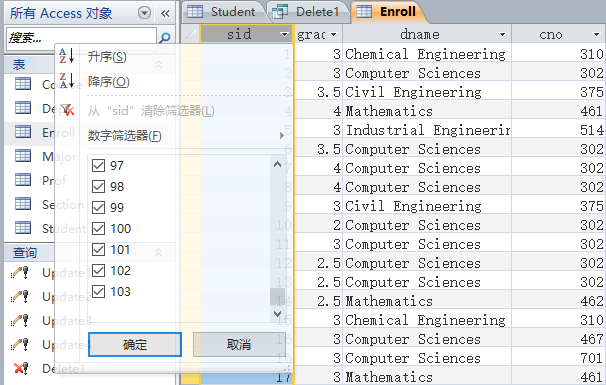


现在尝试删除sid=104的学生，观察Enroll表的变化情况。

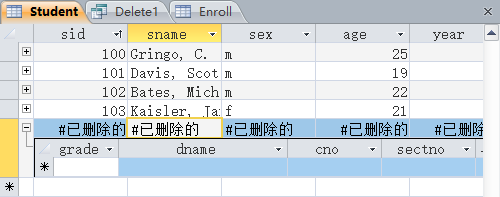
SQL语句（见Delete1）：

1. DELETE from Student WHERE sid=104;

Enroll表的结果：



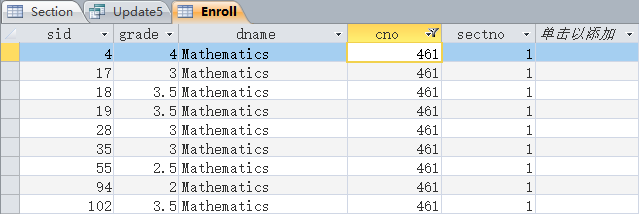
Student表的结果：



可以发现，关联的选课记录也被删除了，这是级联删除的功能。

### 2.4.3 尝试13：级联更新

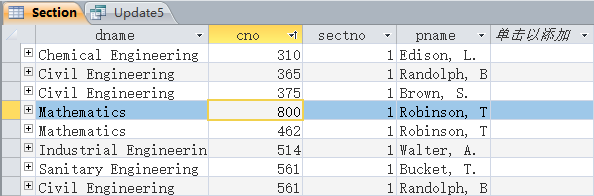
尝试将Course表中dname=Mathematics、cno=461、sectno=1的记录修改为cno=800。该课程的学生选课信息如下。



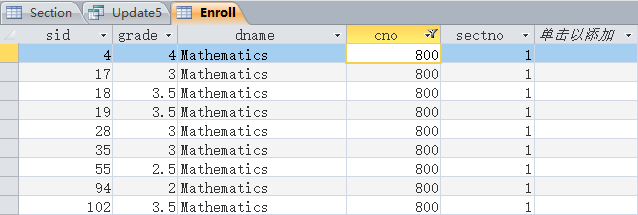
SQL语句（见Update5）：

1. UPDATE Section SET cno = 800
2. WHERE dname='Mathematics' AND cno=461 AND sectno=1;

运行结果：



执行成功，该课程的cno变为了800。

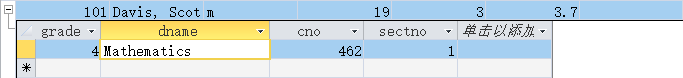


Enroll表中的选课记录也更新为cno=800。这是级联更新的功能。

### 2.4.4 尝试14：级联删除失败

我们编辑关联的属性，将“级联更新相关字段”、“级联删除相关记录”勾选取消。

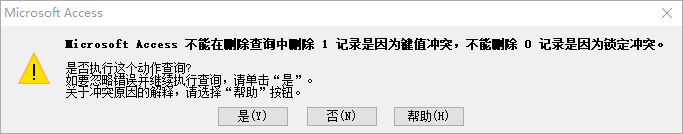
尝试删除sid=101的学生。sid=101的学生选了一门课，如图所示。



SQL语句（见Delete3）：

1. DELETE \*
2. FROM Student
3. WHERE sid=101;

运行结果：



删除记录失败。因为目前的关系不支持级联删除功能，而sid=101的学生存在选课记录，因此不能删除。

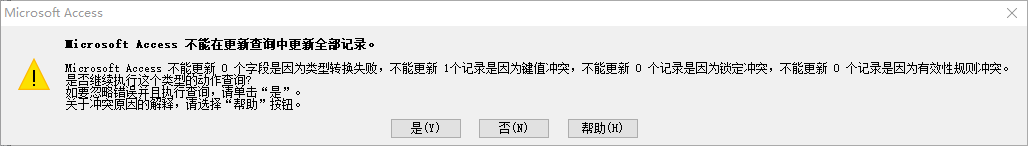
### 2.4.5 尝试15：级联更新失败

尝试将sid=101的学生更新为sid=6000。

SQL语句（见Update7）：

1. UPDATE Student SET sid = 6000
2. WHERE sid=101;

运行结果：



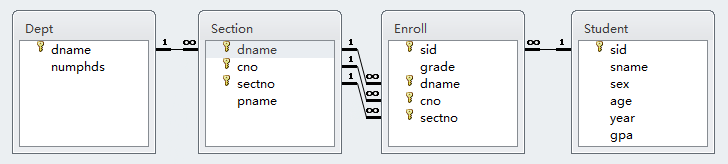
更新记录失败。因为目前的关系不支持级联更新功能，而sid=101的学生存在选课记录，因此不能更新。

### 2.4.4 尝试16：多表级联删除

进一步测试级联删除的功能。这里，我们测试Enroll-Section-Dept 3个表的级联删除。Enroll记录了选课的信息，引用Section的具体课程信息；Section记录了具体课程信息，其中每门课程都有开课学院，引用了Dept学院的信息。

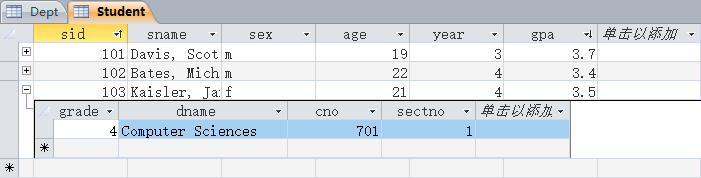
Dept表仅有一个主键dname。

建立关联如下：



现在我们操作Dept表，删除dname=’Computer Sciences’的记录。预计结果：所有与’Computer Sciences’专业相关的课程与选课记录均被删除。

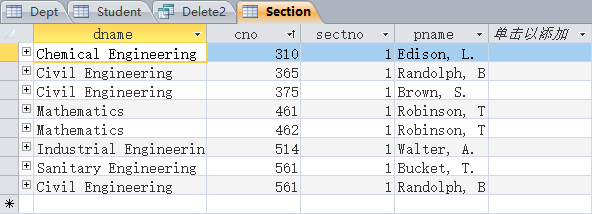
为了对比效果，我们先查看学生的选课记录。下图为sid=103学生的选课记录，这位学生仅选择了’Computer Sciences’专业的课程。删除’Computer Sciences’专业后，预计这名学生的选课记录将消失。



SQL语句（见Delete2）：

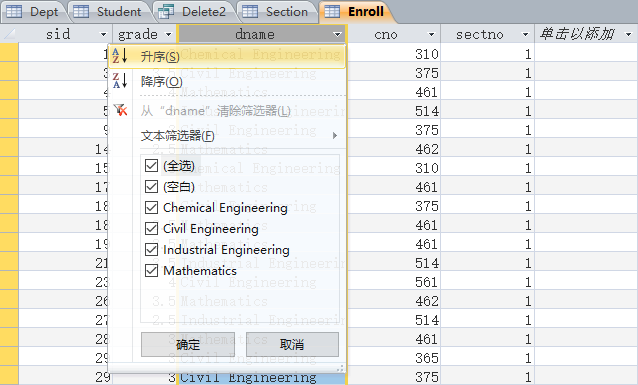
1. DELETE \* FROM Dept
2. WHERE dname='Computer Sciences';

查看Section表：



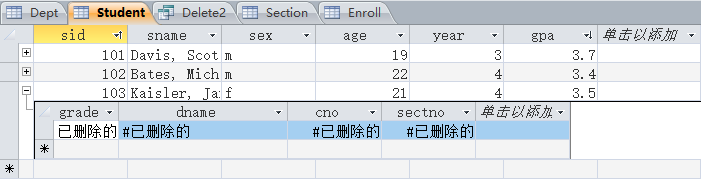
’Computer Sciences’专业的课程被删除了。

查看Enroll表：



’Computer Sciences’专业课程的选课记录被删除了。

查看sid=103学生的选课记录：



（出现这种情况，是因为删除过程中没有关闭Student表，否则sid=103学生的选课记录应该直接消失。）

果然，所有与’Computer Sciences’专业相关的课程与选课记录均被删除。

# 三、实验总结

通过本次实验，我练习了Access中的INSERT、UPDATE、DELETE操作，并通过这些操作考察Access是否遵循完整性约束，对了解三条完整性约束的理解更加深刻。

对于Access的引用完整性约束，我很好奇是否有SQL语句可以直接建立关联，而不是通过Access提供的GUI手动添加。查阅资料和博客后，我得知确实可以，语法如下：

1. *-- alter table 表名 add [constraint 外键名字] foreign key （外键字段） references 父表（主键字段）；*
2. ALTER TABLE t1 ADD CONSTRAINT k\_1 FOREIGN KEY (uid) REFERENCES test (id);
3. ALTER TABLE t1 ADD FOREIGN KEY (uid) REFERENCES test (id);

同样，域完整性约束也可以通过SQL语句添加，语法如下：

1. *-- 在”t1”表创建时为”Id”列创建检查约束，约束 “Id” 列必须只包含大于 0 的整数*
2. CREATE TABLE t1
3. (
4. Id int NOT NULL,
5. CHECK (Id>0)
6. )

主键也可以通过SQL语句规定：

1. create table TEST\_TBL (
2. *-- 主键（主键约束） primary key 唯一的约束该字段里面的数据，不能重复,不可为空*
3. *-- 可用于创建联合主键 primary key(参数列表，参数用","隔开)*
4. id int not null,
5. primary key (id)
6. );

（参考博客：<https://www.cnblogs.com/erma0-007/p/8635335.html>，<https://blog.csdn.net/Mr_PH/article/details/78151342>）