# 一. 实验目的

- 学习建立外键，以及利用FOREIGN KEY...REFERENCES子句以及各种约束保证参照完整性。

# 二. 实验内容

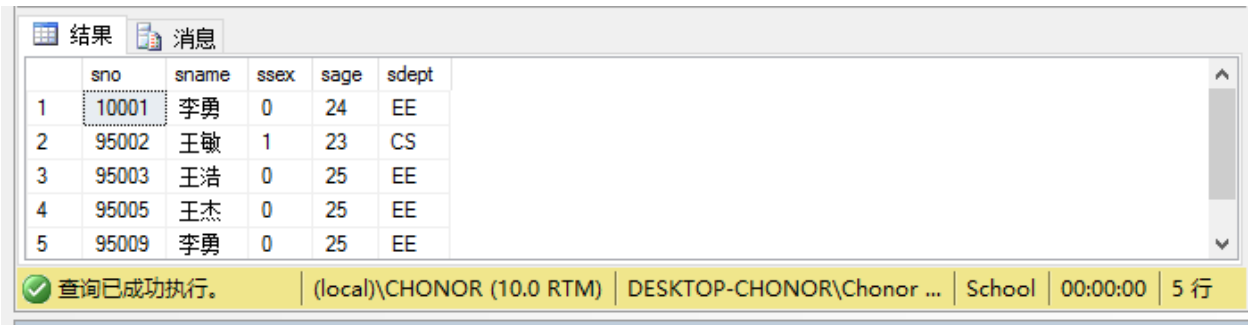
- 不违反参照完整性的插入数据示例
- 违反参照完整性的插入数据示例
- 级联删除
- 两张表的互相参照问题

# 三. 实验准备

- 在数据库 school中建立表 Stu_Union,设置sno为主键。

```
CREATE TABLE Stu_Union(sno CHAR(5) NOT NULL UNIQUE,
                       sname CHAR(8),
                       ssex CHAR(1),
                       sage INT,
                       sdept CHAR(20),
                       CONSTRAINT PK_Stu_Union PRIMARY KEY(sno));

INSERT INTO Stu_Union VALUES ('10001','李勇','0',24,'EE');
INSERT INTO Stu_Union VALUES ('95002','王敏','1',23,'CS');
INSERT INTO Stu_Union VALUES ('95003','王浩','0',25,'EE');
INSERT INTO Stu_Union VALUES ('95005','王杰','0',25,'EE');
INSERT INTO Stu_Union VALUES ('95009','李勇','0',25,'EE');
SELECT * FROM Stu_Union
```

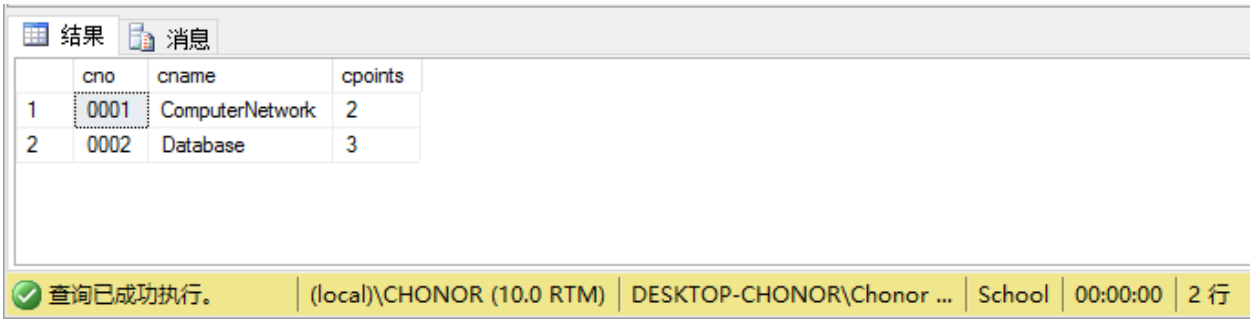
查询结果如下：




- 在数据库 school中建立表Course,令cno为主键。

```
CREATE TABLE Course(
    cno CHAR(4) NOT NULL UNIQUE,
    cname VARCHAR(50) NOT NULL,
    cpoints INT,
    CONSTRAINT PK PRIMARY KEY (cno)
);
INSERT Course VALUES ('0001','ComputerNetwork',2);
INSERT Course VALUES ('0002','Database',3);
SELECT * FROM Course
```

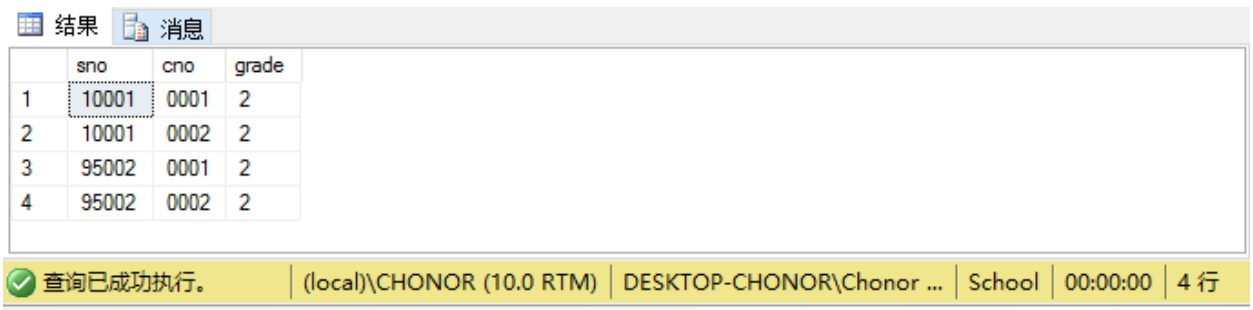
查询结果如下：




- 建立表SC，令sno和cno分别为参照Stu_Union表以及Course表的外键，设定为级联删除，并令（sno，cno）为其主键。在不违反参照完整性的前提下，插入数据。

```
CREATE TABLE SC(
    sno CHAR(5) REFERENCES Stu_Union(sno) ON DELETE CASCADE,
    cno CHAR(4) REFERENCES Course(cno) ON DELETE CASCADE,
    grade INT,
    CONSTRAINT PK_SC PRIMARY KEY(sno,cno)
);
INSERT INTO SC VALUES ('95002','0001',2);
INSERT INTO SC VALUES ('95002','0002',2);
INSERT INTO SC VALUES ('10001','0001',2);
INSERT INTO SC VALUES ('10001','0002',2);
SELECT * FROM SC
```

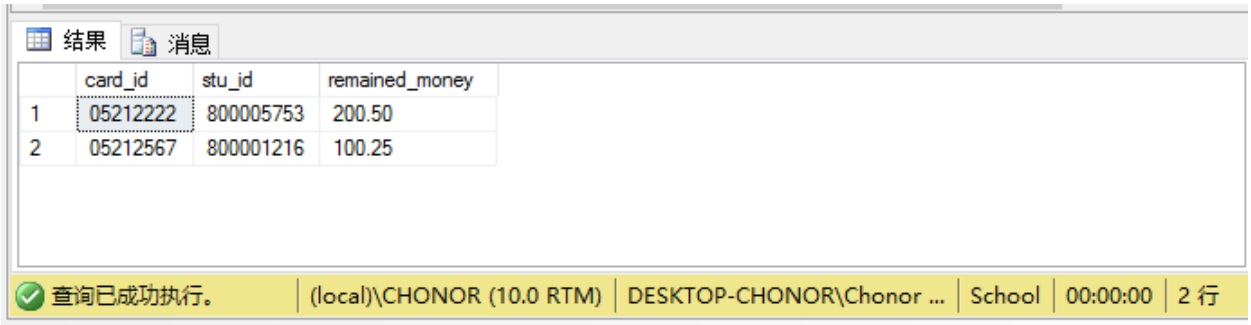
查询结果如下：




- 建立Stu_Card表，令card_id为主键，并令stu_id 为参照student表的外键，并插入数据。

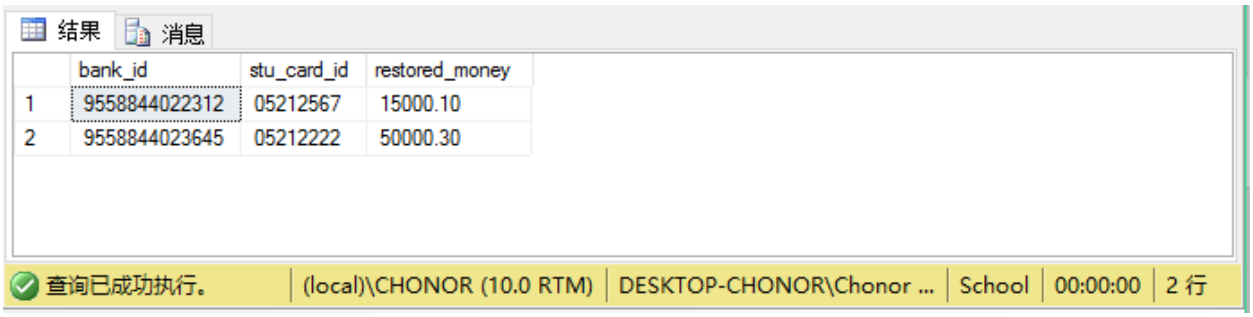
```
CREATE TABLE Stu_Card(
    card_id CHAR(14),
    stu_id CHAR(10)REFERENCES students(sid) ON DELETE CASCADE,
    remained_money DECIMAL(10,2),
    CONSTRAINT PK_stu_card PRIMARY KEY(card_id)
);
INSERT INTO Stu_Card VALUES ('05212567','800001216',100.25);
INSERT INTO Stu_Card VALUES ('05212222','800005753',200.50);
SELECT * FROM Stu_Card;
```

查询结果如下：




- 建立表ICBC_Card表，令card_id为主键，令stu_card_id为参照Stu_Card表的外键，并插入数据

查询结果如下：



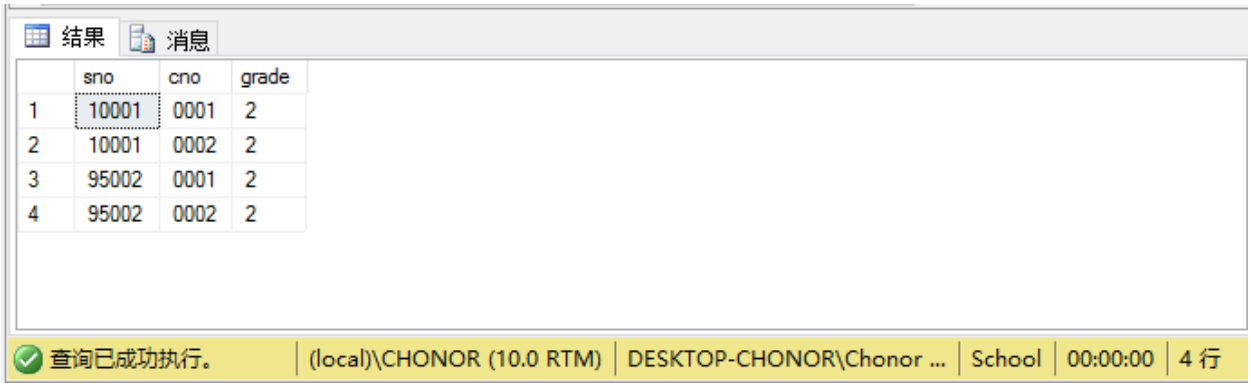

# 三. 实验结果

1. **用alter table语句将SC表中的on delete cascade改为on delete no action,重新插入SC的数据（按照实验一）。再删除Stu_Union中sno为'10001'的数据。观察结果，并分析原因。**

```
 ALTER TABLE SC
    DROP CONSTRAINT FK__SC__cno__4222D4EF;
ALTER TABLE SC
    DROP CONSTRAINT FK__SC__sno__412EB0B6;
ALTER TABLE SC
    ADD CONSTRAINT FK__SC__sno FOREIGN KEY(sno) REFERENCES Stu_Union(sno) ON DELETE NO
ACTION;
ALTER TABLE SC
    ADD CONSTRAINT FK__SC__cno FOREIGN KEY(cno) REFERENCES Course(cno) ON DELETE NO ACTION;
DELETE  FROM SC

INSERT INTO SC VALUES ('95002','0001',2);
INSERT INTO SC VALUES ('95002','0002',2);
```

```
INSERT INTO SC VALUES ('10001','0001',2);
INSERT INTO SC VALUES ('10001','0002',2);
SELECT * FROM SC
```

查询结果如下：




```
DELETE
FROM Stu_Union
WHERE sno='10001'
```

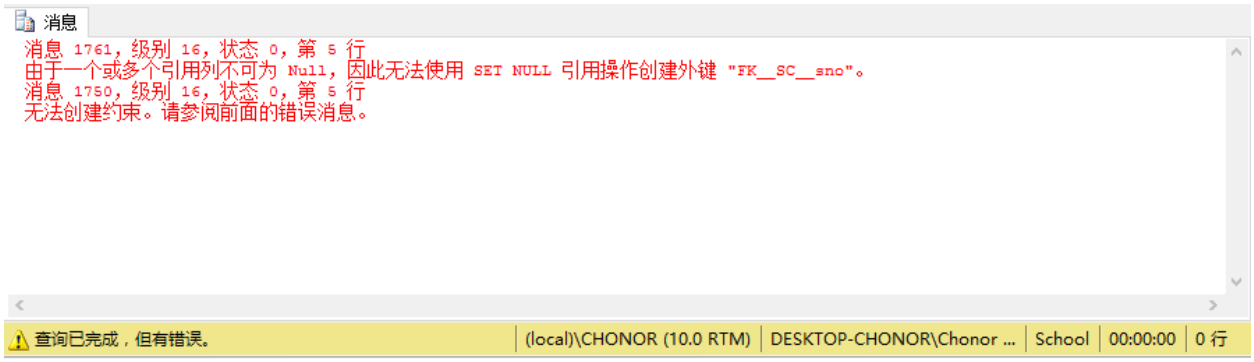
查询结果如下：


此时的外键属性是On delete no action，即当从表中有匹配的记录时，主表中相应的候选键不允许update/delete操作，所以此处删除报错。

2. **用alter table语句将SC表中的on delete no action改为on delete set NULL,重新插入SC的数据（按照实验一）。再删除Stu_Union中sno为'10001'的数据。观察结果，并分析原因**

```
ALTER TABLE SC
    DROP CONSTRAINT FK__SC__cno;
ALTER TABLE SC
    DROP CONSTRAINT FK__SC__sno;
ALTER TABLE SC
    ADD CONSTRAINT FK__SC__sno FOREIGN KEY(sno) REFERENCES Stu_Union(sno) ON DELETE SET
NULL;
ALTER TABLE SC
    ADD CONSTRAINT FK__SC__cno FOREIGN KEY(cno) REFERENCES Course(cno) ON DELETE SET NULL;
```

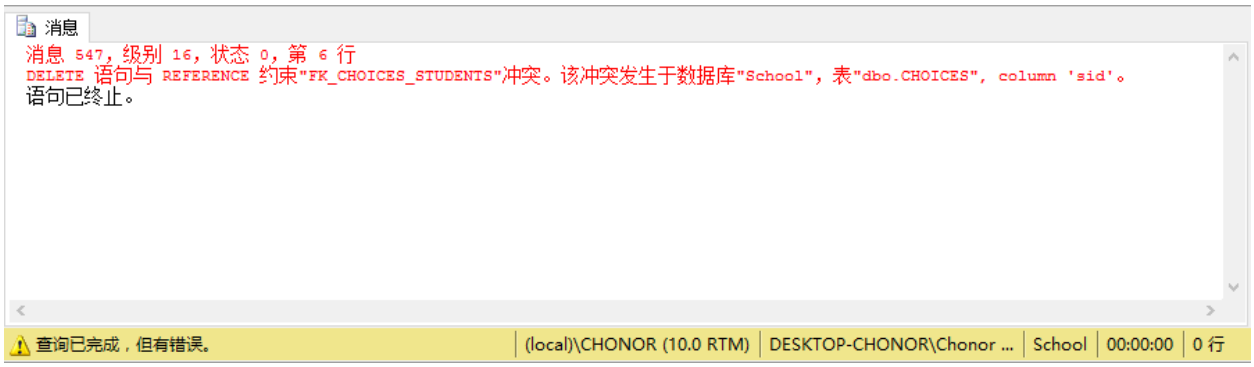
查询结果如下：




此时我们不能修改外键属性为 ON DELETE SET NULL，因为我们在定义sno和cno定义时了 NO NULL，所以我们此处ON DELETE SET NULL是删除后设置为NULL，但是其是不允许为NULL的，此时就会报错。

3. **建立事务T3，修改ICBC_Card表的外键属性，使其变为on delete set NULL,尝试删除students表中一条记录。观察结果，并分析原因**

```
BEGIN TRANSACTION T3
ALTER TABLE ICBC_Card
    DROP CONSTRAINT FK__ICBC_Card__stu_c__3D5E1FD2;
ALTER TABLE ICBC_Card
    ADD CONSTRAINT FK__ICBC_Card__stu_c FOREIGN KEY(stu_card_id) REFERENCES
Stu_Card(card_id) ON DELETE SET NULL;
DELETE
FROM students
WHERE sid='800001216'
COMMIT TRANSACTION T3
```

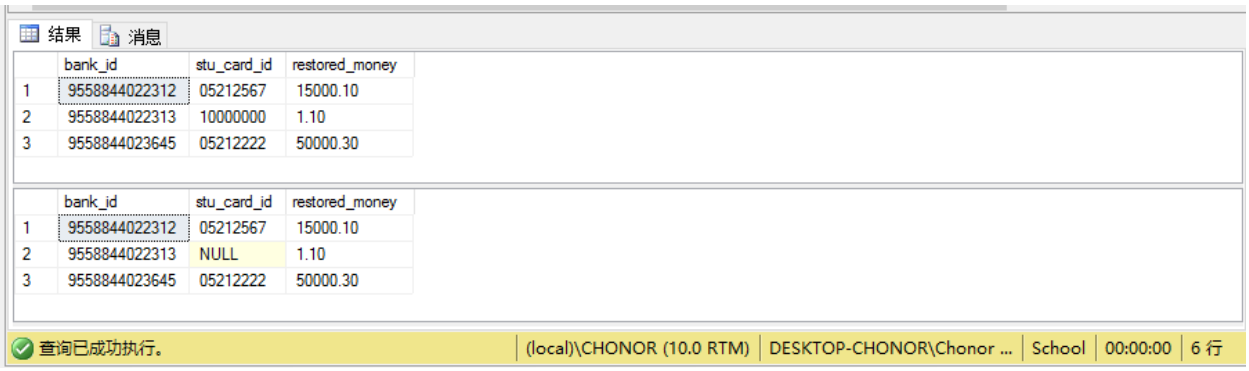



此时我们看到是删除失败的，但是问题不是出现在ICBC_Card的外键中，而是出现在原来数据库中的CHOICES，因为此时CHOICES和表STUDENTS还是存在外键约束的。此时整个事件回滚。

此时我们想要演示一下on delete set NULL 的效果就需要在一个不会影响到CHOICES和STUDENTS的值，此时我们可以向STUDENTS中插入一个全新的值，代码修改为下方。

```
BEGIN TRANSACTION T3
INSERT INTO STUDENTS VALUES ('1000001','a','b',1)
INSERT INTO Stu_Card VALUES ('10000000','1000001',100.25);
INSERT INTO ICBC_Card VALUES ('9558844022313','10000000',1.1);
SELECT * FROM ICBC_Card
ALTER TABLE ICBC_Card
    DROP CONSTRAINT FK__ICBC_Card__stu_c__3D5E1FD2;
ALTER TABLE ICBC_Card
```

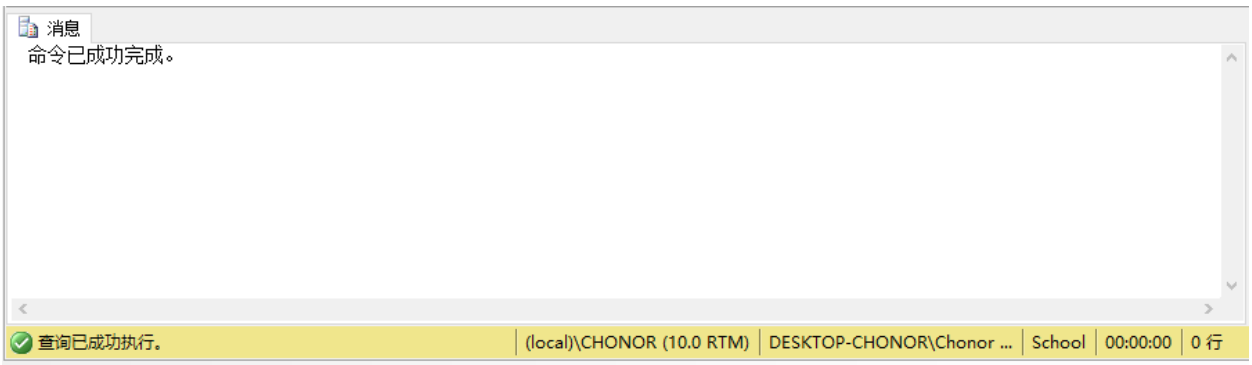
```
    ADD CONSTRAINT FK__ICBC_Card__stu_c FOREIGN KEY(stu_card_id) REFERENCES
Stu_Card(card_id) ON DELETE SET NULL;
DELETE
FROM students
WHERE sid='1000001'
SELECT * FROM ICBC_Card
COMMIT TRANSACTION T3
```




我们可以看到在删除了students中的一个学生后此时的ICBC_Card中对应学生的学生id变成了NULL。

4. **创建一个班里的学生互助表，规定：包括学生编号，学生姓名，学生的帮助对象，每个学生有且仅有一个帮助对象，帮助对象也必须是班里的学生。（表的自参照问题）**

```
CREATE TABLE Help(
    sid char(8)NOT NULL UNIQUE,
    sname VARCHAR(50),
    hid char(8),
    CONSTRAINT PK_help PRIMARY KEY(sid)
);
ALTER TABLE Help
    ADD CONSTRAINT FK_help FOREIGN KEY(hid) REFERENCES Help(sid) On delete no action;
```

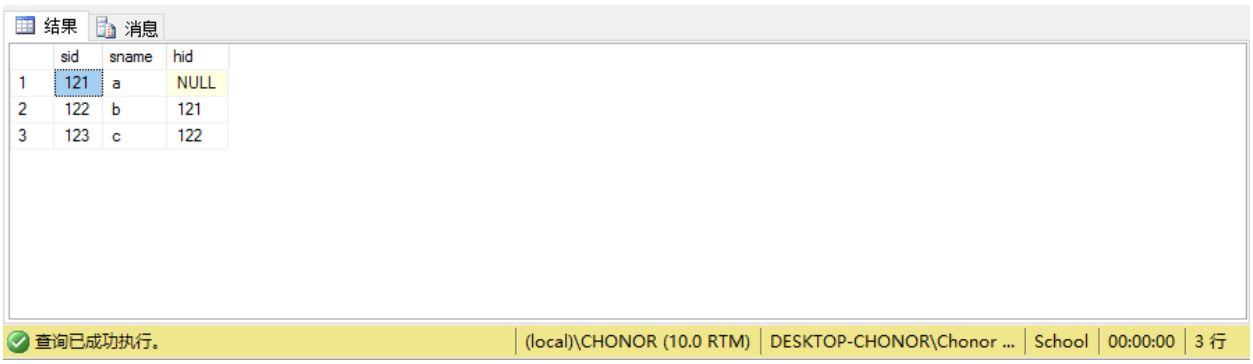



测试一下进行数据插入和删除

```
insert into Help values('121','a',NULL);
insert into Help values('122','b','121');
insert into Help values('123','c','122');
insert into Help values('124','d',NULL);

SELECT * FROM Help
delete FROM Help WHERE sid='121'
```


此时由于表的自参照问题，121数据还在表中还存在匹配记录所以不能删除。

```
delete FROM Help WHERE sid='124'
SELECT * FROM Help
```

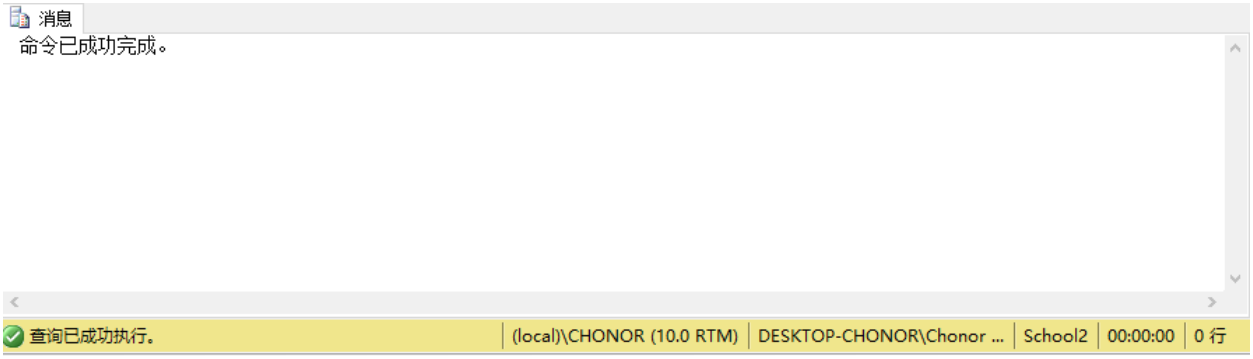



此时我们可以删除sid=124的记录因为表中无匹配的记录，所以可以以正常删除。

5. **学校学生会的每个部门都有一个部长，每个部长领导多个部员，每个部只有一个部员有评测部长的权利，请给出体现这两种关系（领导和评测）的两张互参照的表的定义。（两个表互相参照的问题）**

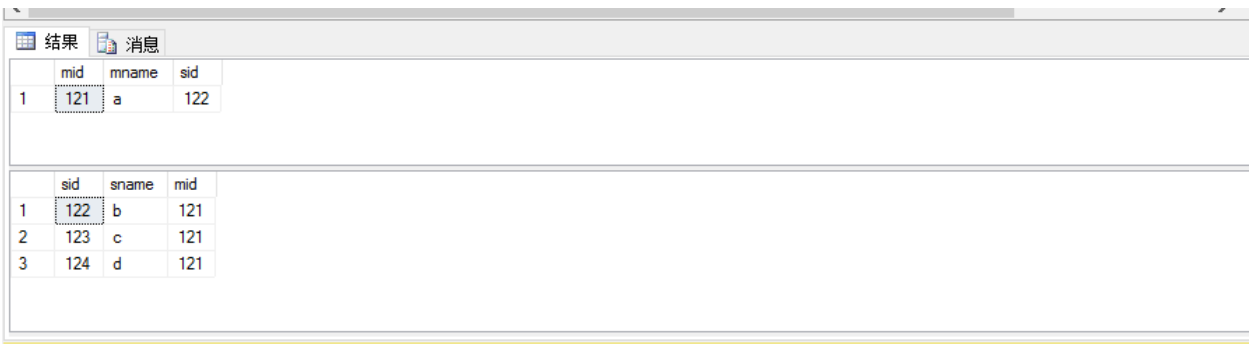
```
CREATE TABLE Minister(
    mid char(8) NOT NULL UNIQUE,
    mname VARCHAR(50),
    sid char(8),
```

```
    CONSTRAINT PK_Minister PRIMARY KEY(mid)
);

CREATE TABLE Staff(
    sid char(8) NOT NULL UNIQUE,
    sname VARCHAR(50),
    mid char(8),
    CONSTRAINT PK_staff PRIMARY KEY(sid),
    CONSTRAINT FK_staff FOREIGN KEY(mid)  REFERENCES Minister(mid)
);

ALTER TABLE Minister
    ADD CONSTRAINT FK_minister FOREIGN KEY(sid) REFERENCES Staff(sid);
```




增加数据测试

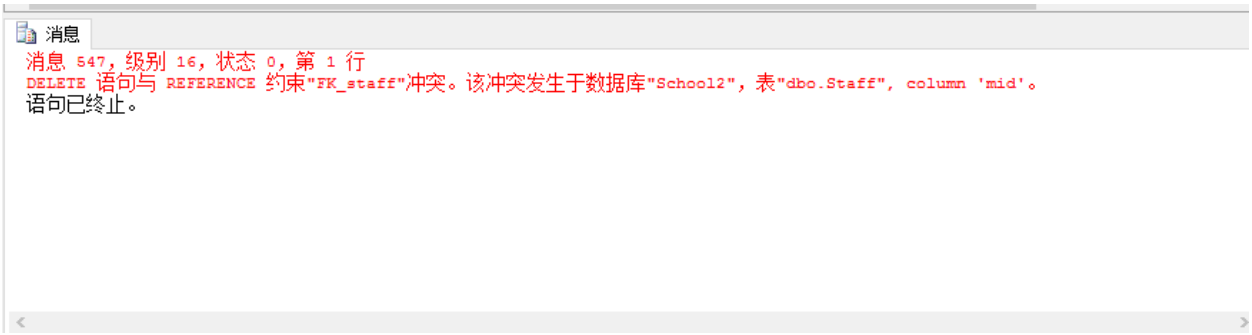
```
insert into Minister values('121','a',NULL);
insert into Staff values('122','b','121');
insert into Staff values('123','c','121');
insert into Staff values('124','d','121');
update Minister
set sid='122'
WHERE mid='121'
SELECT * FROM Minister
SELECT * FROM Staff
```




```
DELETE Staff WHERE sid='122'
```


此时因为外键约束，也就是两个表互相参照的问题，此时Minister表中还有sid=122，所以无法删除。同理一下语句也无法执行

```
DELETE Minister WHERE mid='121'
```




# 四. 实验感想

这次实验主要还是对于完整性，通过实验自己验证当SQL语句执行时，完整性的限制，同时也学习了如何进行表的自参照和两个的互参照的做法，还有如何修改外键，以及外键的设置会如何对表的完整性产生影响。实验中也发现了外键和主键之间的设置要相同不然也是不满足完整性的例如第二个任务。