实验报告

姓名：胡瑞康
学号：22336087

数据库表结构

- **students (sid, sname, email, grade)**
  学生表，包含学生的编号、姓名、邮箱和年级。
- **teachers (tid, tname, email, salary)**
  教师表，包含教师的编号、姓名、邮箱和薪水。
- **courses (cid, cname, hour)**
  课程表，包含课程的编号、名称和课时。
- **choices (no, sid, tid, cid, score)**
  选课表，包含选课记录的编号、学生编号、教师编号、课程编号和成绩。

实验准备（建表操作）

在数据库 school中建立表 Stu_Union,设置sno为主键。建立表Course,令cno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');
```

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 INTO Course VALUES ('0001', 'ComputerNetworks', 2);
INSERT INTO Course VALUES ('0002', 'Database', 3);
```

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。

```
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)
);
```

插入数据

```
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);
```

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。再建立表ICBC_Card表，令card_id为主键，令stu_card_id为参照Stu_Card表的外键

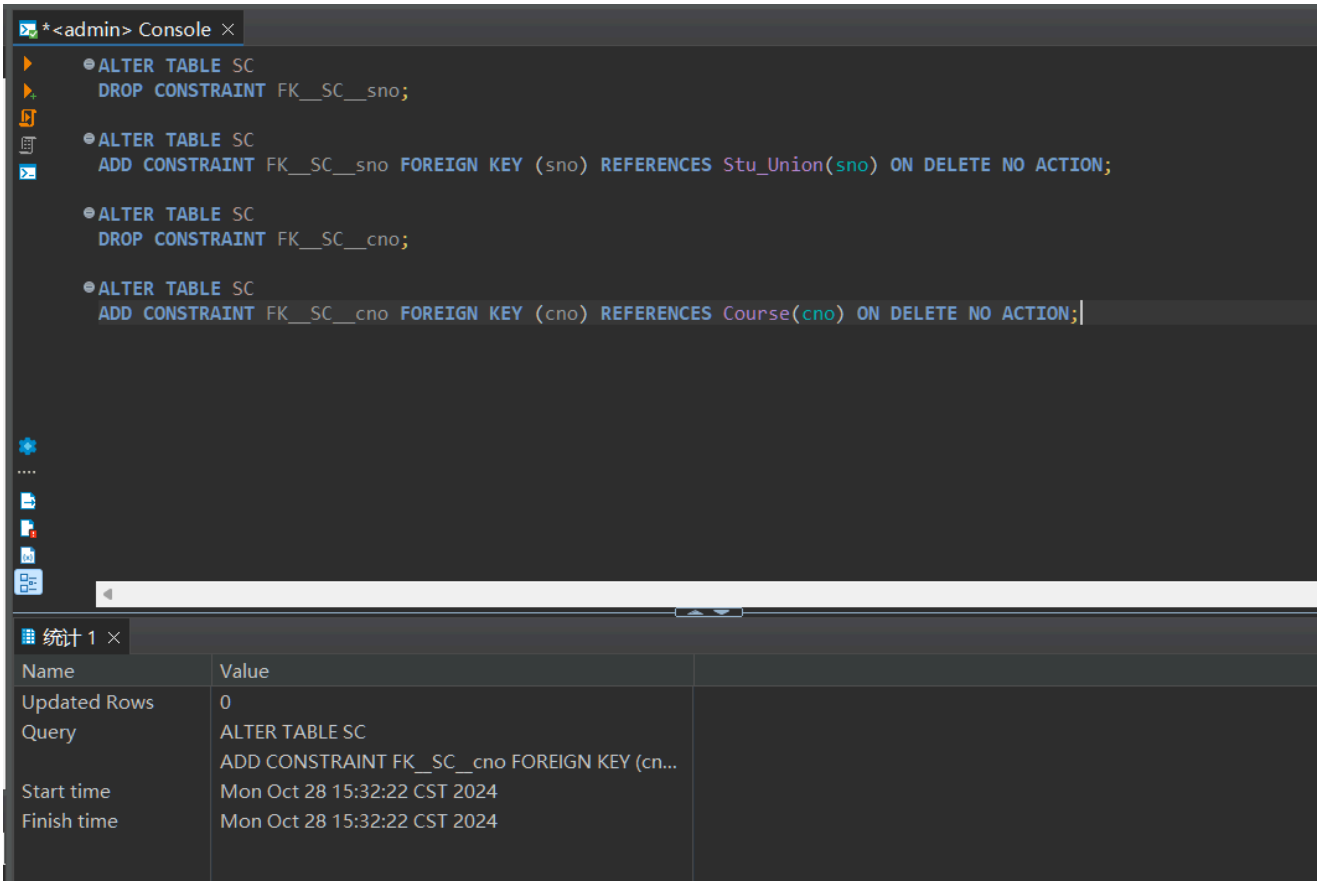
```
CREATE TABLE Stu_Card (
    card_id CHAR(14) NOT NULL,
    stu_id CHAR(10) REFERENCES students(sid) ON DELETE CASCADE,
    remained_money DECIMAL,
    CONSTRAINT PK_stu_card PRIMARY KEY(card_id)
);
insert into Stu_Card values('05212222','800005753',200.50);
insert into Stu_Card values('05212567','800001216',100.25);
```

建立表ICBC_Card表

```
CREATE TABLE ICBC_Card (
    bank_id CHAR(20),
    stu_card_id CHAR(14) REFERENCES Stu_Card(card_id) ON DELETE CASCADE,
    restored_money DECIMAL(10,2),
    CONSTRAINT PK_Icbc_card PRIMARY KEY(bank_id)
);
insert into ICBC_Card values('9558844022312','05212567',15000.1);
insert into ICBC_Card values('9558844023645','05212222',50000.3);
```

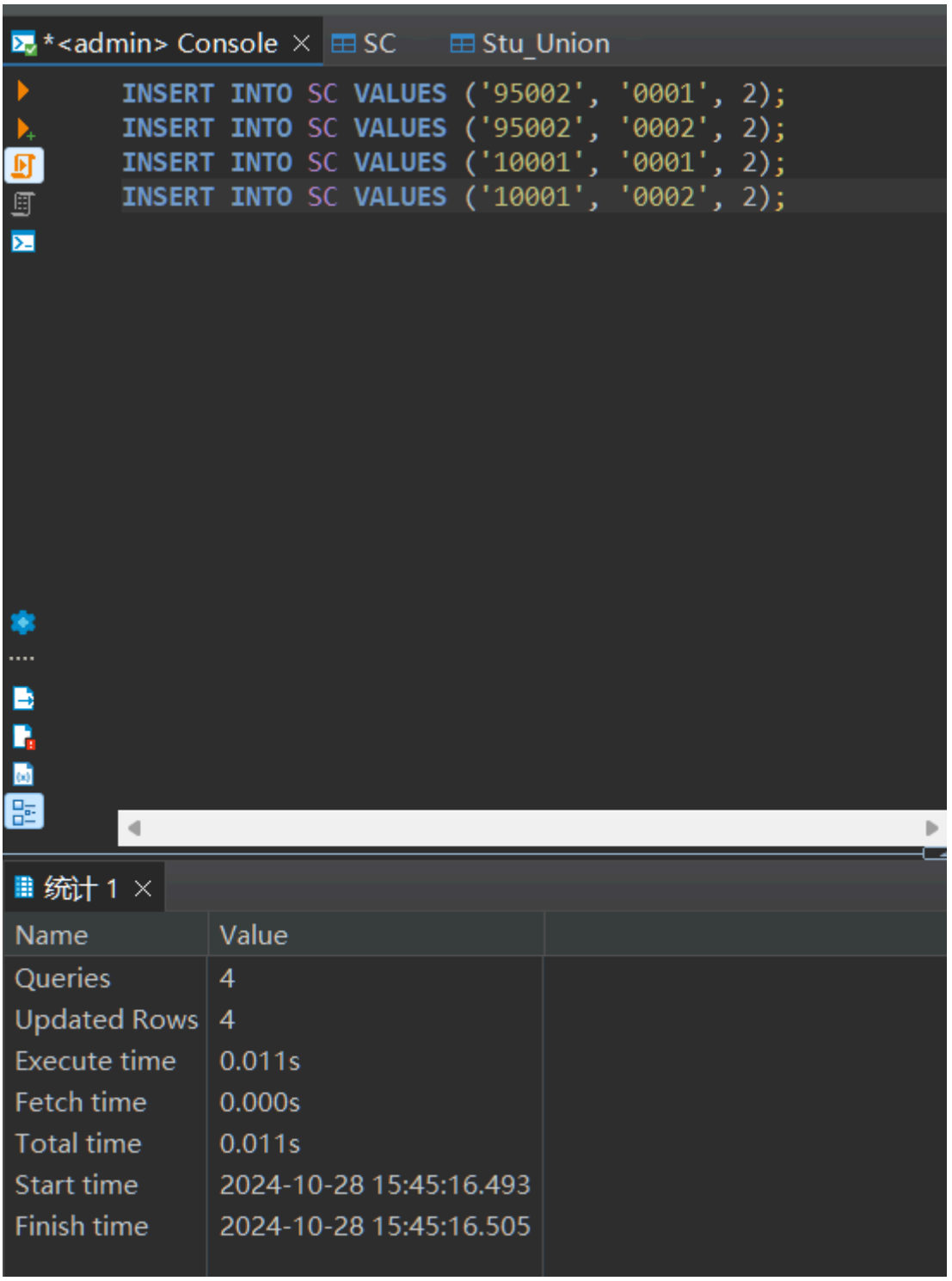
用alter table语句将SC表中的on delete cascade改为on delete no action,重新插入SC的数据（按照实验一）。再删除Stu_Union中sno为‘10001’的数据。观察结果，并分析原因。

```
ALTER TABLE SC
DROP CONSTRAINT FK__SC__sno;

ALTER TABLE SC
ADD CONSTRAINT FK__SC__sno FOREIGN KEY (sno) REFERENCES Stu_Union(sno) ON DELETE NO ACTION;

ALTER TABLE SC
DROP CONSTRAINT FK__SC__cno;

ALTER TABLE SC
ADD CONSTRAINT FK__SC__cno FOREIGN KEY (cno) REFERENCES Course(cno) ON DELETE NO ACTION;
```



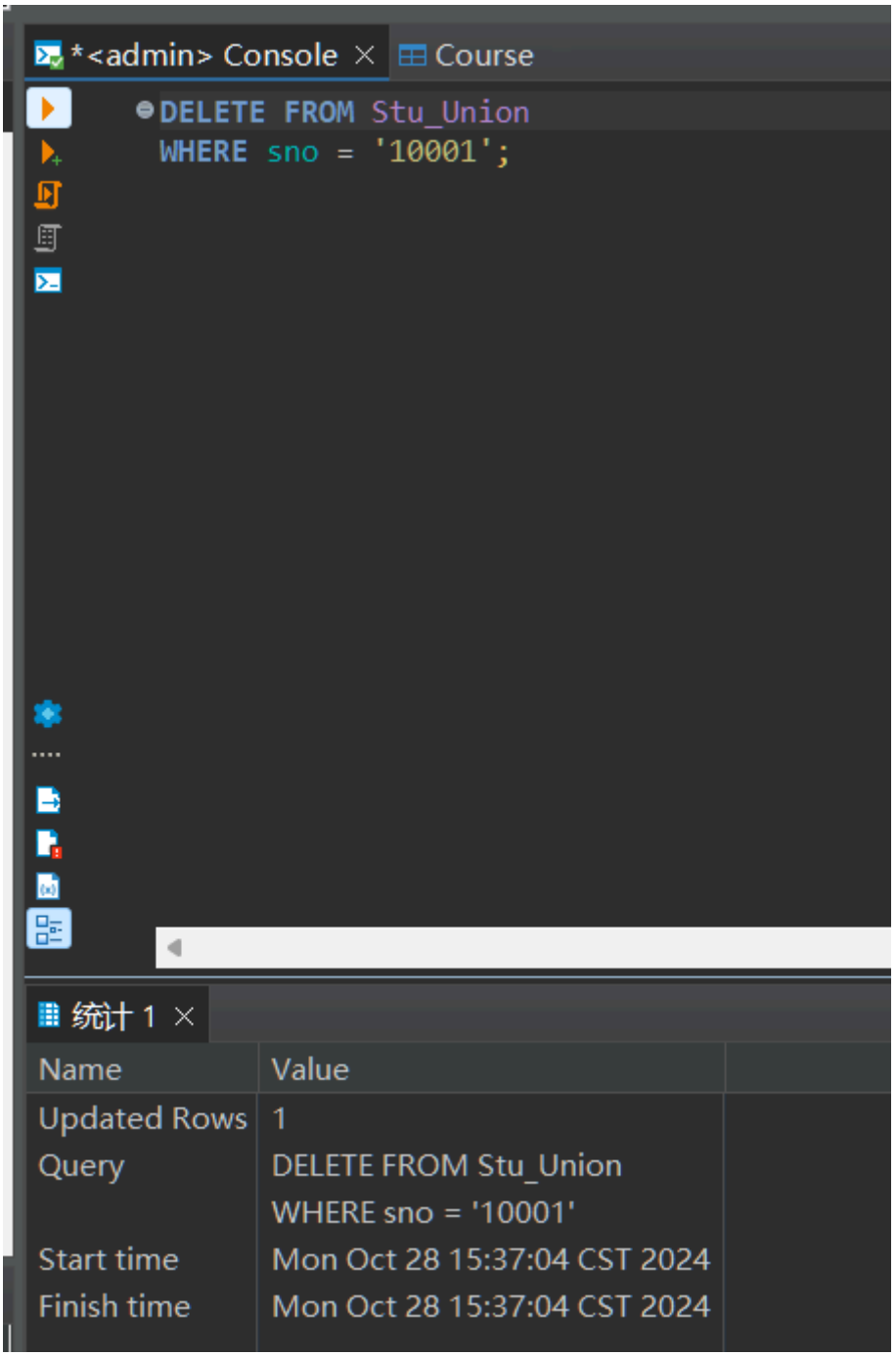
重新插入数据

```
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);
```

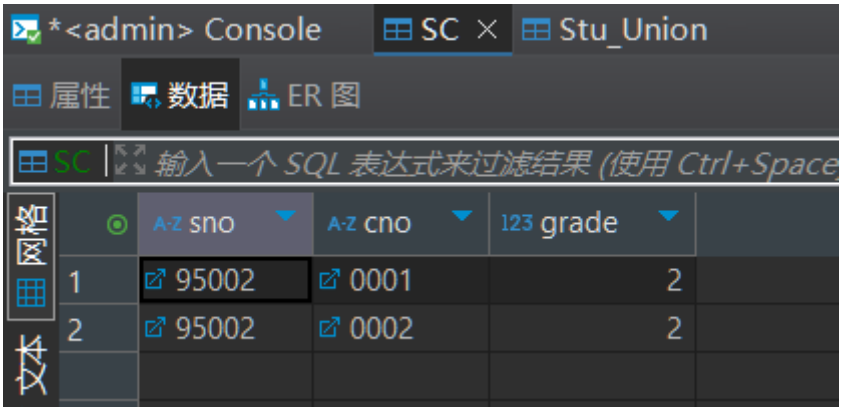


删除Stu_Union中sno为‘10001’的数据

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



观察发现SC表中的sno为‘10001’的数据也被删除了。



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。

由于主键限制，需要先去除主键，否则无法set NULL.

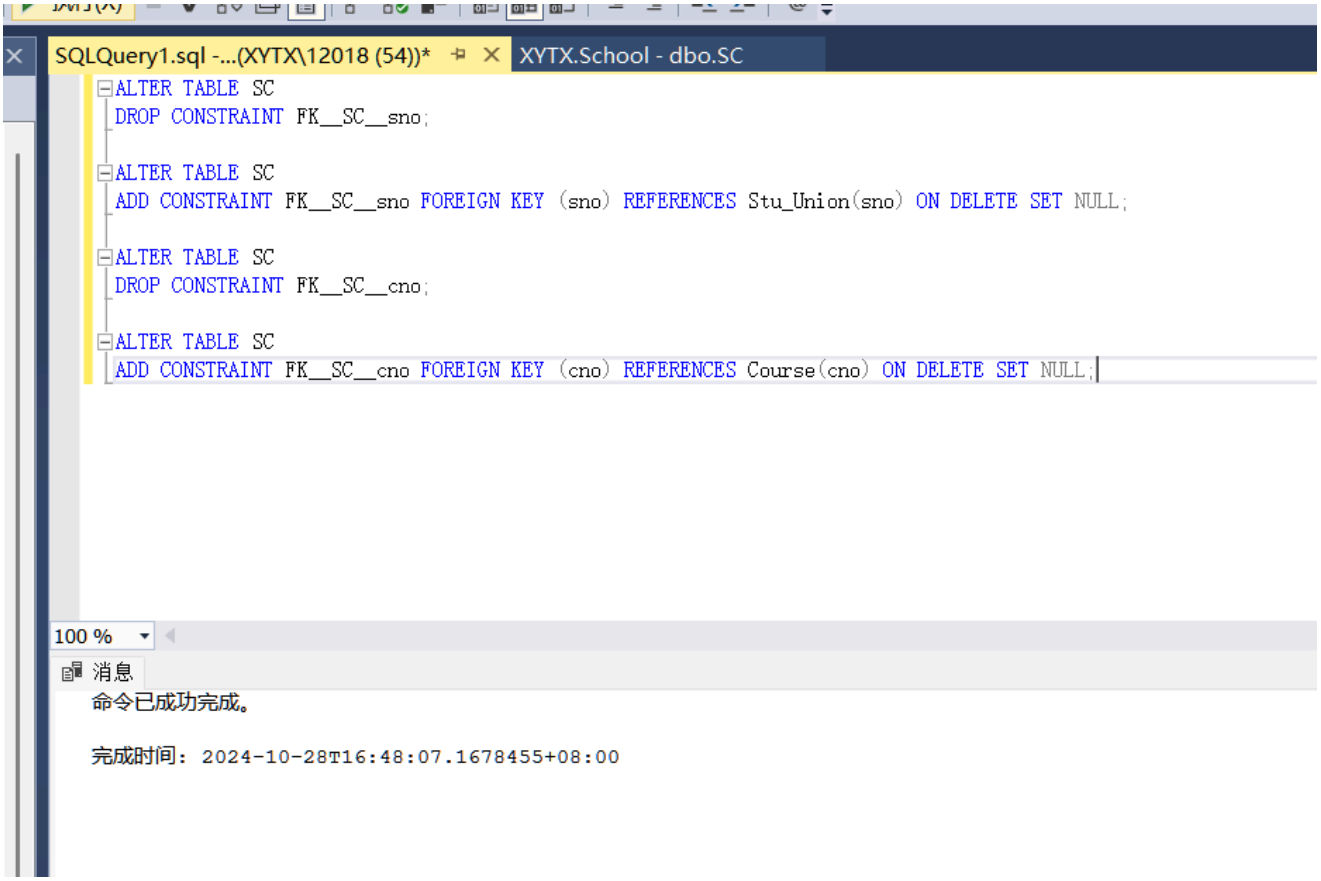
```
ALTER TABLE SC
DROP CONSTRAINT PK_SC;
```

修改允许NULL

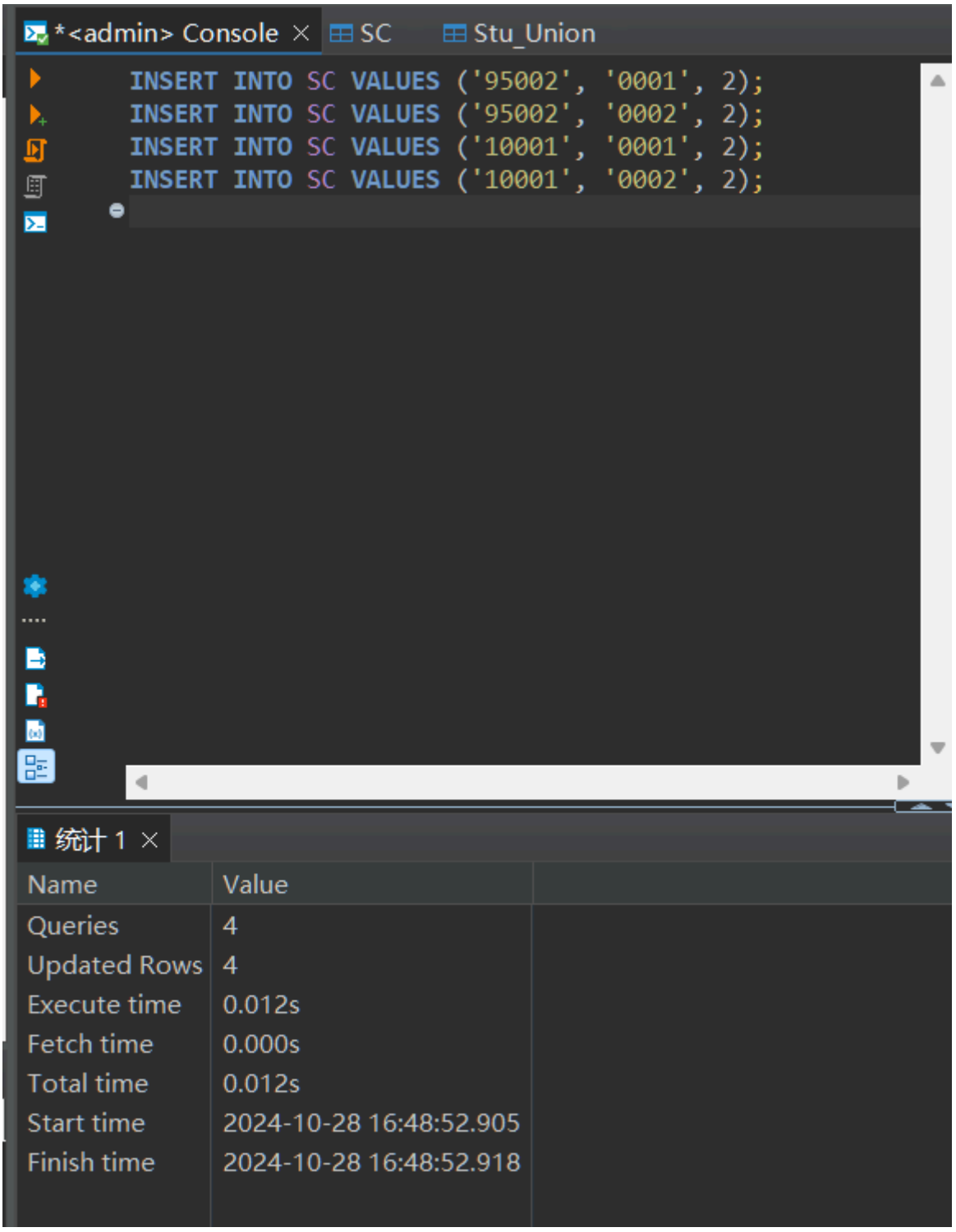
```
ALTER TABLE SC
ALTER COLUMN sno CHAR(5) NULL;

ALTER TABLE SC
ALTER COLUMN cno CHAR(4) NULL;
```

修改on delete set NULL

```
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
DROP CONSTRAINT FK__SC__cno;

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

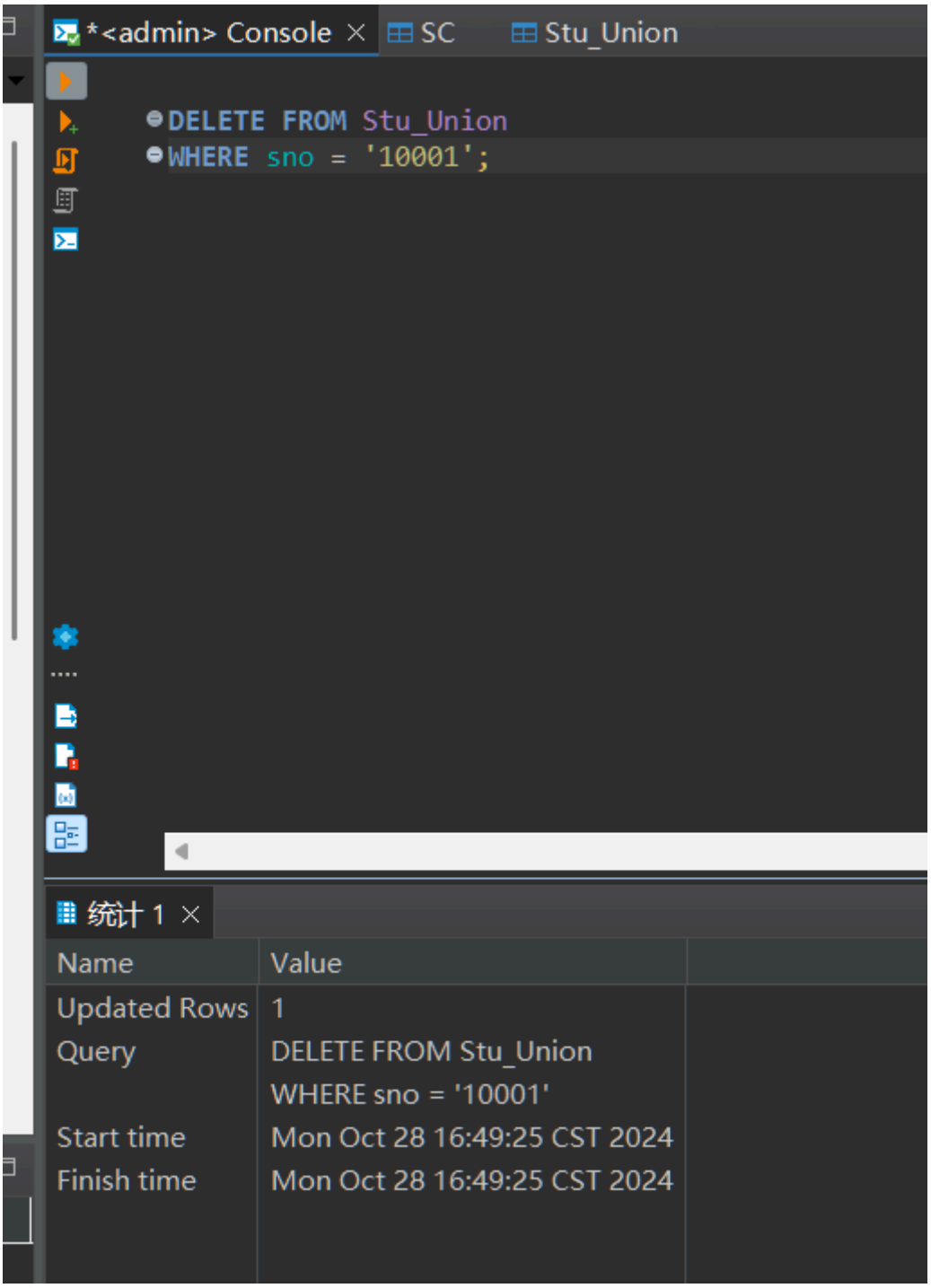

SQLQuery1.sql -...(XYTX\12018 (54))*
XYTX.School - dbo.SC
ALTER TABLE SC
DROP CONSTRAINT FK__SC__sno;
ALTER TABLE SC
ADD CONSTRAINT FK__SC__sno FOREIGN KEY (sno) REFERENCES Stu_Union(sno) ON DELETE SET NULL;
ALTER TABLE SC
DROP CONSTRAINT FK__SC__cno;
ALTER TABLE SC
ADD CONSTRAINT FK__SC__cno FOREIGN KEY (cno) REFERENCES Course(cno) ON DELETE SET NULL;
100 %
消息
命令已成功完成。
完成时间：2024-10-28T16:48:07.1678455+08:00

重新插入

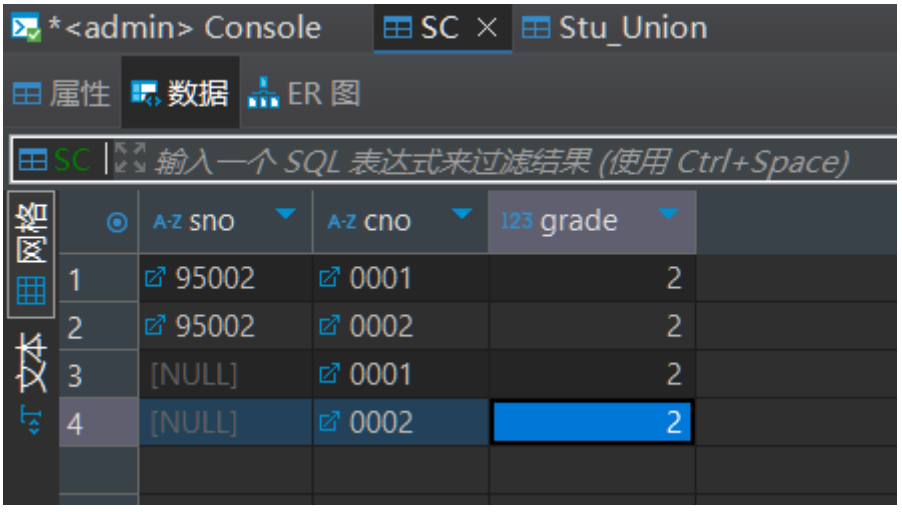


再删除

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

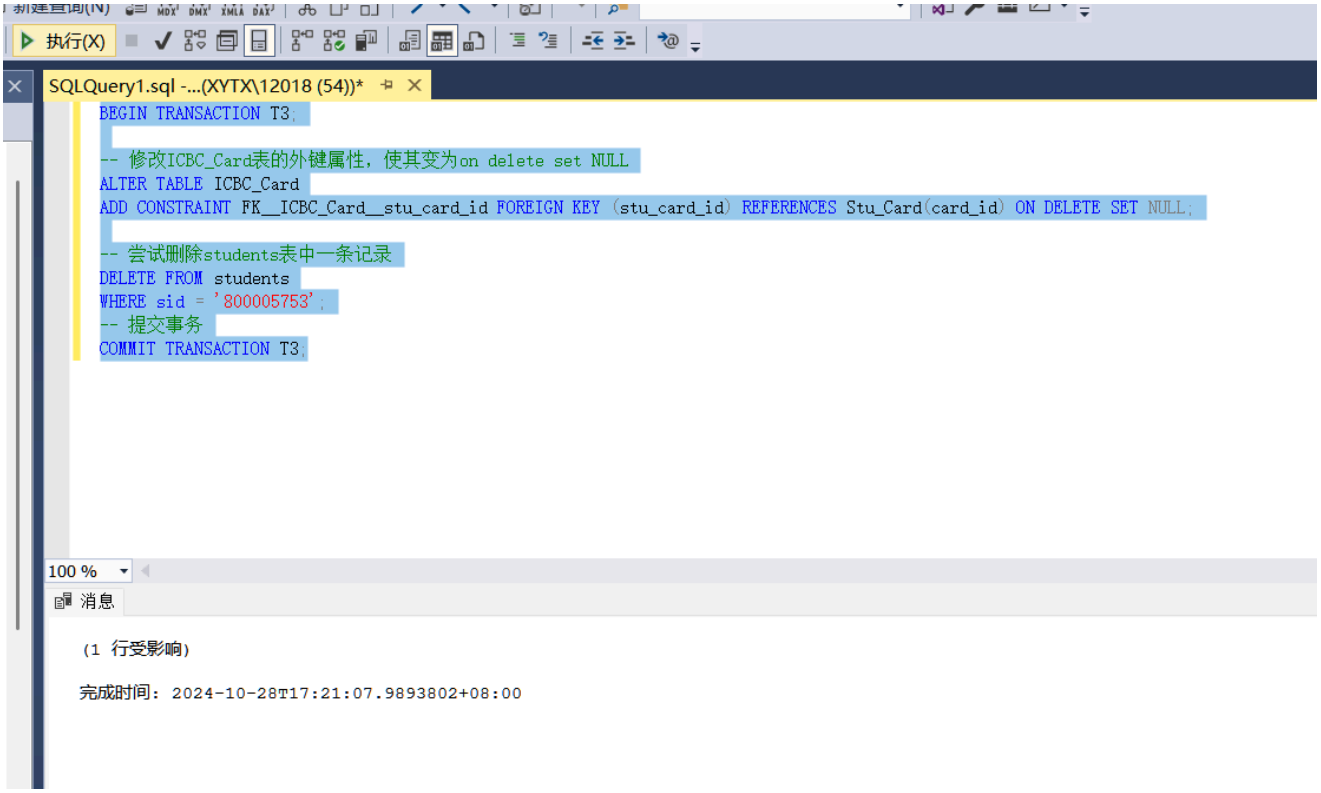


发现SC表中的sno为‘10001’的数据没有被删除，但是sno为NULL。

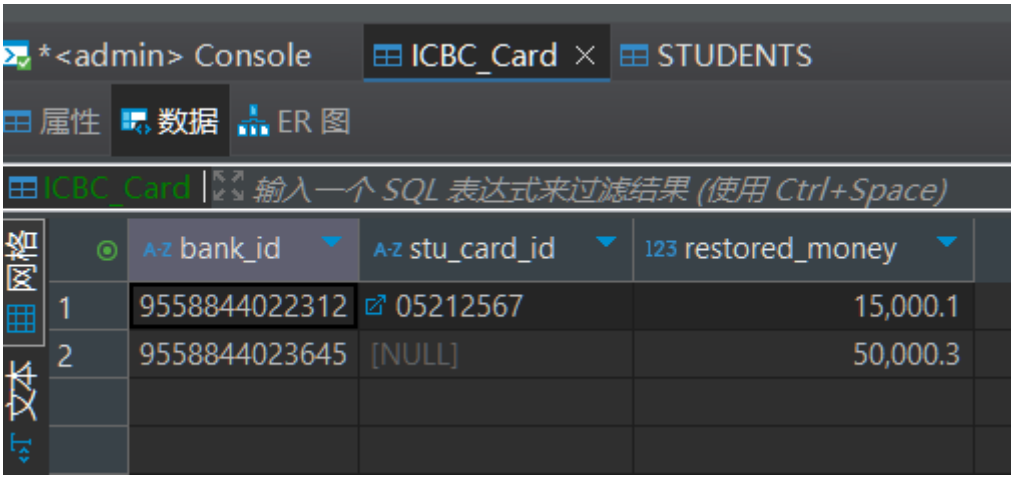


建立事务T3，修改ICBC_Card表的外键属性，使其变为on delete set NULL,尝试删除students表中一条记录。观察结果，并分析原因。

```sql
BEGIN TRANSACTION T3;

-- 修改ICBC_Card表的外键属性，使其变为on delete set NULL
ALTER TABLE ICBC_Card
ADD CONSTRAINT FK__ICBC_Card__stu_card_id FOREIGN KEY (stu_card_id) REFERENCES Stu_Card(card_id)
ON DELETE SET NULL;

-- 尝试删除students表中一条记录
DELETE FROM students
WHERE sid = '800005753';
-- 提交事务
COMMIT TRANSACTION T3;
```



结果：ICBC_Card相关记录的stu_card_id字段被设置为NULL。

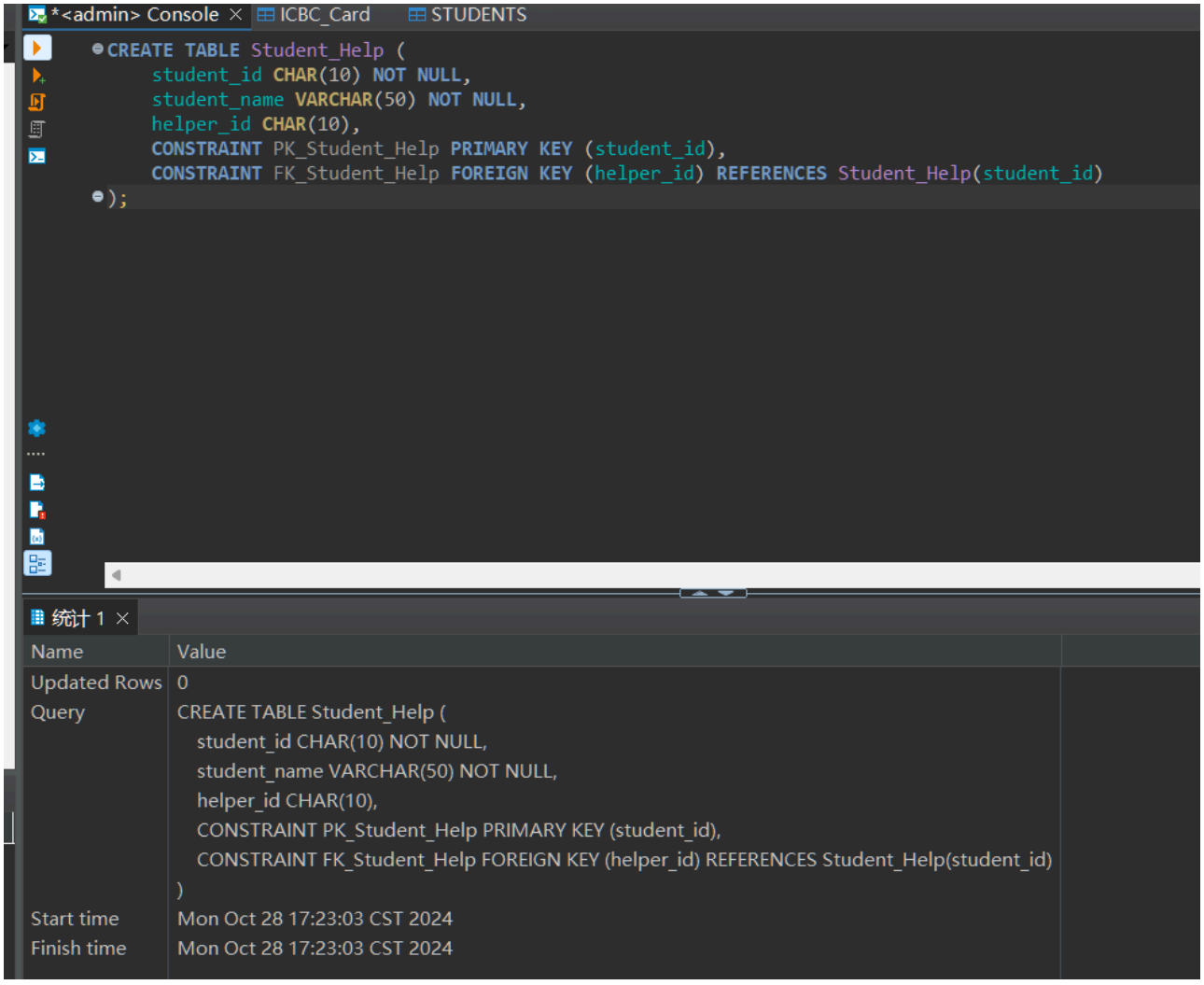


创建一个班里的学生互助表，规定：包括学生编号，学生姓名，学生的帮助对象，每个学生有且仅有一个帮助对象，帮助对象也必须是班里的学生。（表的自参照问题）

```
CREATE TABLE Student_Help (
    student_id CHAR(10) NOT NULL,
    student_name VARCHAR(50) NOT NULL,
    helper_id CHAR(10),
    CONSTRAINT PK_Student_Help PRIMARY KEY (student_id),
    CONSTRAINT FK_Student_Help FOREIGN KEY (helper_id) REFERENCES Student_Help(student_id)
);
```

解释：

- `student_id`：学生的编号，作为主键。
- `student_name`：学生的姓名。
- `helper_id`：学生的帮助对象的编号，作为外键，参照`Student_Help`表的`student_id`字段。
- `CONSTRAINT FK_Student_Help FOREIGN KEY (helper_id) REFERENCES Student_Help(student_id)`：确保`helper_id`必须是`Student_Help`表中存在的`student_id`。



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）

成员表记录每个成员的编号、姓名、所属部门编号以及是否有评测部长的权利。部门编号作为外键，参照部门表中的部门编号。

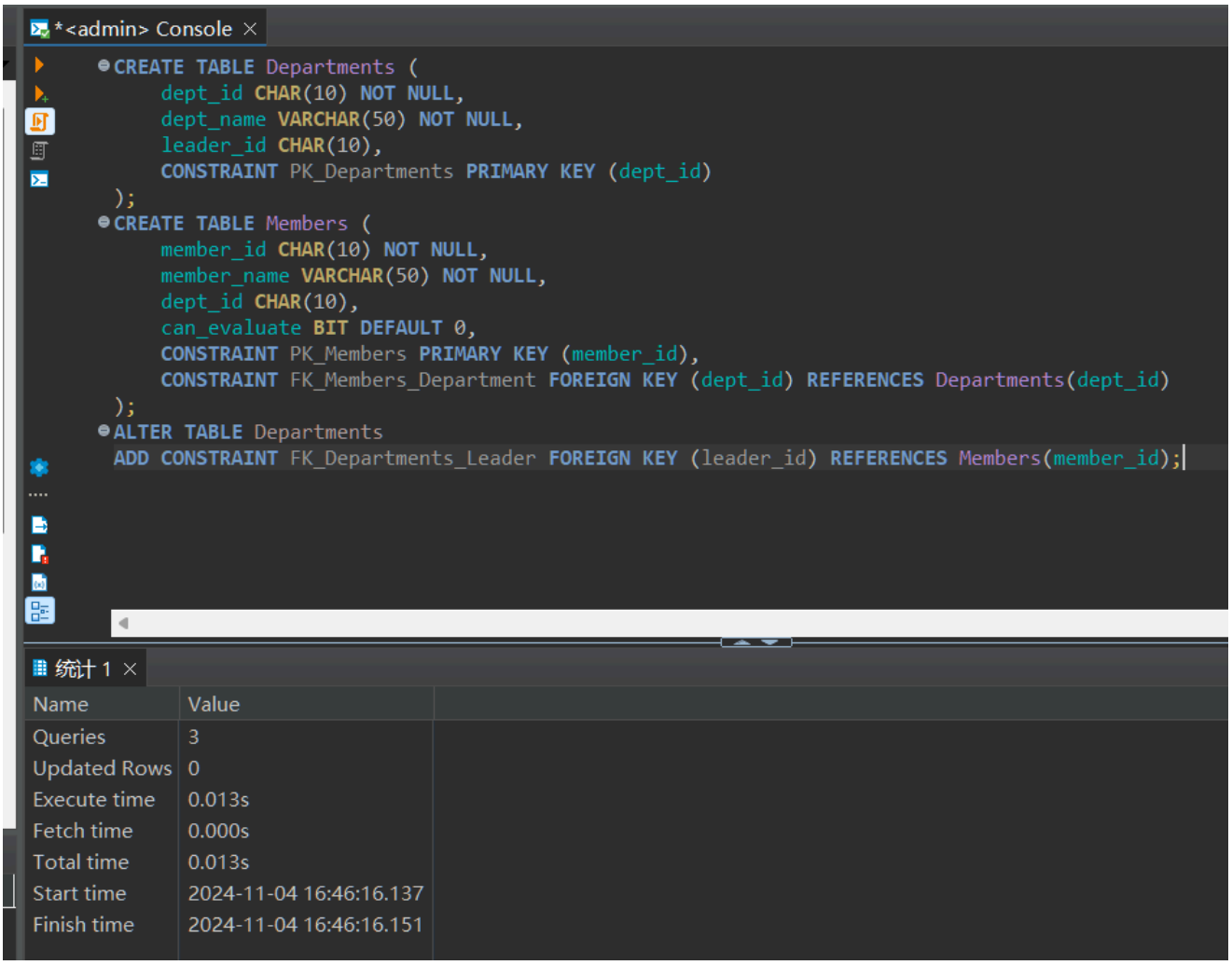
```
CREATE TABLE Members (
    member_id CHAR(10) NOT NULL,
    member_name VARCHAR(50) NOT NULL,
    dept_id CHAR(10),
    can_evaluate BIT DEFAULT 0,
    CONSTRAINT PK_Members PRIMARY KEY (member_id),
    CONSTRAINT FK_Members_Department FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);
```

部门表记录每个部门的编号、名称以及部长的编号。部长编号作为外键，参照成员表中的成员编号。

```
CREATE TABLE Departments (
    dept_id CHAR(10) NOT NULL,
    dept_name VARCHAR(50) NOT NULL,
    leader_id CHAR(10),
    CONSTRAINT PK_Departments PRIMARY KEY (dept_id),
    CONSTRAINT FK_Departments_Leader FOREIGN KEY (leader_id) REFERENCES Members(member_id)
);
```

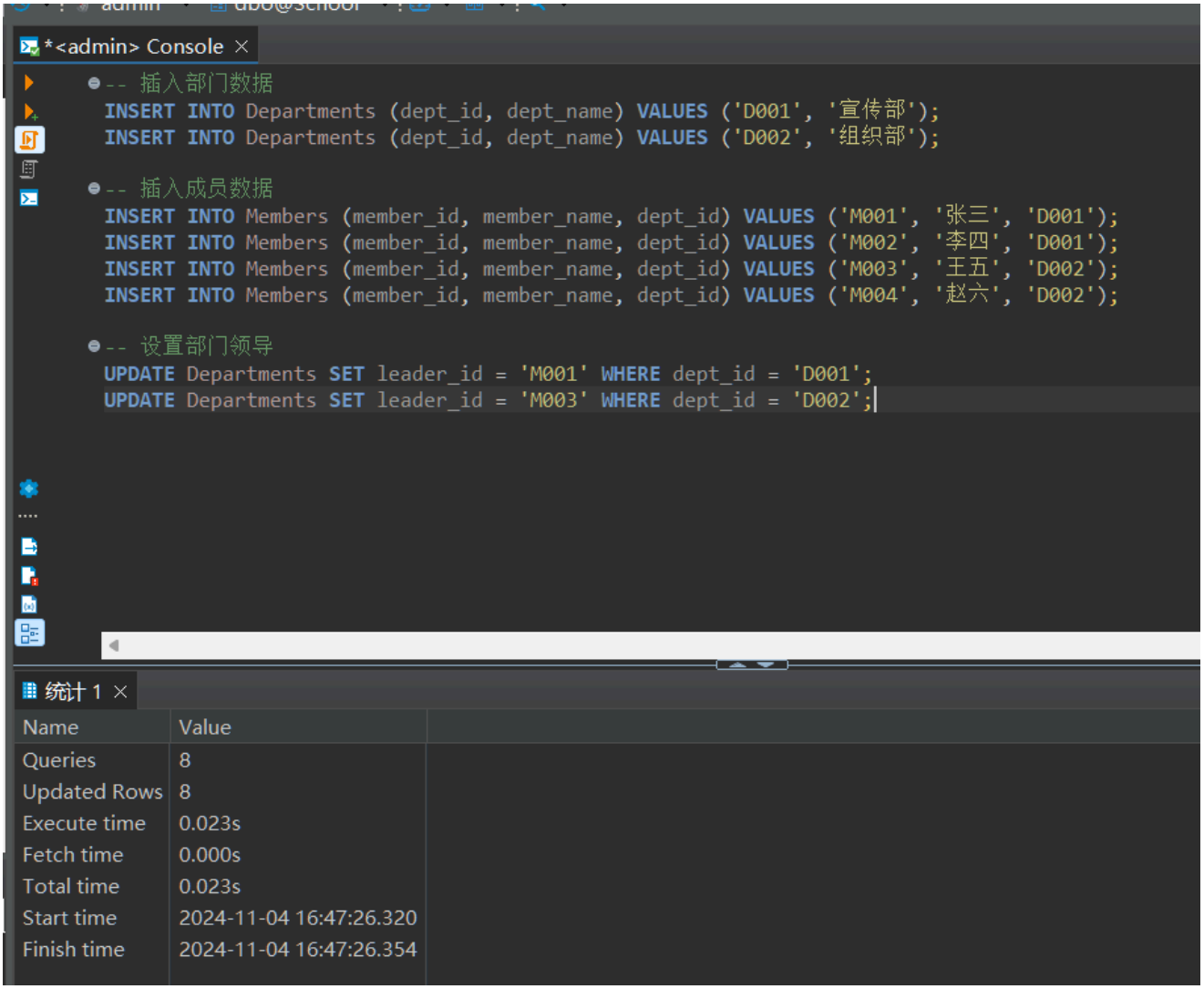
为了防止创建表的时候不存在另一个表，导致约束失败，采用下面的脚本先创建表，再ALTER。

```
CREATE TABLE Departments (
    dept_id CHAR(10) NOT NULL,
    dept_name VARCHAR(50) NOT NULL,
    leader_id CHAR(10),
    CONSTRAINT PK_Departments PRIMARY KEY (dept_id)
);
CREATE TABLE Members (
    member_id CHAR(10) NOT NULL,
    member_name VARCHAR(50) NOT NULL,
    dept_id CHAR(10),
    can_evaluate BIT DEFAULT 0,
    CONSTRAINT PK_Members PRIMARY KEY (member_id),
    CONSTRAINT FK_Members_Department FOREIGN KEY (dept_id) REFERENCES Departments(dept_id)
);
ALTER TABLE Departments
ADD CONSTRAINT FK_Departments_Leader FOREIGN KEY (leader_id) REFERENCES Members(member_id);
```



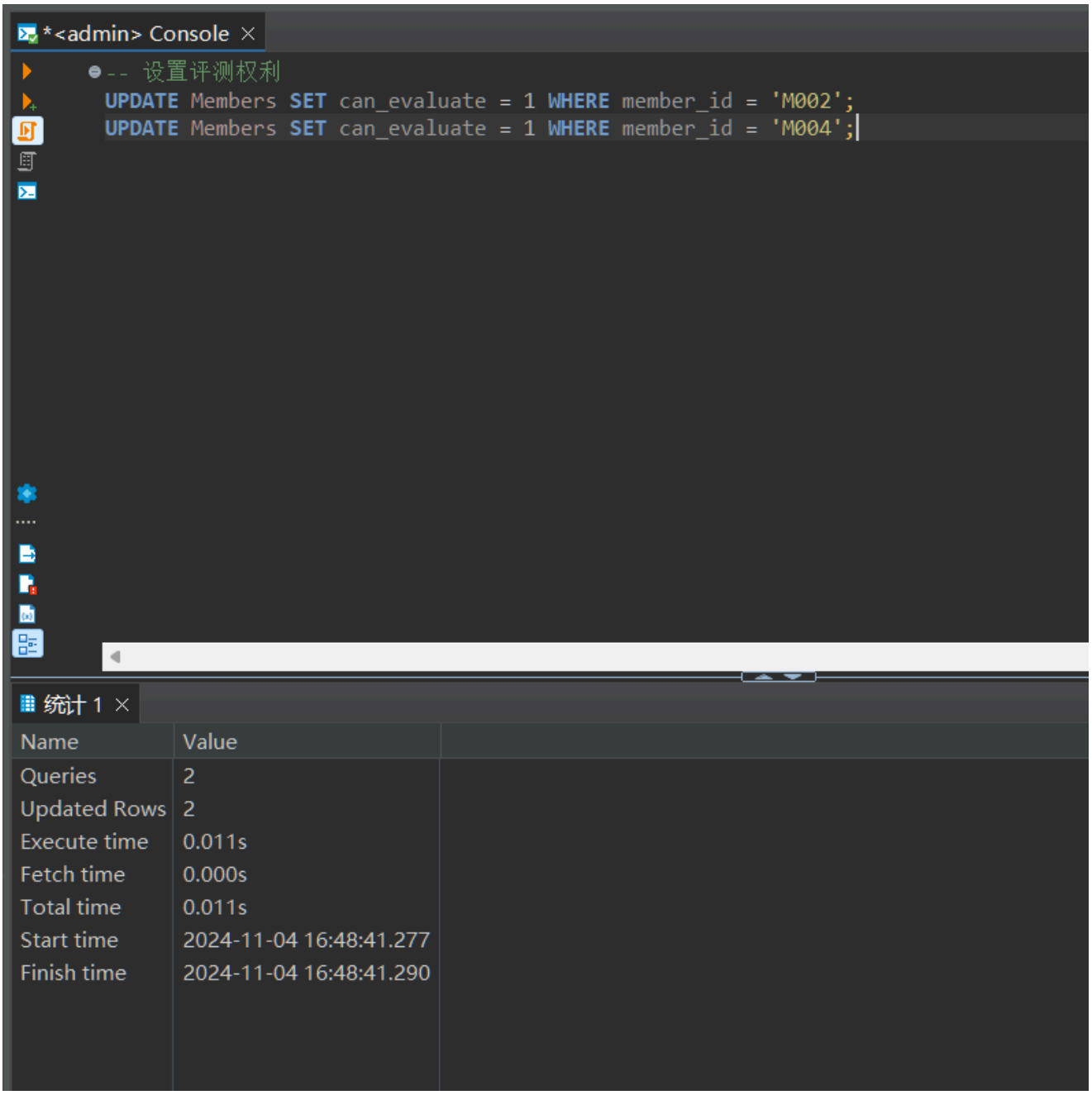
插入测试数据

```
-- 插入部门数据
INSERT INTO Departments (dept_id, dept_name) VALUES ('D001', '宣传部');
INSERT INTO Departments (dept_id, dept_name) VALUES ('D002', '组织部');

-- 插入成员数据
INSERT INTO Members (member_id, member_name, dept_id) VALUES ('M001', '张三', 'D001');
INSERT INTO Members (member_id, member_name, dept_id) VALUES ('M002', '李四', 'D001');
INSERT INTO Members (member_id, member_name, dept_id) VALUES ('M003', '王五', 'D002');
INSERT INTO Members (member_id, member_name, dept_id) VALUES ('M004', '赵六', 'D002');

-- 设置部门领导
UPDATE Departments SET leader_id = 'M001' WHERE dept_id = 'D001';
UPDATE Departments SET leader_id = 'M003' WHERE dept_id = 'D002';
```



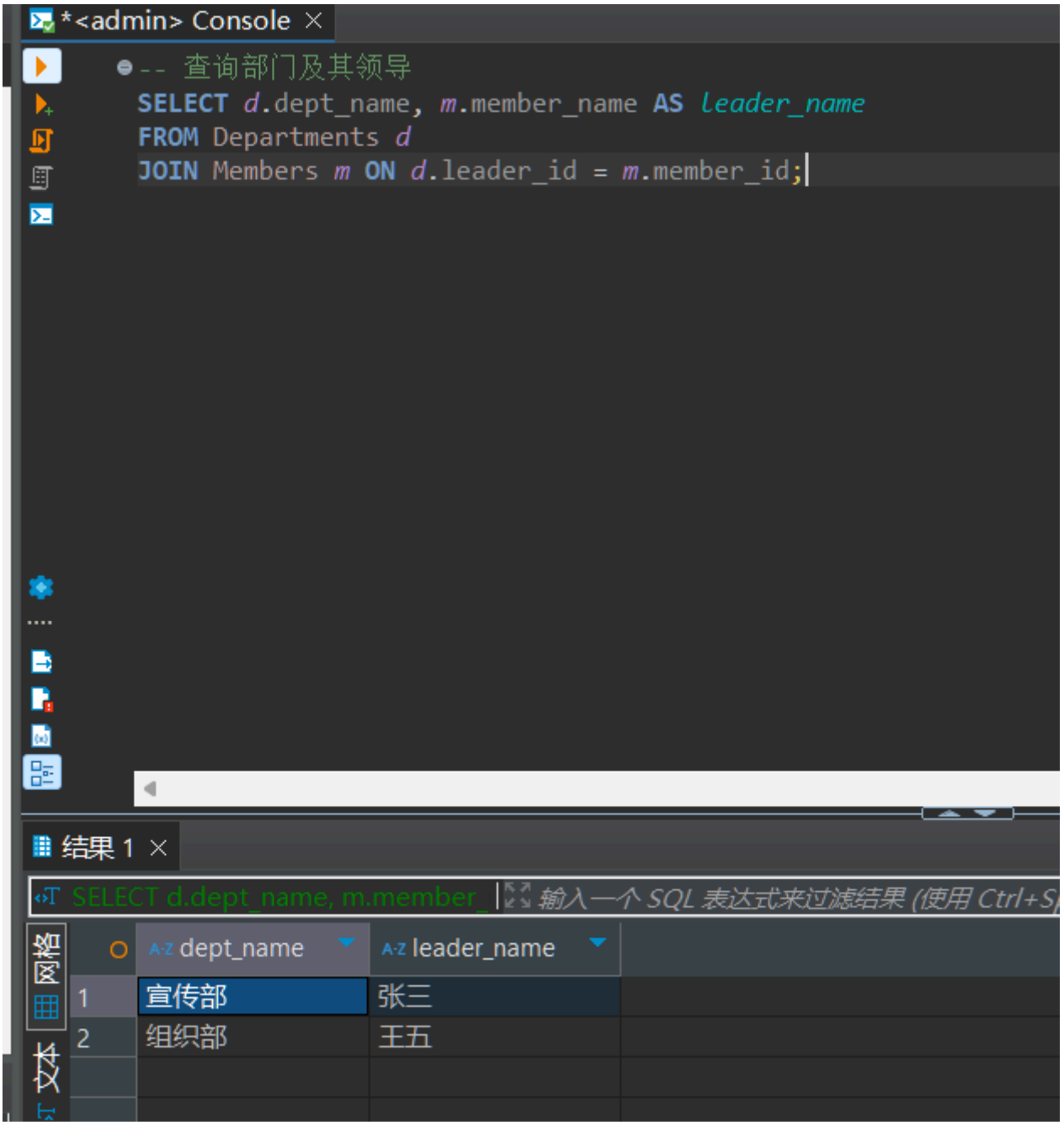
设置评测权利

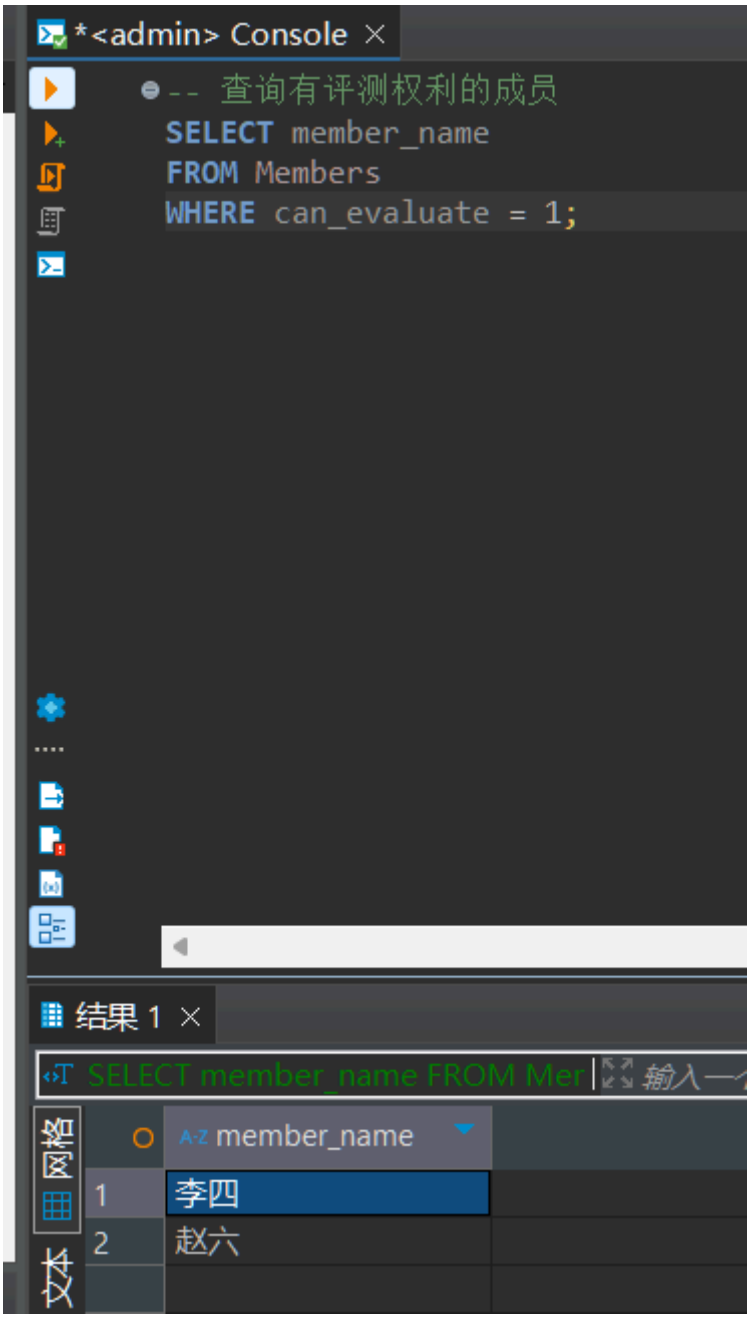
```
UPDATE Members SET can_evaluate = 1 WHERE member_id = 'M002';
UPDATE Members SET can_evaluate = 1 WHERE member_id = 'M004';
```



查询验证

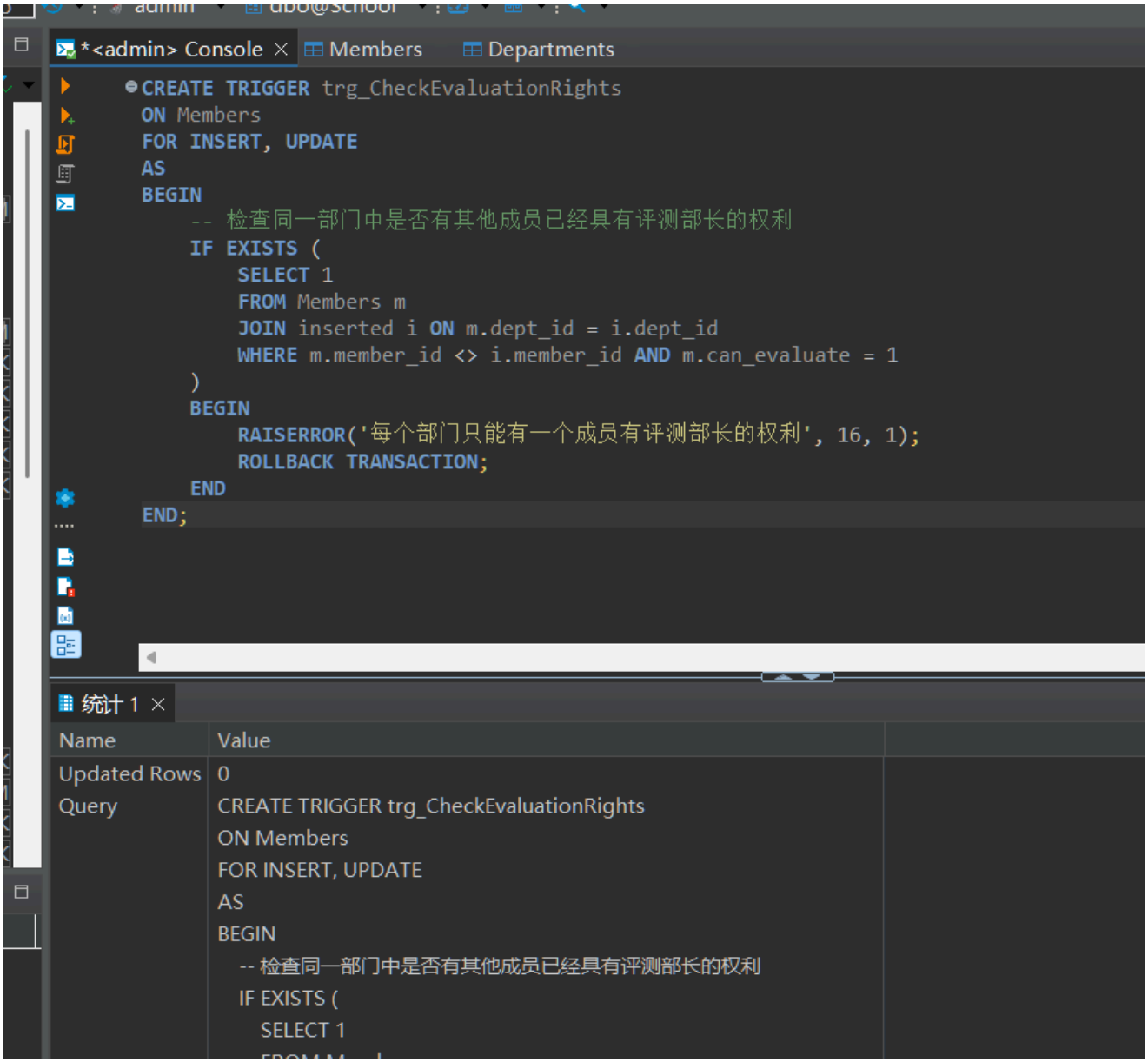
```sql
-- 查询部门及其领导
SELECT d.dept_name, m.member_name AS leader_name
FROM Departments d
JOIN Members m ON d.leader_id = m.member_id;

-- 查询有评测权利的成员
SELECT member_name
FROM Members
WHERE can_evaluate = 1;
```


*<admin> Console
-- 查询部门及其领导
SELECT d.dept_name, m.member_name AS leader_name
FROM Departments d
JOIN Members m ON d.leader_id = m.member_id;
结果 1
SELECT d.dept_name, m.member_
输入一个 SQL 表达式来过滤结果 (使用 Ctrl+S
dept_name
leader_name
1
宣传部
张三
2
组织部
王五



为了实现每个部只能有一个部员有评测部长的权利，需要使用触发器。

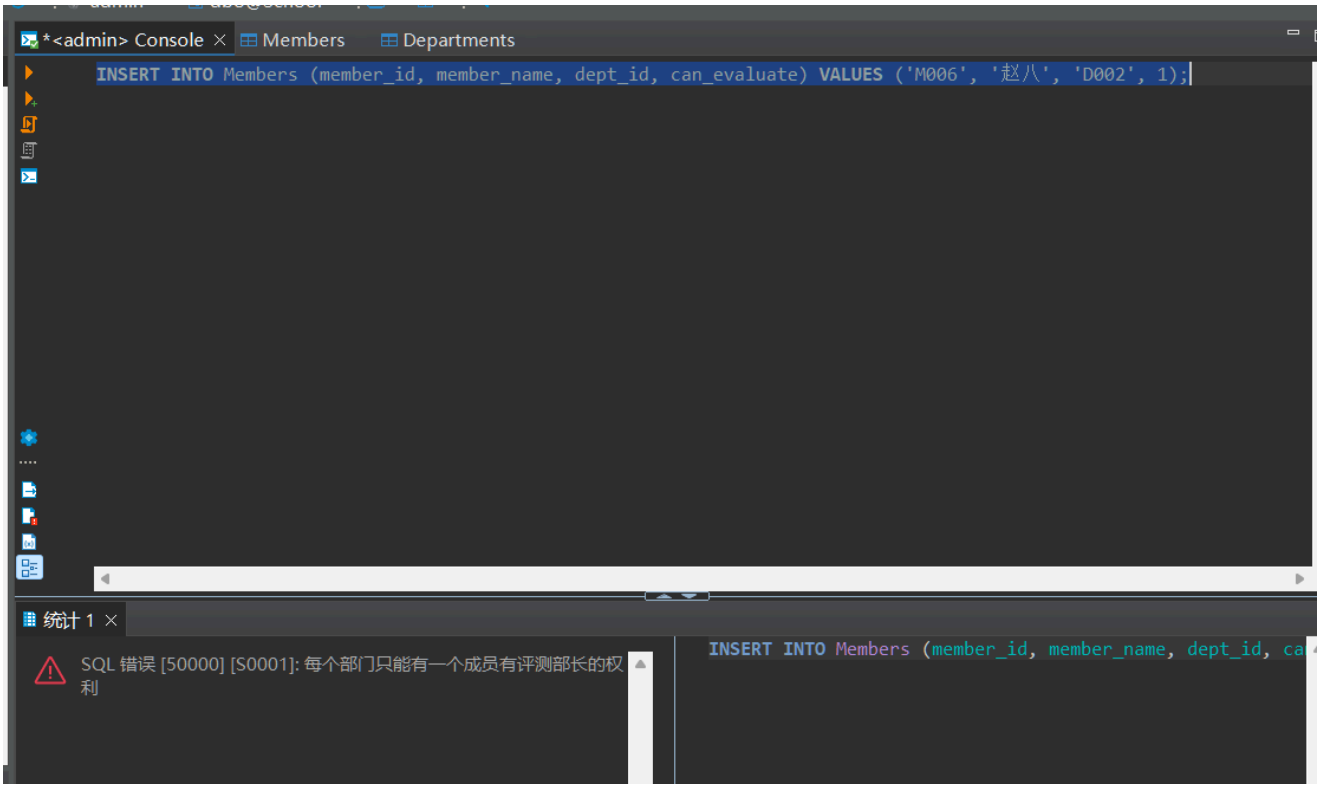
```
CREATE TRIGGER trg_CheckEvaluationRights
ON Members
FOR INSERT, UPDATE
AS
BEGIN
    -- 检查同一部门中是否有其他成员已经具有评测部长的权利
    IF EXISTS (
        SELECT 1
        FROM Members m
        JOIN inserted i ON m.dept_id = i.dept_id
        WHERE m.member_id <> i.member_id AND m.can_evaluate = 1
    )
    BEGIN
        RAISERROR('每个部门只能有一个成员有评测部长的权利', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
```



验证约束

- 每个部员只能有一个部员有评测部长的权利

```
INSERT INTO Members (member_id, member_name, dept_id, can_evaluate) VALUES ('M006', '赵八',
'D002', 1);
```


*<admin> Console
Members
Departments
INSERT INTO Members (member_id, member_name, dept_id, can_evaluate) VALUES ('M006', '赵八', 'D002', 1);
统计 1
SQL 错误 [50000] [S0001]: 每个部门只能有一个成员有评测部长的权利
INSERT INTO Members (member_id, member_name, dept_id, ca