杭州电子科技大学

数据库原理

实 验 报 告

学　　院　网络空间安全学院

专　　业

班　　级

学　　号

学生姓名

教师姓名

完成日期

成　　绩

实验三 T-SQL 语句操作（二）

一、 实验目的、

熟练掌握 T-SQL 的统计功能与视图、关系图、触发器和存储过程的使用

二、 实验内容

（1）SELECT 语句的数据统计功能
（2）视图的创建与应用
（3）数据库关系图的使用
（4）触发器的使用
（5）存储过程的使用

三、 实验环境

BIOS 信息：

```
SMBIOSBIOSVersion : 1.19.1
Manufacturer      : Dell Inc.
Name              : 1.19.1
SerialNumber      : C5NY4N3
Version           : DELL-2
```

操作系统版本信息：

```
BuildNumber             : 19045
BuildType               : Multiprocessor Free
OSType                  : 18
ServicePackMajorVersion : 0
ServicePackMinorVersion : 0
```

使用软件信息：

Docker+MSSQL Server+DataGrip

四、　主要操作步骤及实验结果记录

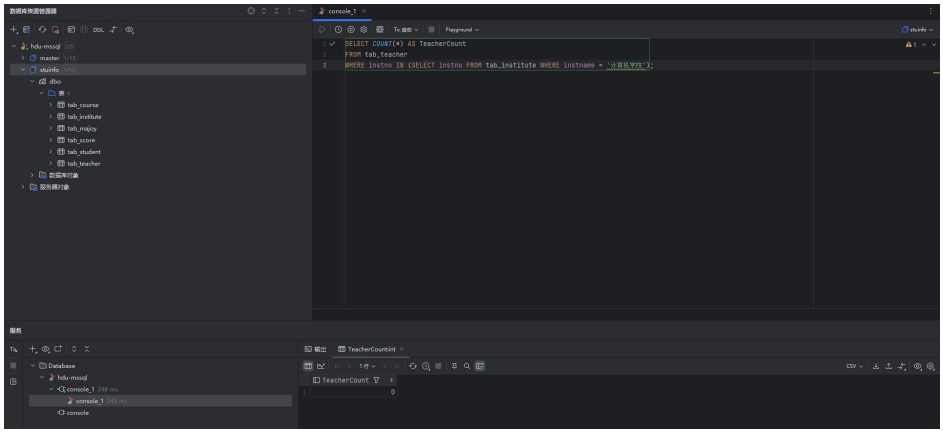
对应 T-SQL 语句过多，不方便一一截图，这里贴出所有对应代码，并给出 T-SQL 语句对应的及时，然后贴出其中具有代表性的运行截图。

（一）SELECT 语句的数据统计功能

1）统计计算机学院的教师人数

SELECT COUNT(*) AS TeacherCountFROM tab_teacherWHERE instno IN (SELECT instno FROM tab_institute WHERE instname = '计算机学院');

这条 SQL 语句首先通过子查询获取计算机学院的机构号，然后在教师表中统计该机构号下的教师人数。



2）统计网络工程专业每个学生的选课课程数

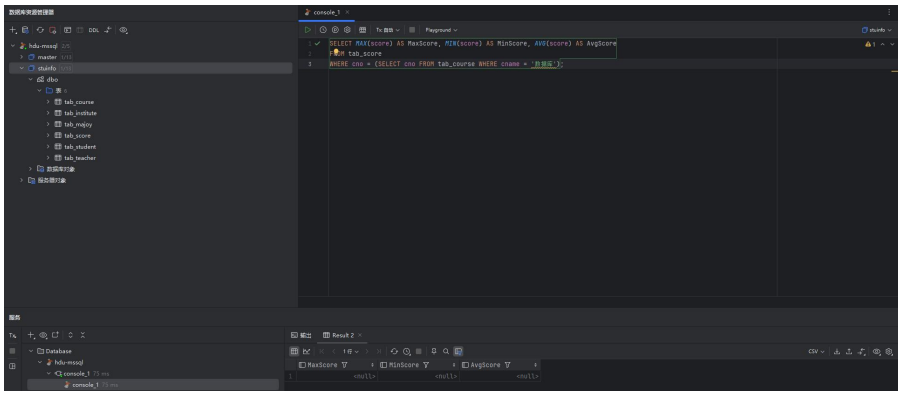
SELECT s.sno, COUNT(sc.cno) AS CourseCountFROM tab_student sLEFT JOIN tab_score sc ON s.sno = sc.snoLEFT JOIN tab_course c ON sc.cno = c.cnoWHERE s.majoyno = (SELECT majoyno FROM tab_majoy WHERE majoyname = '网络工程')GROUP BY s.sno;

这条 SQL 语句通过左连接学生表、成绩表和课程表，统计网络工程专业每个学生的选课课程数。

3）统计“数据库”课程的最高分、最低分和平均成绩

SELECT MAX(score) AS MaxScore, MIN(score) AS MinScore, AVG(score) AS AvgScoreFROM tab_scoreWHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库');

这条 SQL 语句通过子查询获取“数据库”课程的课程号，然后在成绩表中统计该课程的最高分、最低分和平均成绩。



4）查询“数据库”课程高于平均分的学生学号和成绩

SELECT sno, scoreFROM tab_scoreWHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库')AND score > (SELECT AVG(score) FROM tab_score WHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库'));

这条 SQL 语句首先通过子查询获取“数据库”课程的课程号，然后筛选出成绩高于该课程平均分的学生学号和成绩。

5）查询“数据库”课程成绩最高的学生学号和姓名（采用 MAX 函数和子查询）

SELECT s.sno, s.snameFROM tab_student sJOIN tab_score sc ON s.sno = sc.snoWHERE sc.score = (SELECT MAX(score) FROM tab_score WHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库'));

这条 SQL 语句通过子查询获取“数据库”课程的最高分，然后找出该课程成绩最高的学生学号和姓名。

6）查询平均分高于 75 分的课程号

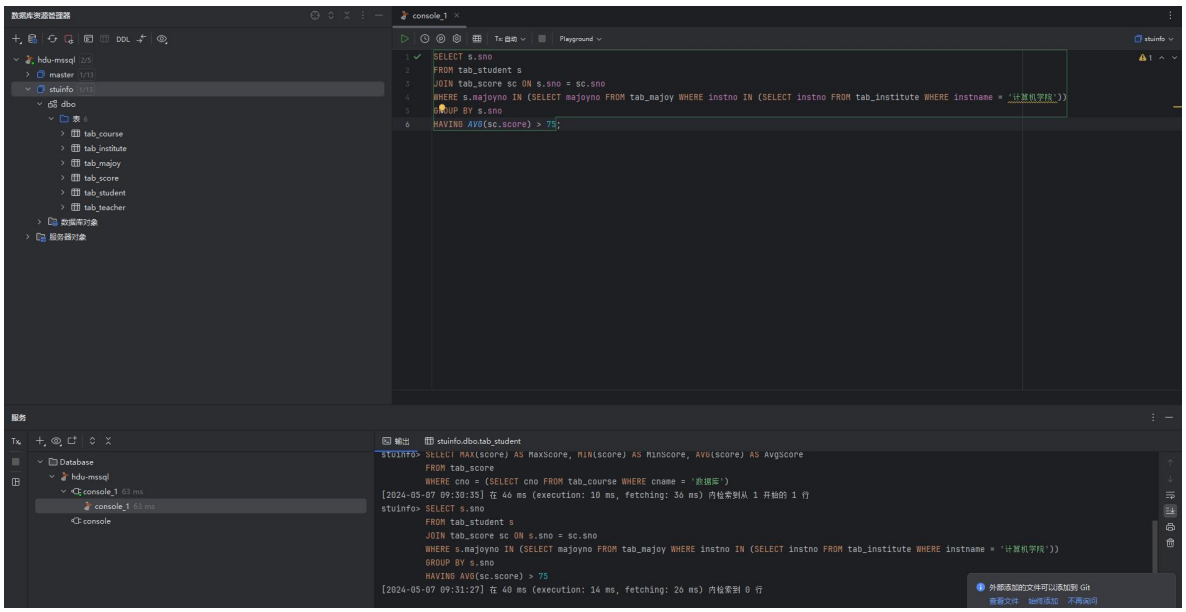
SELECT cnoFROM tab_scoreGROUP BY cnoHAVING AVG(score) > 75;

这条 SQL 语句按课程号分组，筛选出平均分高于 75 分的课程号。

7）查询计算机学院平均分高于 75 分的学生学号

SELECT s.snoFROM tab_student sJOIN tab_score sc ON s.sno = sc.snoWHERE s.majoyno IN (SELECT majoyno FROM tab_majoy WHERE instno IN (SELECT instno FROM tab_institute WHERE instname = '计算机学院'))GROUP BY s.snoHAVING AVG(sc.score) > 75;

这条 SQL 语句首先通过子查询获取计算机学院的机构号和专业号，然后找出该学院专业平均分高于 75 分的学生学号。



8）统计有不及格成绩的学生人数

SELECT COUNT(DISTINCT sno) AS StudentCountFROM tab_scoreWHERE score < 60;

这条 SQL 语句统计出有不及格成绩的学生人数，通过 DISTINCT 确保每个学生只计算一次。

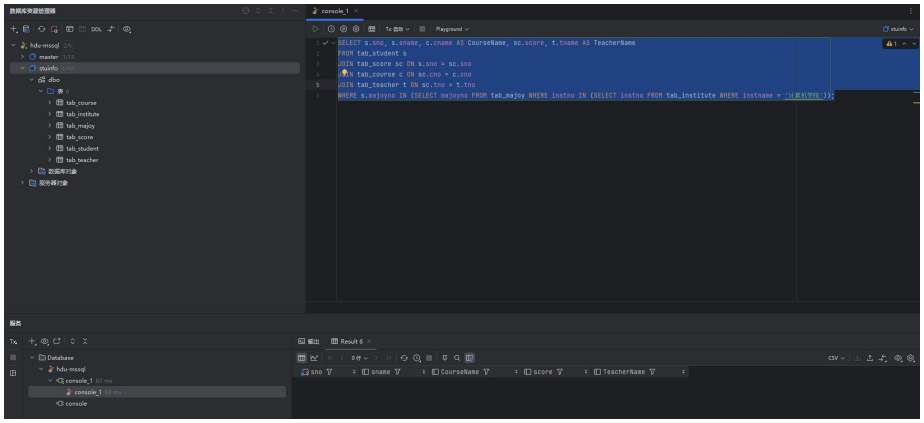
9）查询选课人数少于 10 人的课程号
SELECT cnoFROM tab_scoreGROUP BY cnoHAVING COUNT(DISTINCT sno) < 10;
这条 SQL 语句按课程号分组，筛选出选课人数少于 10 人的课程号。

（二）视图的创建与应用
1）查询全校的教师的工号、姓名、所在学院名
SELECT t.tno, t.tname, i.instname AS InstituteNameFROM tab_teacher tJOIN tab_institute i ON t.instno = i.instno;
这条 SQL 语句通过教师表和学院表的关联，查询全校教师的工号、姓名以及所在学院名。

2）查询授课教师的工号、姓名、讲授课程名
SELECT t.tno, t.tname, c.cname AS CourseNameFROM tab_teacher tJOIN tab_course c ON t.tno = c.tno;
这条 SQL 语句通过教师表和课程表的关联，查询授课教师的工号、姓名以及讲授课程名。

3）查询 “数据结构”课程的学生的学号、姓名、课程名、成绩、任课教师名
SELECT s.sno, s.sname, c.cname AS CourseName, sc.score, t.tname AS TeacherNameFROM tab_student sJOIN tab_score sc ON s.sno = sc.snoJOIN tab_course c ON sc.cno = c.cnoJOIN tab_teacher t ON sc.tno = t.tnoWHERE c.cname = '数据结构';
这条 SQL 语句通过学生表、成绩表、课程表和教师表的关联，查询“数据结构”课程的学生的学号、姓名、课程名、成绩以及任课教师名。

4）查询“计算机学院”所有学生的学号、姓名、课程名、成绩、任课教师名
SELECT s.sno, s.sname, c.cname AS CourseName, sc.score, t.tname AS TeacherName
FROM tab_student s
JOIN tab_score sc ON s.sno = sc.sno
JOIN tab_course c ON sc.cno = c.cno
JOIN tab_teacher t ON sc.tno = t.tno
WHERE s.majoyno IN (SELECT majoyno FROM tab_majoy WHERE instno IN (SELECT instno FROM tab_institute WHERE instname = '计算机学院'));
这条 SQL 语句通过子查询获取计算机学院的机构号和专业号，然后查询该学院所有学生的学号、姓名、课程名、成绩以及任课教师名。



5）查询所有授课的“教授”的工号、姓名和授课课程名

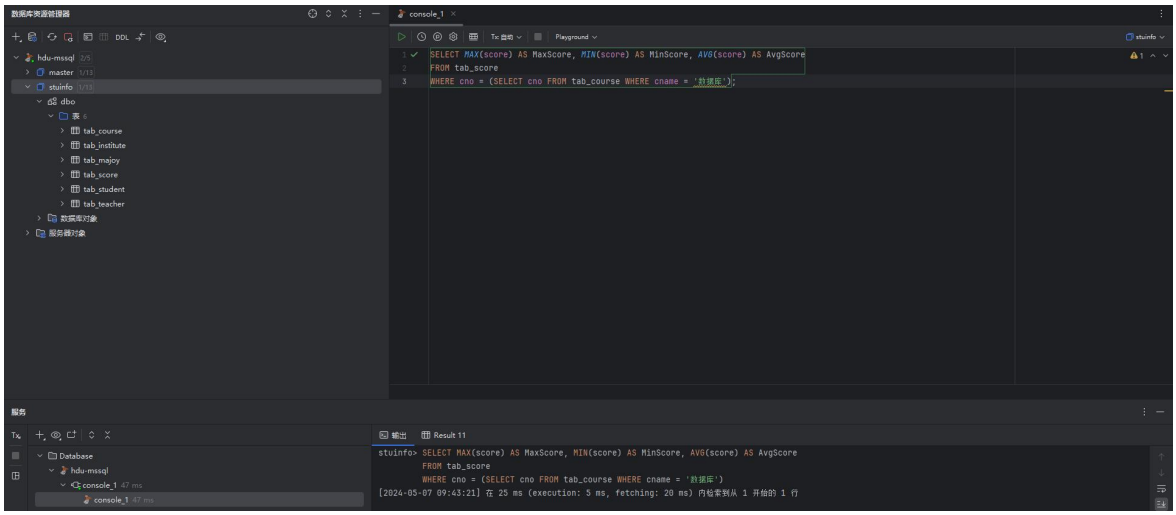
SELECT t.tno, t.tname, c.cname AS CourseNameFROM tab_teacher tJOIN tab_course c ON t.tno = c.tnoWHERE t.title = '教授';

这条 SQL 语句查询所有授课的“教授”的工号、姓名以及授课课程名。

6）统计“数据库”课程的最高分、最低分和平均成绩

SELECT MAX(score) AS MaxScore, MIN(score) AS MinScore, AVG(score) AS AvgScoreFROM tab_scoreWHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库');

这条 SQL 语句通过子查询获取“数据库”课程的课程号，然后统计该课程的最高分、最低分和平均成绩。



7）查询“数据库”课程高于平均分的学生学号和成绩

SELECT sno, scoreFROM tab_scoreWHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库')AND score > (SELECT AVG(score) FROM tab_score WHERE cno = (SELECT cno FROM tab_course WHERE cname = '数据库'));

这条 SQL 语句首先通过子查询获取“数据库”课程的课程号，然后筛选出成绩高于该课程平均分的学生学号和成绩。

8）查询平均分最高的课程号、课程名、任课教师、平均成绩

```
SELECT TOP 1 cno, CourseName, TeacherName, AvgScore
FROM (
    SELECT c.cno, c.cname AS CourseName, t.tname AS TeacherName, AVG(sc.score) AS AvgScore
    FROM tab_course c
    JOIN tab_score sc ON c.cno = sc.cno
    JOIN tab_teacher t ON c.tno = t.tno
    GROUP BY c.cno, c.cname, t.tname
) AS SubQuery
ORDER BY AvgScore DESC;
```

这条 SQL 语句查询平均分最高的课程号、课程名、任课教师以及平均成绩，并按平均成绩降序排序，只返回最高平均分的课程。

9）查询平均分高于 75 分的学号、姓名、平均成绩、专业名称、学院名称

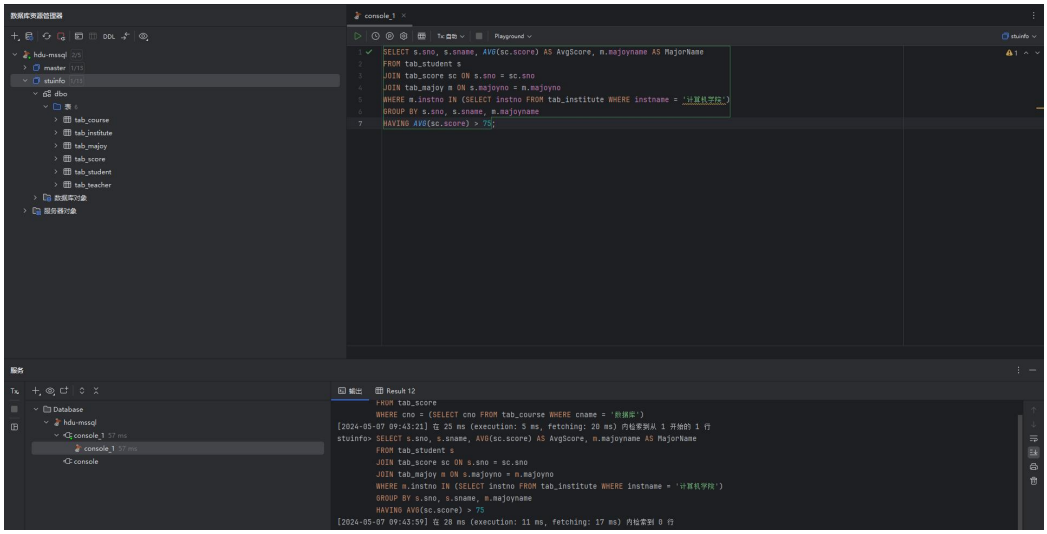
SELECT s.sno, s.sname, AVG(sc.score) AS AvgScore, m.majoyname AS MajorName, i.instname AS InstituteNameFROM tab_student sJOIN tab_score sc ON s.sno = sc.snoJOIN tab_majoy m ON s.majoyno = m.majoynoJOIN tab_institute i ON m.instno = i.instnoGROUP BY s.sno, s.sname, m.majoyname, i.instnameHAVING AVG(sc.score) > 75;

这条 SQL 语句通过学生表、成绩表、专业表和学院表的关联，查询平均分高于 75 分的学生学号、姓名、平均成绩、专业名称以及学院名称。

10）查询计算机学院平均分高于 75 分的学生学号、姓名、平均成绩、专业名称

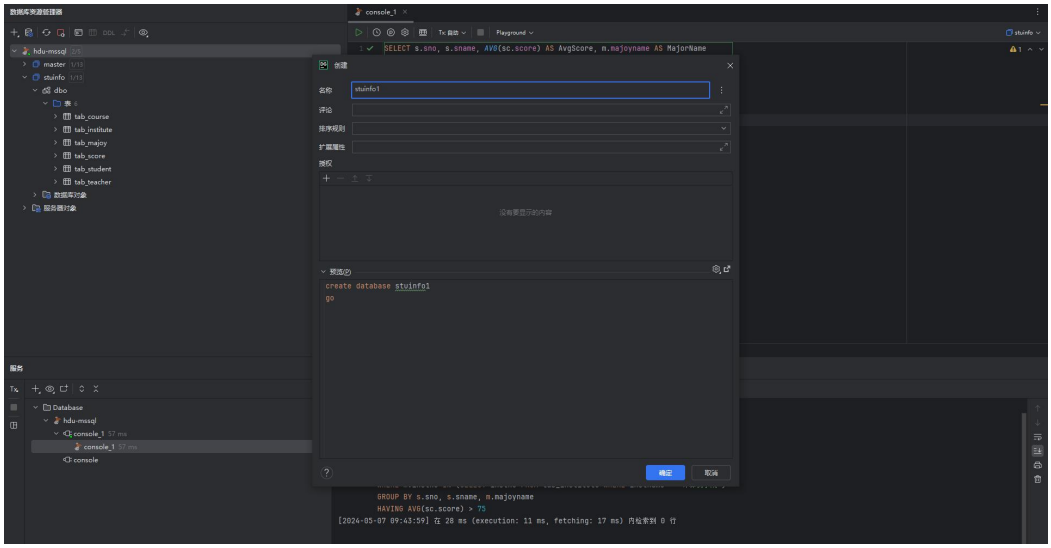
SELECT s.sno, s.sname, AVG(sc.score) AS AvgScore, m.majoyname AS MajorNameFROM tab_student sJOIN tab_score sc ON s.sno = sc.snoJOIN tab_majoy m ON s.majoyno = m.majoynoWHERE m.instno IN (SELECT instno FROM tab_institute WHERE instname = '计算机学院')GROUP BY s.sno, s.sname, m.majoynameHAVING AVG(sc.score) > 75;

这条 SQL 语句首先通过子查询



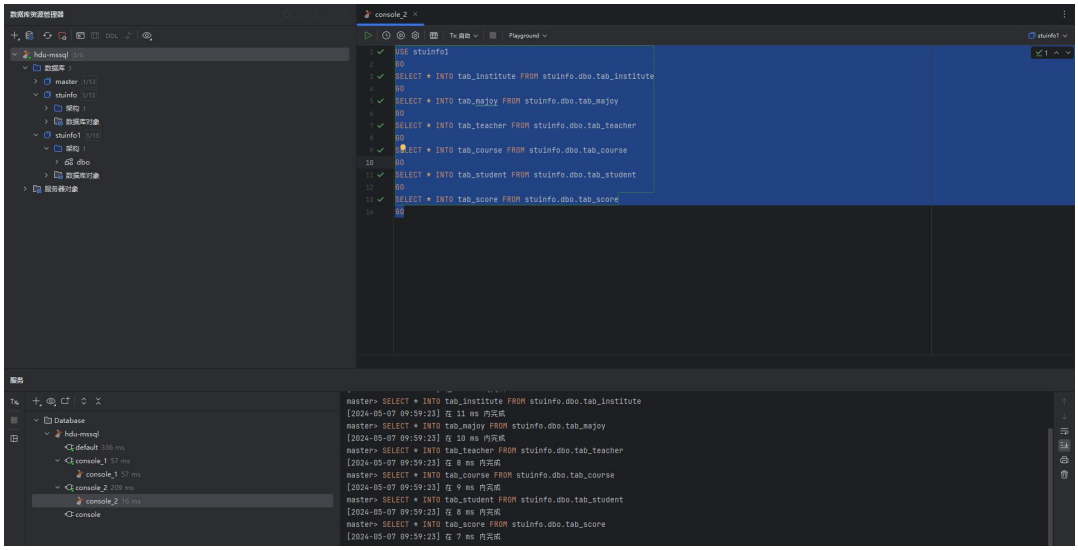
（三）数据库关系图的关系

（1）创建数据库“stuinfo1”

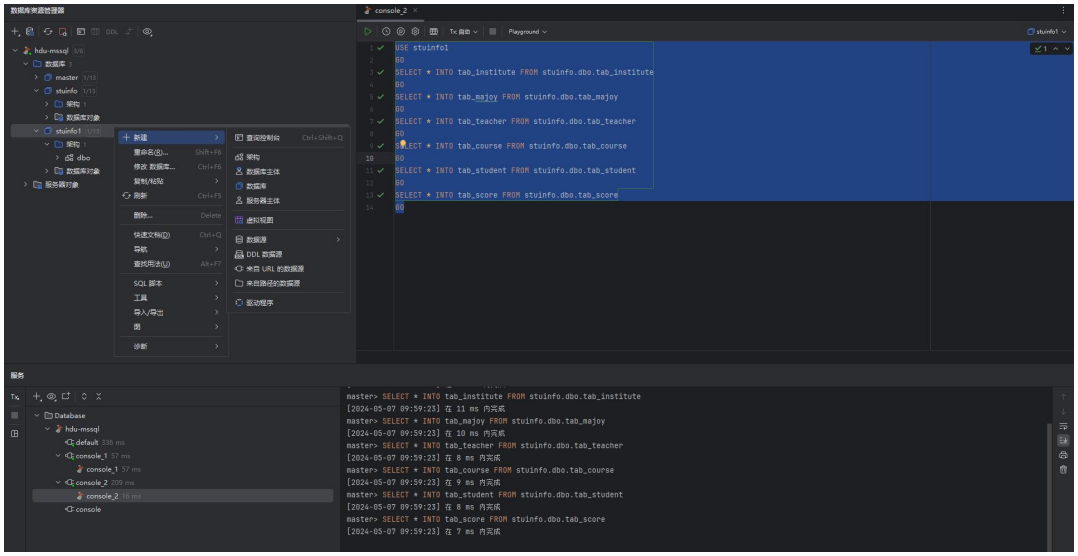


利用 SELECT 的 INTO 子句将“stuinfo”数据库中的基本复制到“stuinfo1”数据库。T-SQL 语句如下：

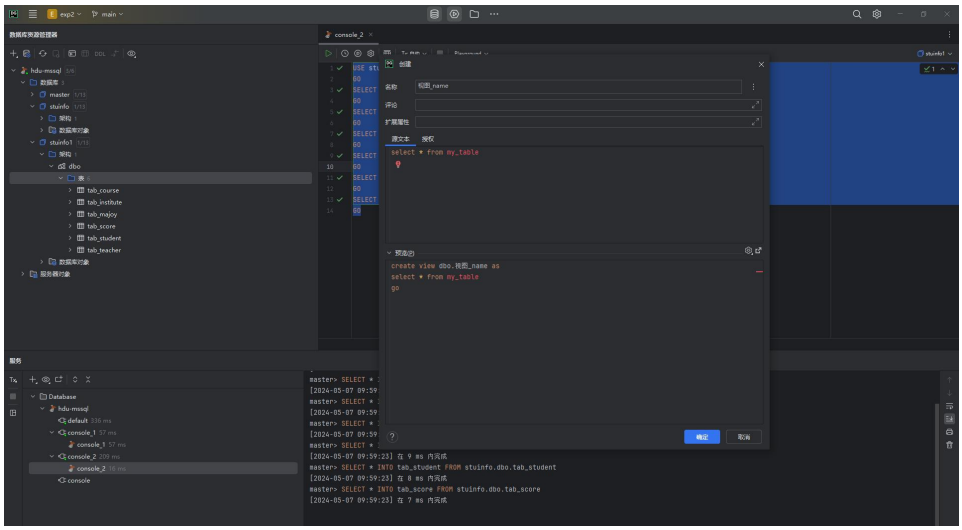
```
USE stuinfo1
GO
SELECT * INTO tab_institute FROM stuinfo.dbo.tab_institute
GO
SELECT * INTO tab_majoy FROM stuinfo.dbo.tab_majoy
GO
SELECT * INTO tab_teacher FROM stuinfo.dbo.tab_teacher
GO
SELECT * INTO tab_course FROM stuinfo.dbo.tab_course
GO
SELECT * INTO tab_student FROM stuinfo.dbo.tab_student
GO
SELECT * INTO tab_score FROM stuinfo.dbo.tab_score
GO
```



（2）打开 “stuinfo1”数据库文件夹，选中“数据库关系图”，点击右键在快捷菜单中点击“新建数据库关系图”。

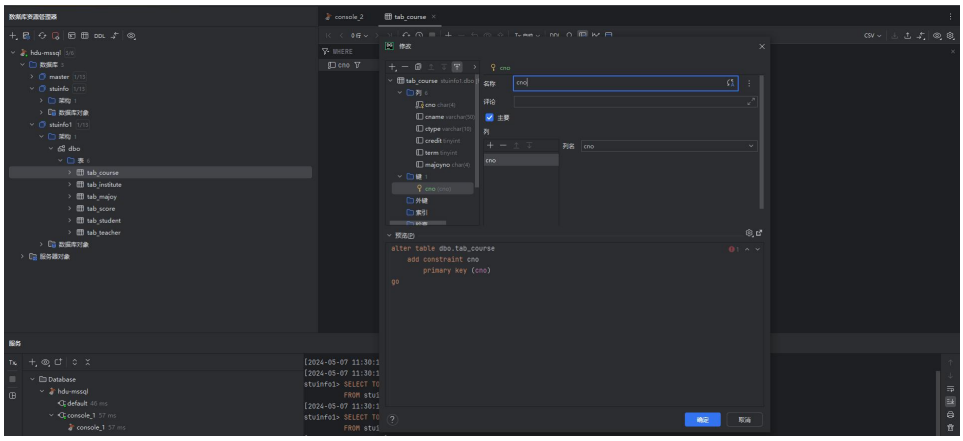


（3）在“添加表”对话框中将基本表添加到关系图编辑框中



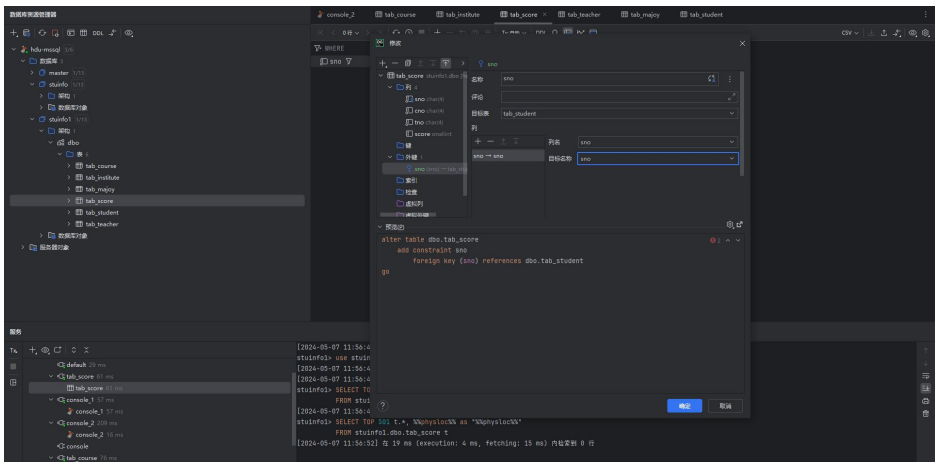
（4）定义各个基本表的主键

按照实验三 PPT 的要求定义各个基本表的逐渐，表比较多，不方便一一截图，这里截取 tab_course 表的主键 cno 的创建过程做演示：



（5）定义各基本表之间的外部键约束，同时关系图也创建了。

按照实验三 PPT 的要求定义各基本表之间的外键约束，同时创建关系图。具体绑定外键关系根据实验一而定。因为绑定的关系较多，此处不一一截图，以下贴图为将 tab_score 表的 sno 列与 tab_student 表的主键 sno 列绑定的视图：



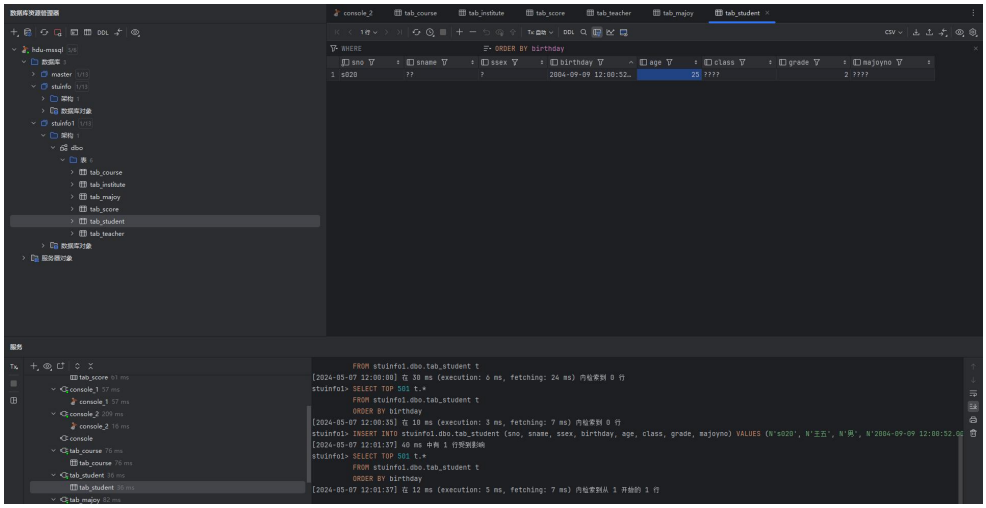
（6）关闭关系图编辑器，保存关系图。

（四）触发器的使用

1. 操作准备

（1）在 tab_student 表中添加一个学生记录，如:

s020，王五，男，1999-8-9, …”。



（2）在 tab_score 表中添加该学生的选课记录和成绩。

2. 操作场景：学生“王五”退学处理。

3. 具体操作要求如下：

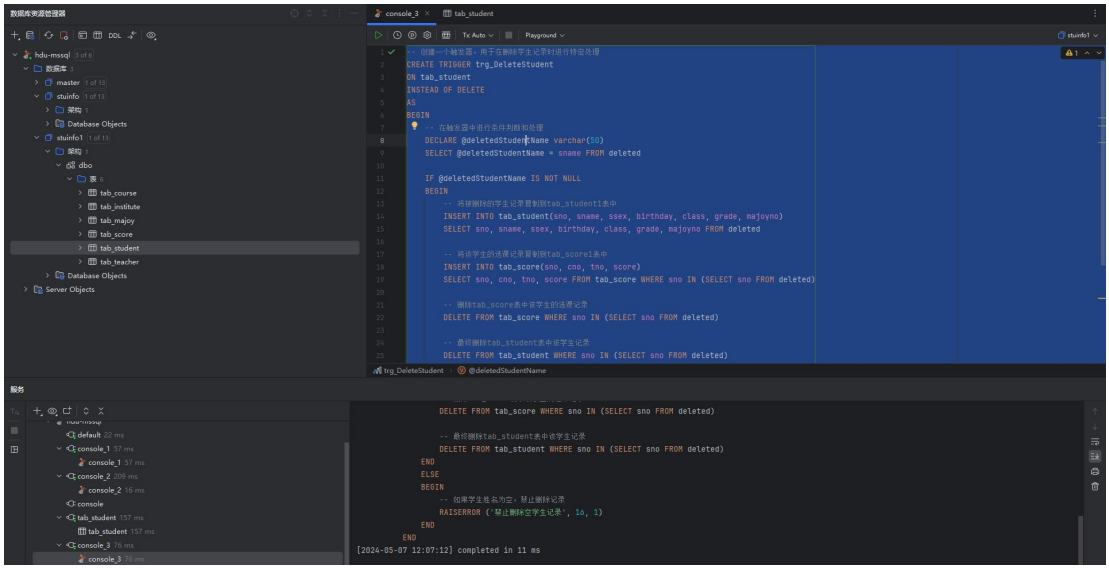
设计一个删除 tab_student 表中记录的触发器，如果删除的记录是“王五”，则将该学生记录复制 tab_student1 表中，同时在 tab_score 表中删除该学生选课记录，并将该学生的记录复制 tab_score1 表中；否则禁止删除记录。

该节要求对应的 T-SQL 语句如下：

```
-- 创建一个触发器，用于在删除学生记录时进行特定处理
CREATE TRIGGER trg_DeleteStudent
ON tab_student
INSTEAD OF DELETE
AS
BEGIN
    -- 在触发器中进行条件判断和处理
    DECLARE @deletedStudentName varchar(50)
    SELECT @deletedStudentName = sname FROM deleted

    IF @deletedStudentName IS NOT NULL
    BEGIN
        -- 将被删除的学生记录复制到 tab_student1 表中
        INSERT INTO tab_student(sno, sname, ssex, birthday, class, grade, majoyno)
        SELECT sno, sname, ssex, birthday, class, grade, majoyno FROM deleted

        -- 将该学生的选课记录复制到 tab_score1 表中
        INSERT INTO tab_score(sno, cno, tno, score)
        SELECT sno, cno, tno, score FROM tab_score WHERE sno IN (SELECT sno FROM deleted)
```

```
        -- 删除 tab_score 表中该学生的选课记录
        DELETE FROM tab_score WHERE sno IN (SELECT sno FROM deleted)

        -- 最终删除 tab_student 表中该学生记录
        DELETE FROM tab_student WHERE sno IN (SELECT sno FROM deleted)
    END
    ELSE
    BEGIN
        -- 如果学生姓名为空，禁止删除记录
        RAISERROR ('禁止删除空学生记录', 16, 1)
    END
END
```

对应操作的 DataGrip 视图截图如下：



4. 创建触发器

学生实验时创建触发器的 T-SQL 语句略有不同，已附在上面，可以查看。

```
CREATE TRIGGER trig_delete    ON tab_student
        FOR DELETE
        AS IF '王五' NOT IN (SELECT sname    FROM deleted)
                    BEGIN
            PRINT ' This record not is  王五, Can not delete! '
            ROLLBACK TRANSACTION
    END
                ELSE
    BEGIN
        INSERT INTO tab_student1 SELECT * FROM deleted
        INSERT INTO tab_score1 SELECT tab_score.* FROM tab_score,deleted
                        WHERE tab_score.sno=deleted.sno
        DELETE FROM tab_score WHERE sno IN (select sno from deleted)
    END
```

5. 运行触发器

在 SSMS 的查询编辑器执行以下命令

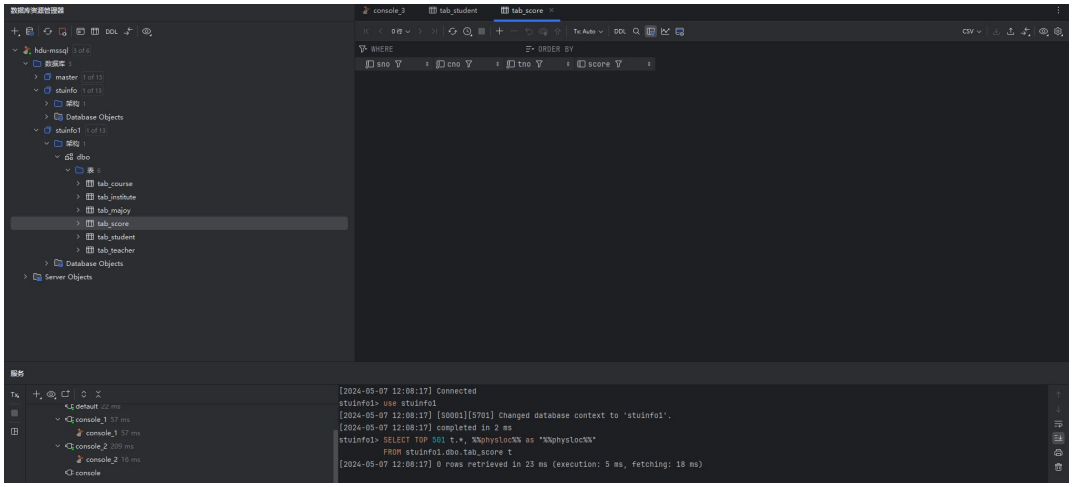
（1）DELETE FROM tab_student WHERE sname <> '王五'

然后打开 tab_student、tab_student1、tab_score 和 tab_score1 观察表记录。

（2）DELETE FROM tab_student WHERE sname= '王五'

然后打开 tab_student、tab_student1、tab_score 和 tab_score1 观察表记录。

可以观察到姓名为“王五”的学生信息已经被触发器触发操作同步删除。



（五）存储过程

（1）存储过程创建

```
USE stuinfo
/**************创建视图****************/
IF EXISTS (SELECT * FROM sysobjects WHERE name = 'View_score' AND type = 'V')
      DROP VIEW View_score
GO
CREATE VIEW View_score
AS
            SELECT a.sno, sname, a.cno, cname, a.tno, tname,term,score
      FROM tab_score a INNER JOIN tab_student b ON a.sno = b.sno
                  INNER JOIN tab_teacher c ON a.tno = c.tno
                  INNER JOIN tab_course d ON a.cno = d.cno
GO
/**************创建存储过程****************/
IF EXISTS (SELECT * FROM sysobjects WHERE name = 'stu_max_score' AND type = 'P')
      DROP PROCEDURE stu_max_score
GO
CREATE PROCEDURE stu_max_score
@term smallint
AS
       SELECT sname AS 姓名,max(score)AS 最高分,min(score)AS 最低分,AVG(score) AS 平均分
      FROM View_score  WHERE term=@term GROUP BY sname
GO
```

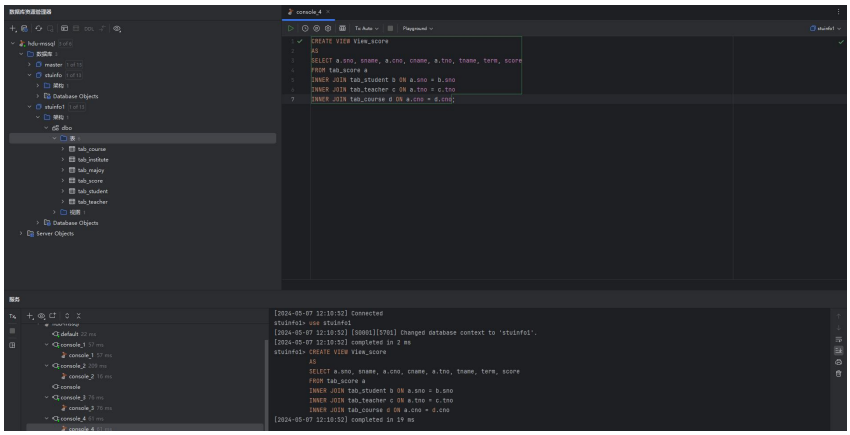
（2）存储过程执行

EXECUTE stu_max_score　1

学生实验中，使用以下 SQL 语句创建视图 View_score，该视图用于联结学生、教师和课程表以便获取学生的成绩信息：

```
CREATE VIEW View_score
AS
SELECT a.sno, sname, a.cno, cname, a.tno, tname, term, score
FROM tab_score a
INNER JOIN tab_student b ON a.sno = b.sno
INNER JOIN tab_teacher c ON a.tno = c.tno
INNER JOIN tab_course d ON a.cno = d.cno;
```

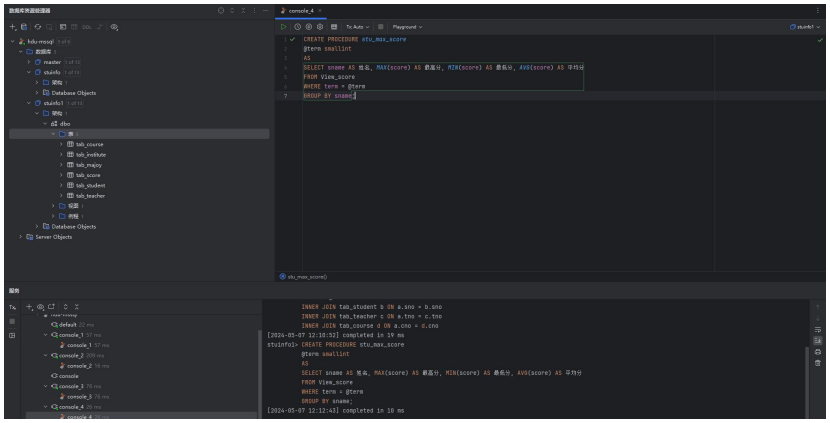
操作成功的截图如下：



接着，使用以下 SQL 语句创建存储过程 stu_max_score，该存储过程接受一个参数@term，用于指定学期，然后查询该学期所有学生的最高分、最低分和平均成绩：

```
CREATE PROCEDURE stu_max_score
@term smallint
AS
SELECT sname AS 姓名, MAX(score) AS 最高分, MIN(score) AS 最低分, AVG(score) AS 平均分
FROM View_score
WHERE term = @term
GROUP BY sname;
```

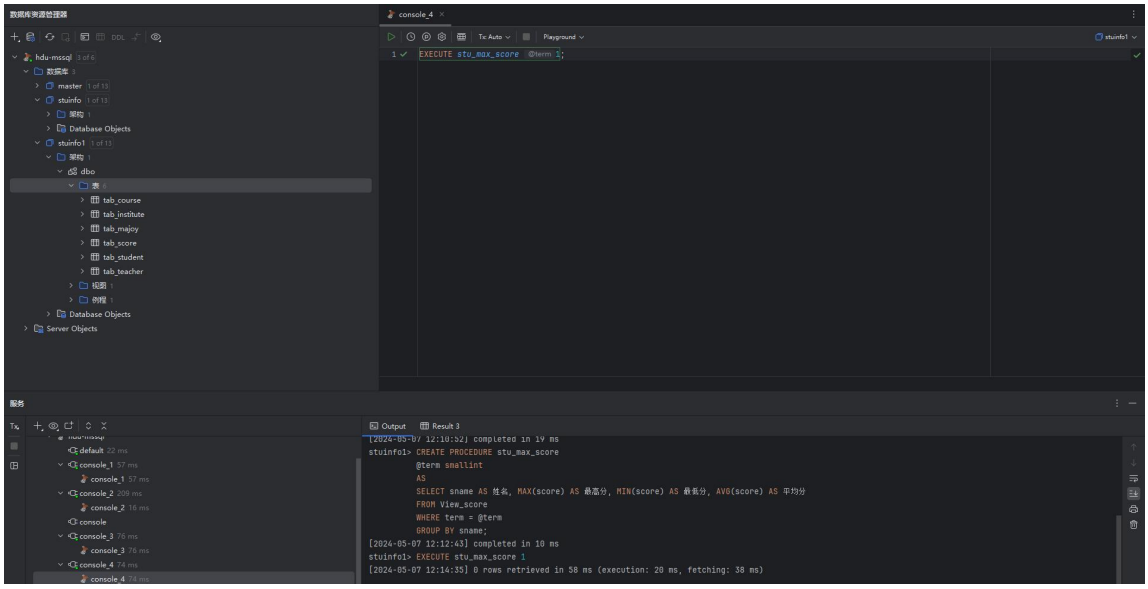
对应操作成功的截图如下：



最后，为了验证前面的操作是否成功，尝试执行存储过程 stu_max_score，传入学期参数进行查询。比如使用以下 SQL 语句执行存储过程并传入学期参数：

EXECUTE stu_max_score 1;

对应 T-SQL 语句操作成功截图如下：



至此，实验完成。

五、 实验分析总结及心得

在本次实验中，通过对 T-SQL 的统计功能、视图、关系图、触发器和存储过程的使用，加深了对数据库操作的理解和掌握。通过 SELECT 语句的数据统计功能，我们可以灵活地进行数据分析和统计，从而获取所需的信息。创建视图可以方便地组织和展示数据，提高数据查询的效率。数据库关系图的使用帮助我们更直观地了解表之间的关系，便于设计和优化数据库结构。触发器的使用可以实现对数据的自动化处理，提高数据的完整性和一致性。存储过程的应用则可以将复杂的操作封装起来，提高数据库操作的效率和可维护性。

在实验过程中，遇到了触发器设计的问题，需要根据特定条件对数据进行处理。通过分析需求和触发器的工作原理，成功设计了能够满足要求的触发器，实现了学生“王五”退学处理的功能。此外，通过创建视图和存储过程，可以更方便地进行数据查询和统计，提高了操作的效率和准确性。

通过本次实验，我深刻理解了 T-SQL 的各种功能和应用场景，掌握了数据库操作的基本技能。在实践中遇到问题时，通过分析原理和查阅资料，能够找到解决方案并成功实现目标。通过实验学到的知识和技术将对今后的数据库开发和管理工作产生积极影响，帮助我更好地应对复杂的数据处理需求。

此外，感觉本次实验的实验报告中需要附上大段的 T-SQL 语句，感觉用.docx 确实不是很方便，但也在某种程度上强化了我使用办公软件的能力，相信这也可以在我未来的工作中提供帮助。