**《数据库系统原理》实验报告**

### **实验题目：**实验13 存储过程实验

**姓名：**

**实验日期：**

**实验内容及完成情况：**（可续页）

### 实验13 存储过程实验

**实验学时：2学时**

**实验类型：设计**

**实验要求： 必做**

**一、实验目的**

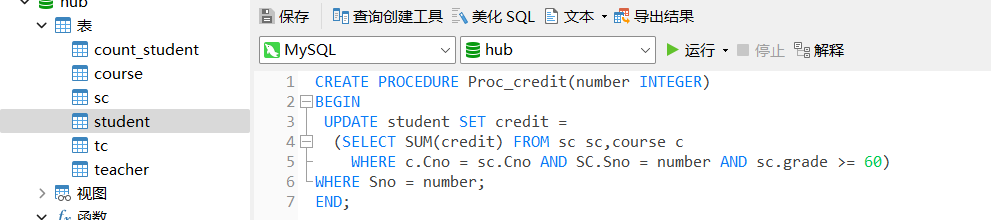
本实验的目的是使学生熟练掌握数据库PL/SQL编程语言，以及数据库存储过程的设计和使用方法；掌握数据库PL/SQL编程语言以及数据库自定义函数的设计和使用方法；掌握PL/SQL游标的设计、定义和使用方法，理解PL/SQL游标按行操作和SQL按结果集操作的区别和联系。

**二、实验内容**

1.存储过程实验：存储过程定义，存储过程运行，存储过程更名，存储过程删除，存储过程的参数传递。掌握PL/SQL编程语言和编程规范，规范设计存储过程。

（1）有参数的存储过程

①定义一个存储过程，更新指定学号学生的总学分。



CREATE PROCEDURE Proc\_credit(number INTEGER)

BEGIN

UPDATE student SET credit =

(SELECT SUM(credit)

FROM sc sc, course c

WHERE c.Cno = sc.Cno

AND sc.Sno = number

AND sc.grade >= 60)

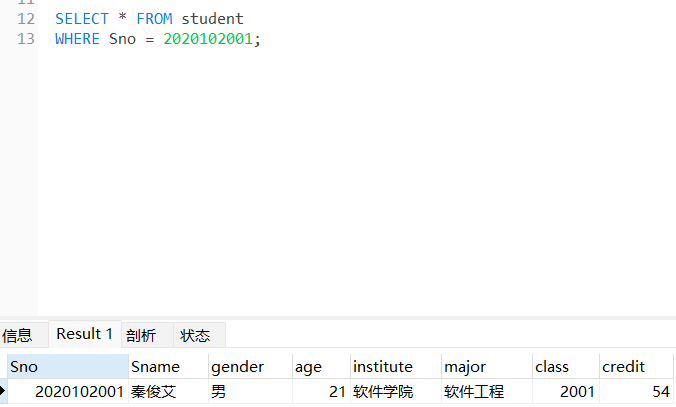
WHERE Sno = number;

END;

②将student表中学号为2020102001的学生的总学分credit改为0，然后执行上述定义的存储过程Proc\_credit(2020102001)。



1. 查看存储过程执行结果。



（3）修改存储过程

在navicat中点击hub->函数->Proc\_credit->设计函数

进入存储过程定义页面进行存储过程的修改，并把名字改为proc\_Credit。

（2）删除存储过程

删除存储过程proc\_Credit。



1. 自定义函数实验：自定义函数定义，自定义函数运行，自定义函数更名，自定义函数删除，自定义函数的参数传递。掌握PL/SQL和编程规范，规范设计自定义函数。
2. 定义函数查询职称为“讲师”的教师姓名：

使用函数查询 teacher 表中职称为“讲师”（professional\_title = '讲师'）的教师姓名。

DELIMITER $$

CREATE FUNCTION GetLecturers()

RETURNS VARCHAR(1000) DETERMINISTIC

BEGIN

DECLARE result VARCHAR(1000);

SELECT GROUP\_CONCAT(Tname) INTO result

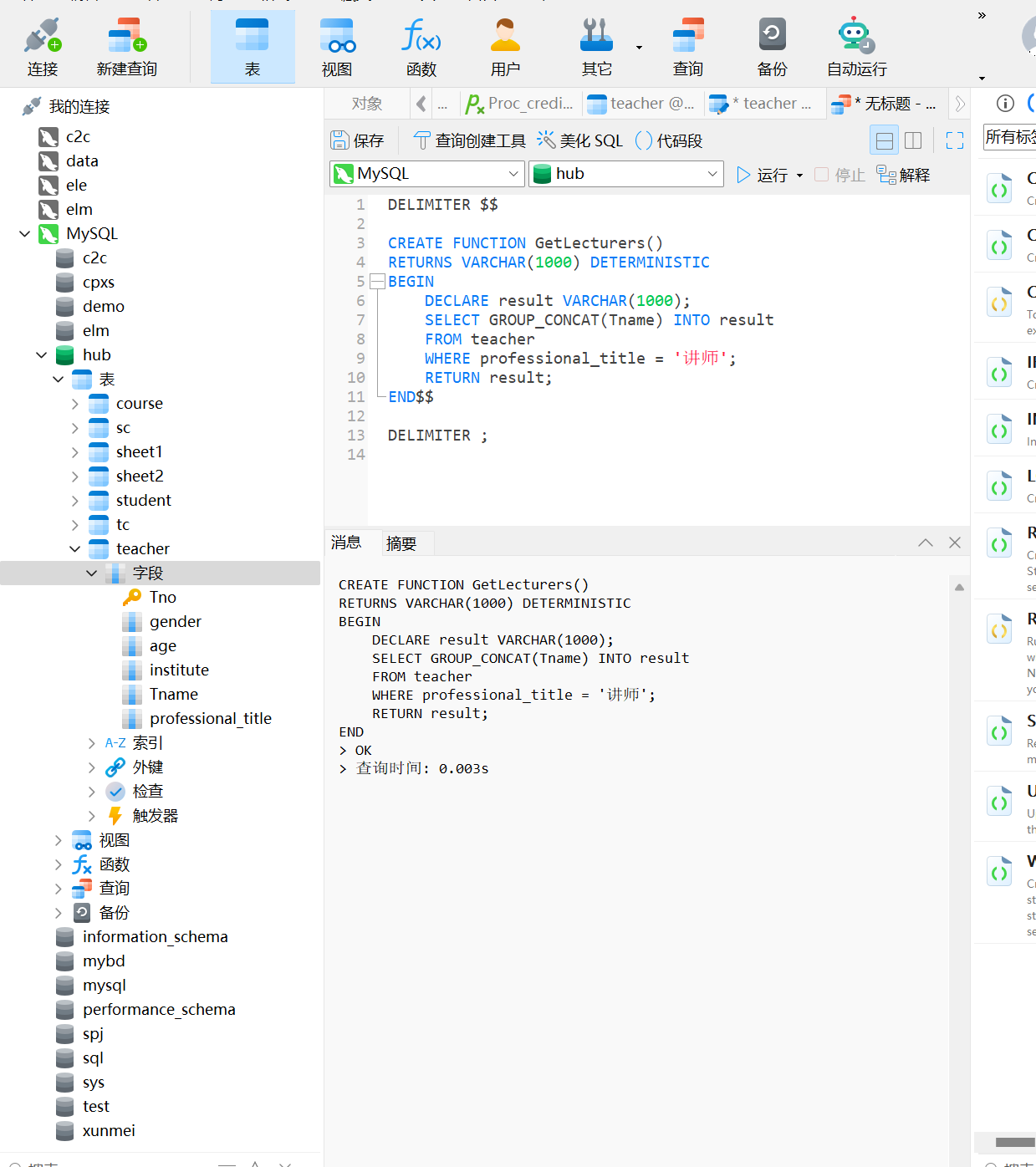
FROM teacher

WHERE professional\_title = '讲师';

RETURN result;

END$$

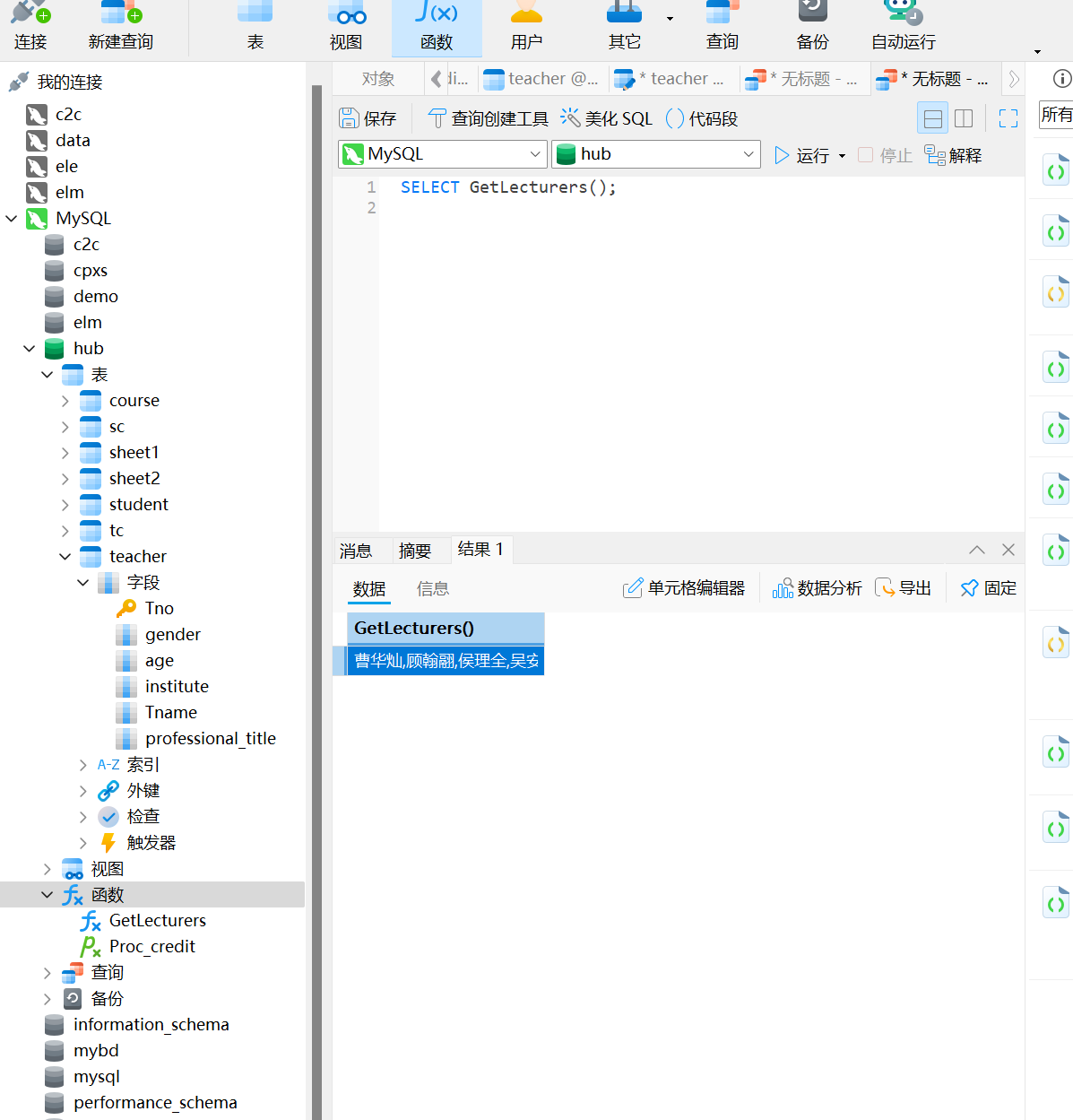
DELIMITER ;



1. 测试函数运行

调用定义的函数 GetLecturers 查询职称为“讲师”的教师姓名：

SELECT GetLecturers();



1. 使用函数更改职称为“副教授”

编写存储过程，将 professional\_title 从“讲师”修改为“副教授”。

创建存储过程：

DELIMITER $$

CREATE PROCEDURE UpdateToAssociateProfessor()

BEGIN

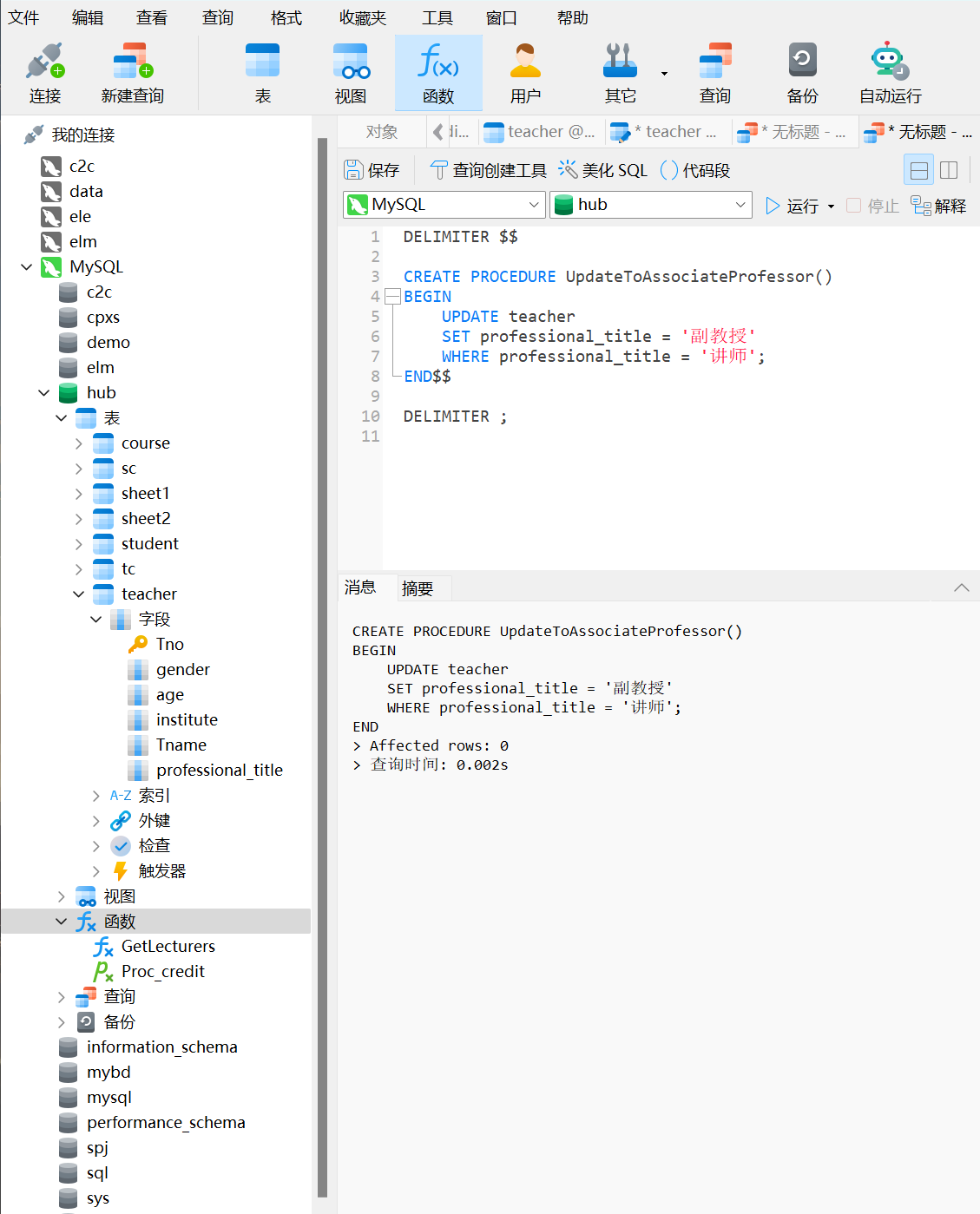
UPDATE teacher

SET professional\_title = '副教授'

WHERE professional\_title = '讲师';

END$$

DELIMITER ;

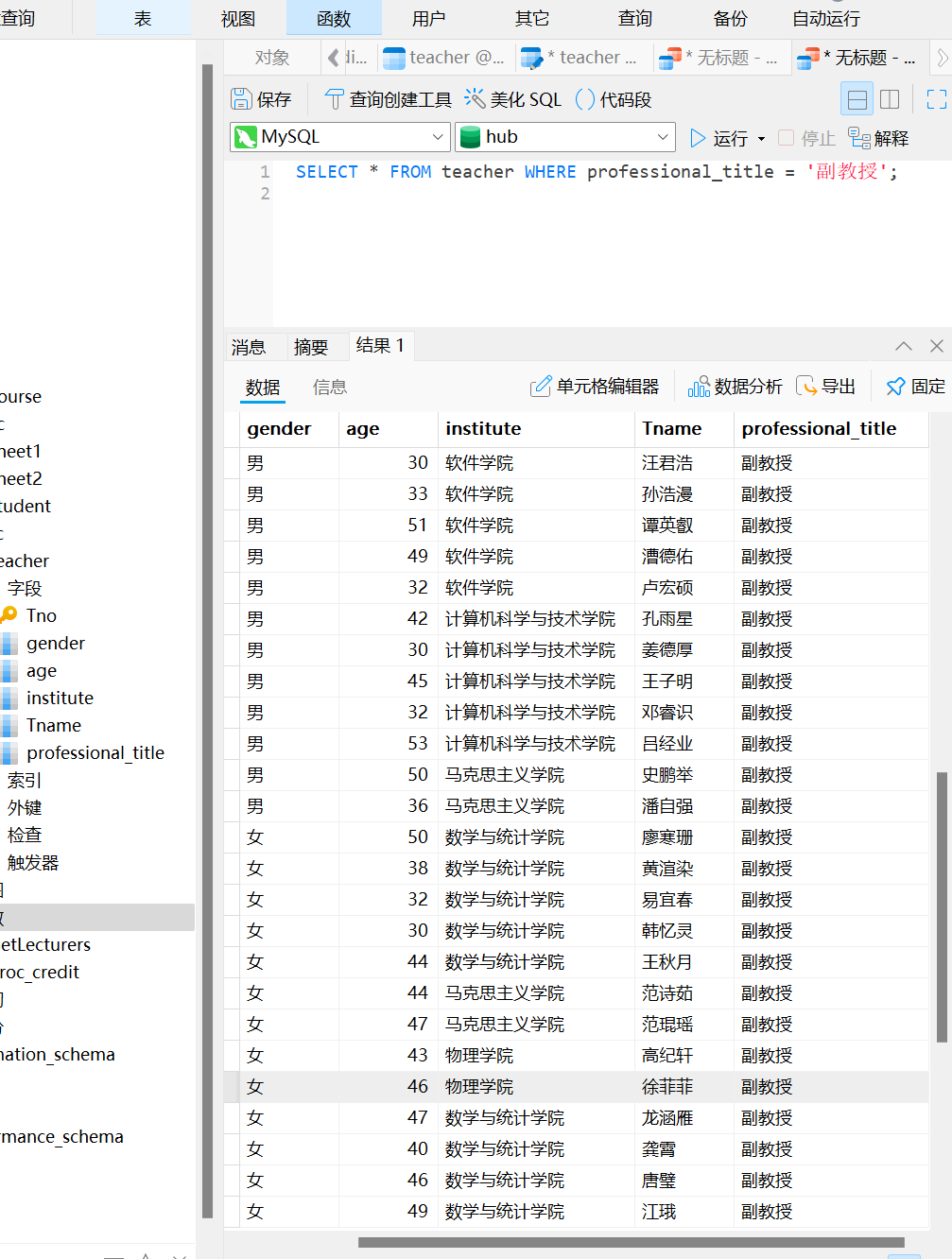


**执行存储过程**：

CALL UpdateToAssociateProfessor();

**验证更新结果**：

SELECT \* FROM teacher WHERE professional\_title = '副教授';



1. 更名自定义函数

重新创建新函数或删除旧函数：

删除旧函数：

DROP FUNCTION IF EXISTS GetLecturers;

重新创建新函数：

DELIMITER $$

CREATE FUNCTION GetAssociateProfessors()

RETURNS VARCHAR(1000) DETERMINISTIC

BEGIN

DECLARE result VARCHAR(1000);

SELECT GROUP\_CONCAT(Tname) INTO result

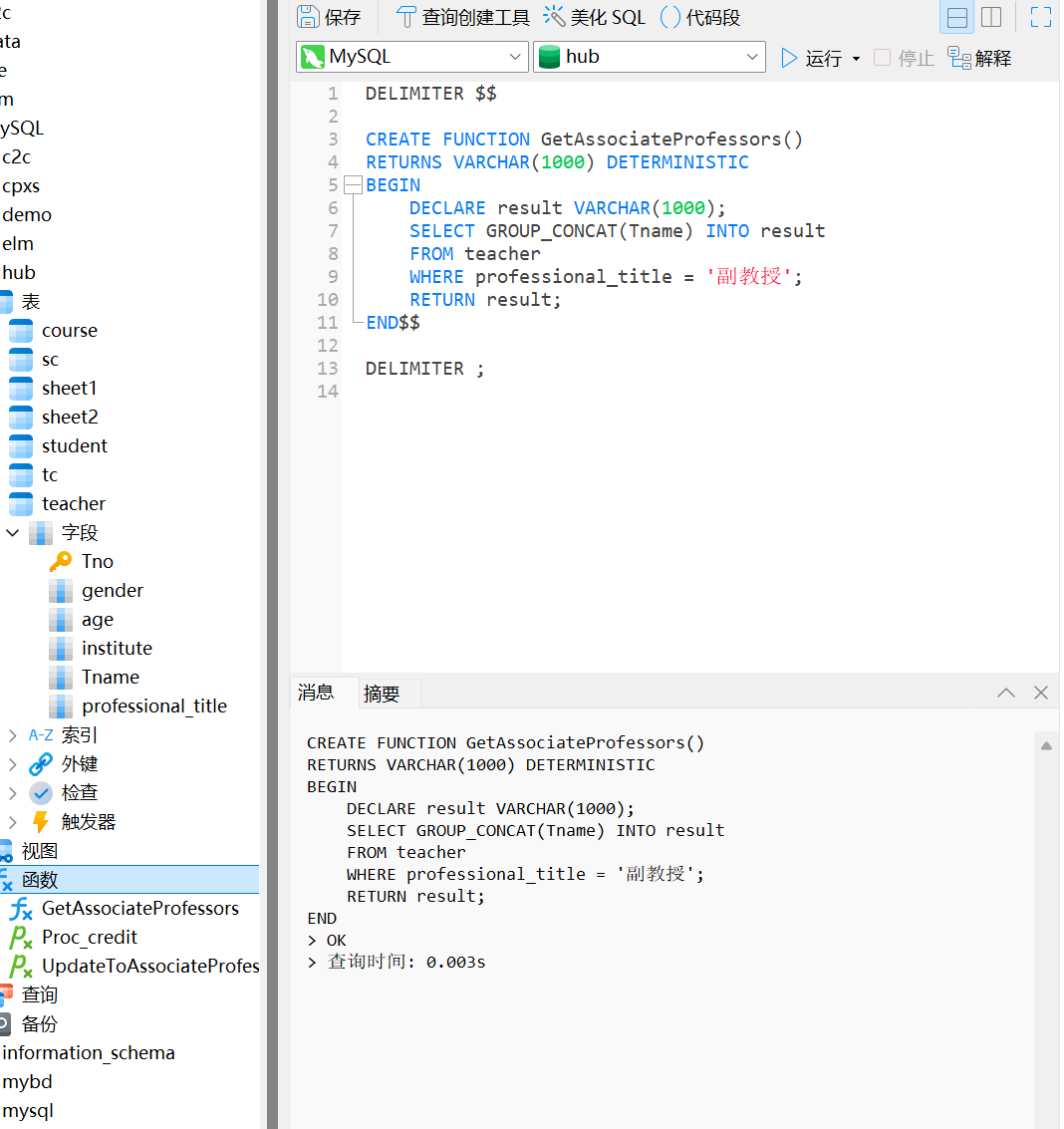
FROM teacher

WHERE professional\_title = '副教授';

RETURN result;

END$$

DELIMITER ;



1. 删除函数

DROP FUNCTION IF EXISTS GetAssociateProfessors;

删除存储过程：

DROP PROCEDURE IF EXISTS UpdateToAssociateProfessor;

1. 恢复原始数据

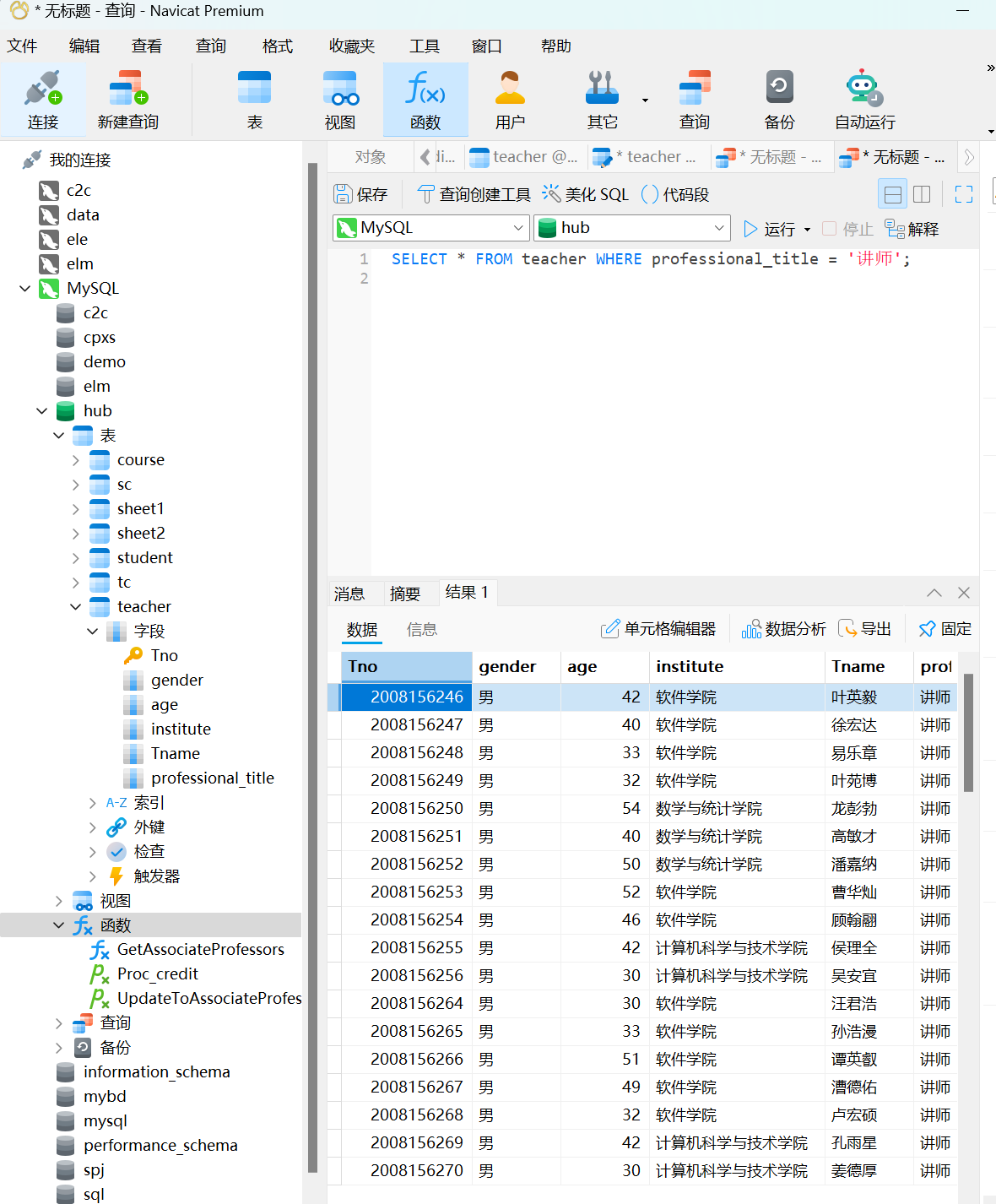
UPDATE teacher

SET professional\_title = '讲师'

WHERE professional\_title = '副教授';

验证恢复：

SELECT \* FROM teacher WHERE professional\_title = '讲师';

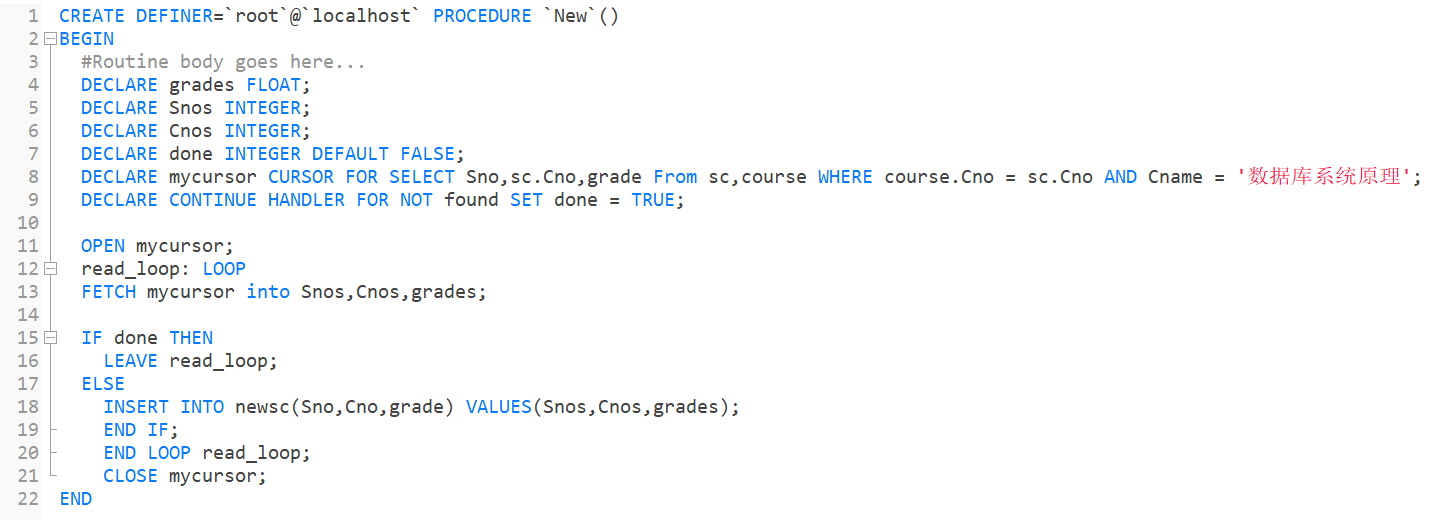


3、游标定义、游标使用。掌握各种类型游标的特点、区别与联系。

（1）普通游标

①定义一个新的表newsc，其结构与sc表完全一致，用于验证游标实验。

②定义一个存储过程，利用游标将sc表中的数据库系统原理课程的选课记录及相应成绩复制到newsc表中。



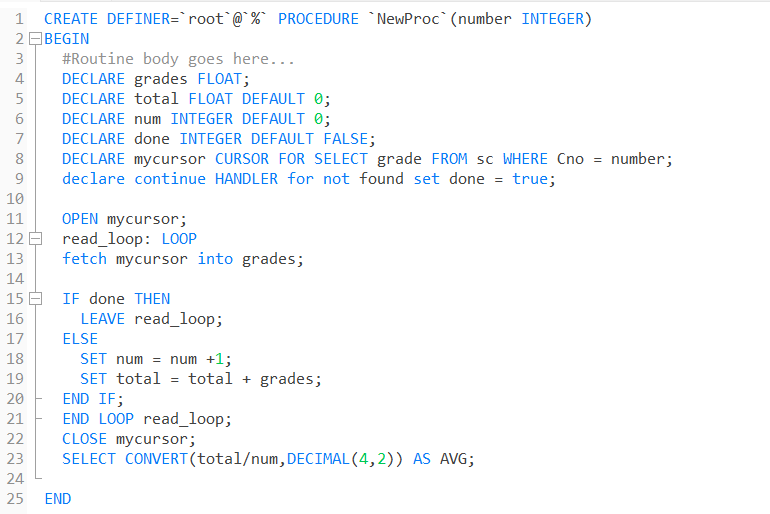
③将该过程重命名为Proc\_Cursor，执行存储过程Proc\_Cursor();



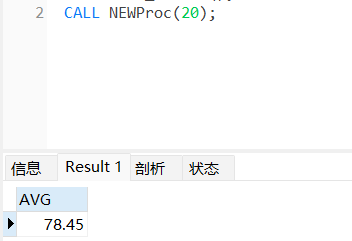
④验证存储过程的正确性，利用SQL语句查询或打开newsc表查看相关数据。

1. 带参数的游标

①定义一个存储过程，用游标计算数据库系统原理课程的平均成绩。



②执行存储过程NewProc()并验证结果。



**三、思考题**

1.试总结几种调试存储过程的方法。

1）输出调试信息：

使用 SELECT 或 SET 语句在存储过程中输出变量的值或状态，用于检查流程是否正确。

SELECT 'Debug point reached', variable\_name;

1. 调试工具

使用数据库管理工具（如 MySQL Workbench）的调试功能，可以逐步执行存储过程，查看变量状态和流程走向。

1. 通过日记记录：

在存储过程中插入调试日志，记录关键变量或操作步骤，日志表可供后续分析。

4）模拟调用测试：

使用不同的输入参数调用存储过程，并记录输出结果。

CALL procedure\_name(param1, param2);

1. 捕获错误：

使用 DECLARE ... HANDLER 捕获存储过程中的错误，便于定位问题。

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION

BEGIN

SET @error\_message = 'An error occurred.';

END;

1. 存储过程中的SELECT语句与普通的SELECT语句格式有何不同？执行方法有何不同？

**普通 SELECT 语句**：

直接运行即可在终端或工具中返回结果集。执行方法：

SELECT \* FROM table\_name WHERE condition;

**存储过程中的 SELECT 语句**：

主要用于：

* 将结果存入变量（INTO 子句）。
* 输出结果到调用者（用于 OUT 参数或返回结果集）。

执行方式：

* 定义在存储过程内，通过调用存储过程触发查询。

CREATE PROCEDURE GetTeacherNames()

BEGIN

SELECT Tname FROM teacher;

END;

CALL GetTeacherNames();

**四、课后习题**

1、尝试定义无参数的存储过程、有局部变量的存储过程、有输出参数的存储过程，执行相应存储过程并查看存储过程执行结果。

1）定义无参数的存储过程：

无参数存储过程：查询 teacher 表中所有教师信息。

创建存储过程：

DELIMITER $$

CREATE PROCEDURE GetAllTeachers()

BEGIN

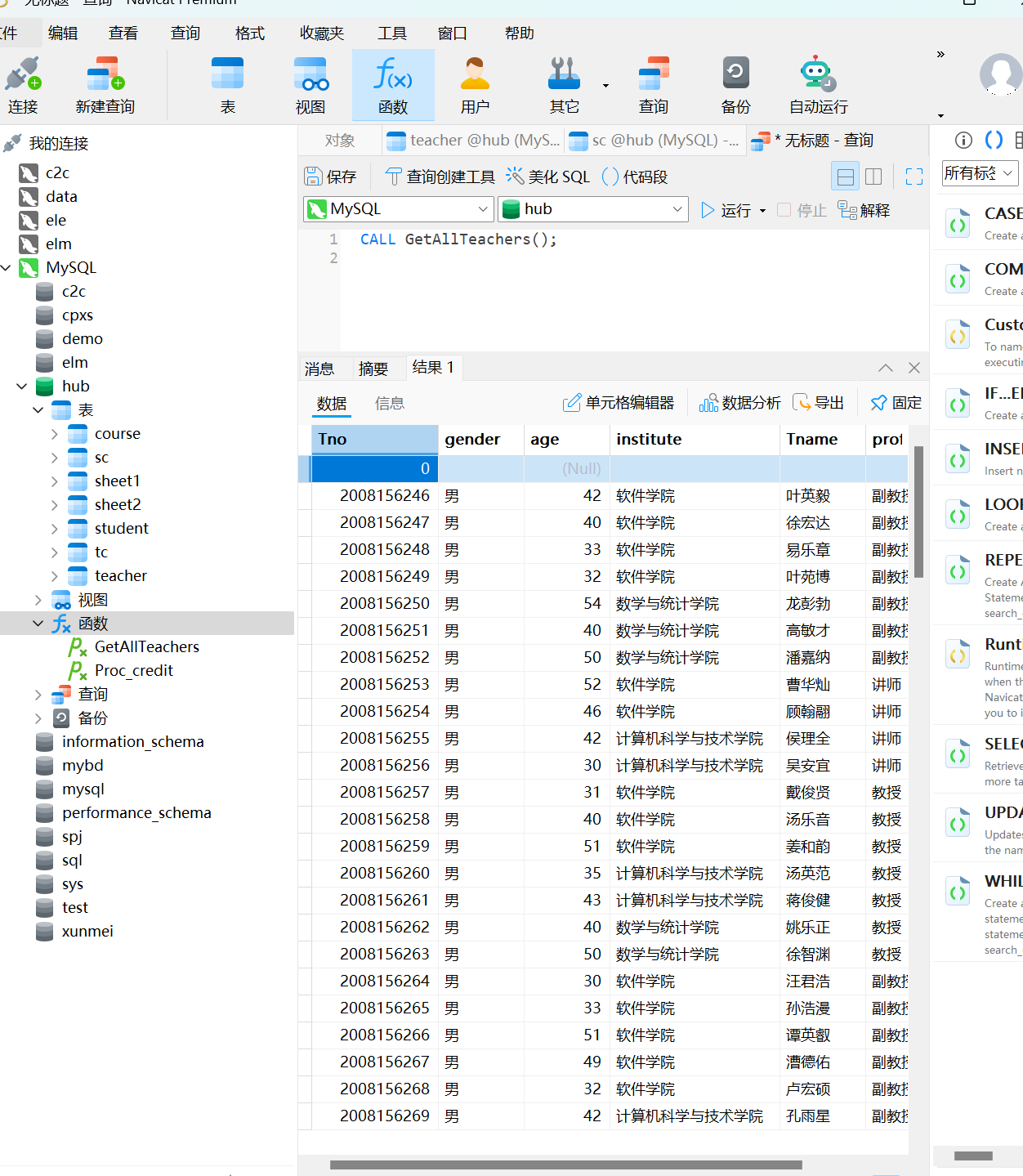
SELECT \* FROM teacher;

END$$

DELIMITER ;

执行存储过程：

CALL GetAllTeachers();



1. 定义有局部变量的存储过程

有局部变量的存储过程：计算 teacher 表中职称为“讲师”的教师人数。

创建存储过程：

DELIMITER $$

CREATE PROCEDURE CountLecturers()

BEGIN

DECLARE lecturer\_count INT; -- 声明局部变量

SELECT COUNT(\*) INTO lecturer\_count

FROM teacher

WHERE professional\_title = '讲师';

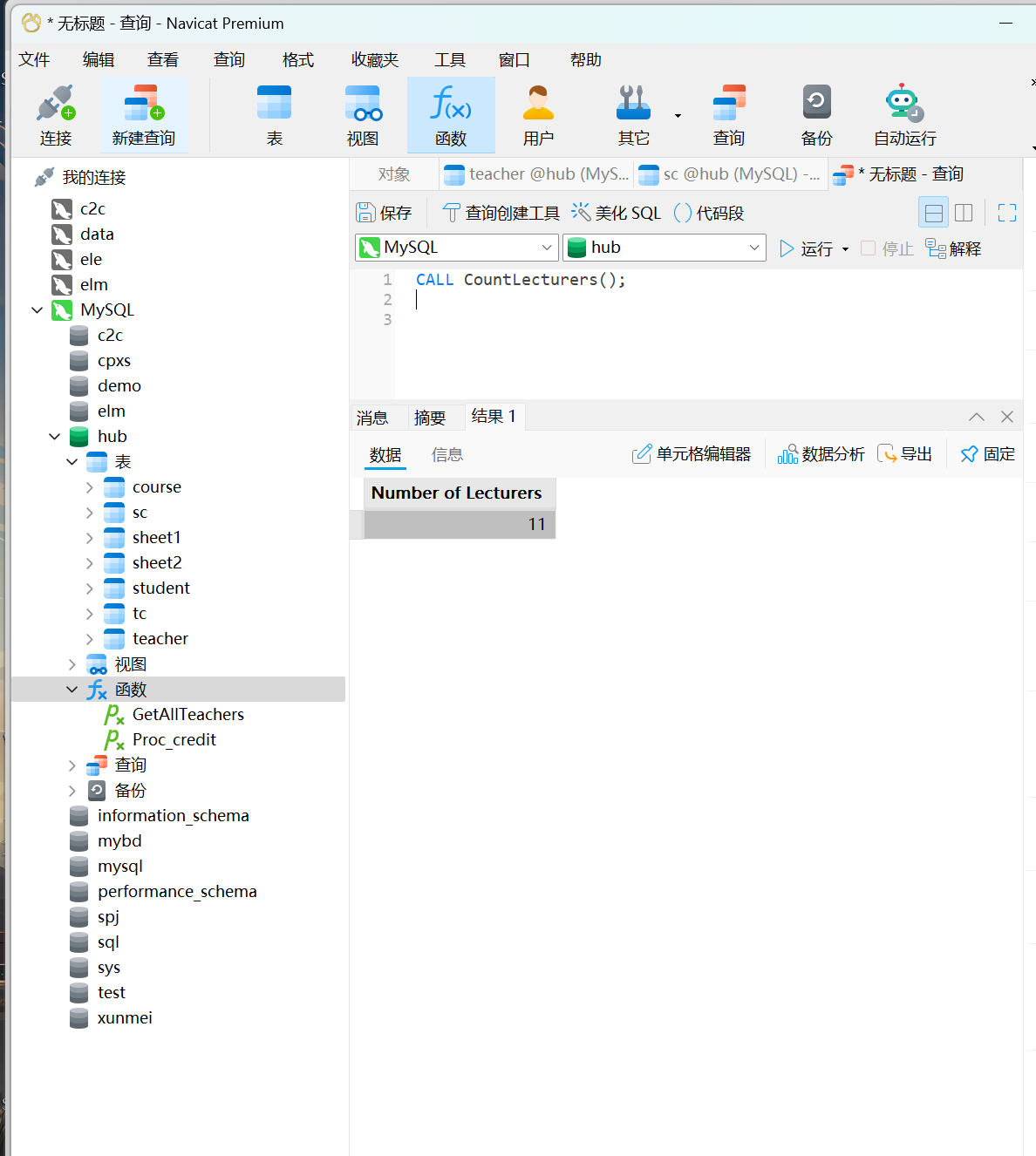
SELECT lecturer\_count AS 'Number of Lecturers'; -- 返回结果

END$$

DELIMITER ;

执行存储过程：

CALL CountLecturers();



1. 定义有输出参数的存储过程

有输出参数的存储过程：根据教师编号返回对应教师的姓名。

创建存储过程：

DELIMITER $$

CREATE PROCEDURE GetTeacherName(IN teacher\_id VARCHAR(10), OUT teacher\_name VARCHAR(50))

BEGIN

SELECT Tname INTO teacher\_name

FROM teacher

WHERE Tno = teacher\_id;

END$$

DELIMITER ;

执行存储过程：

-- 声明一个变量用于接收输出结果

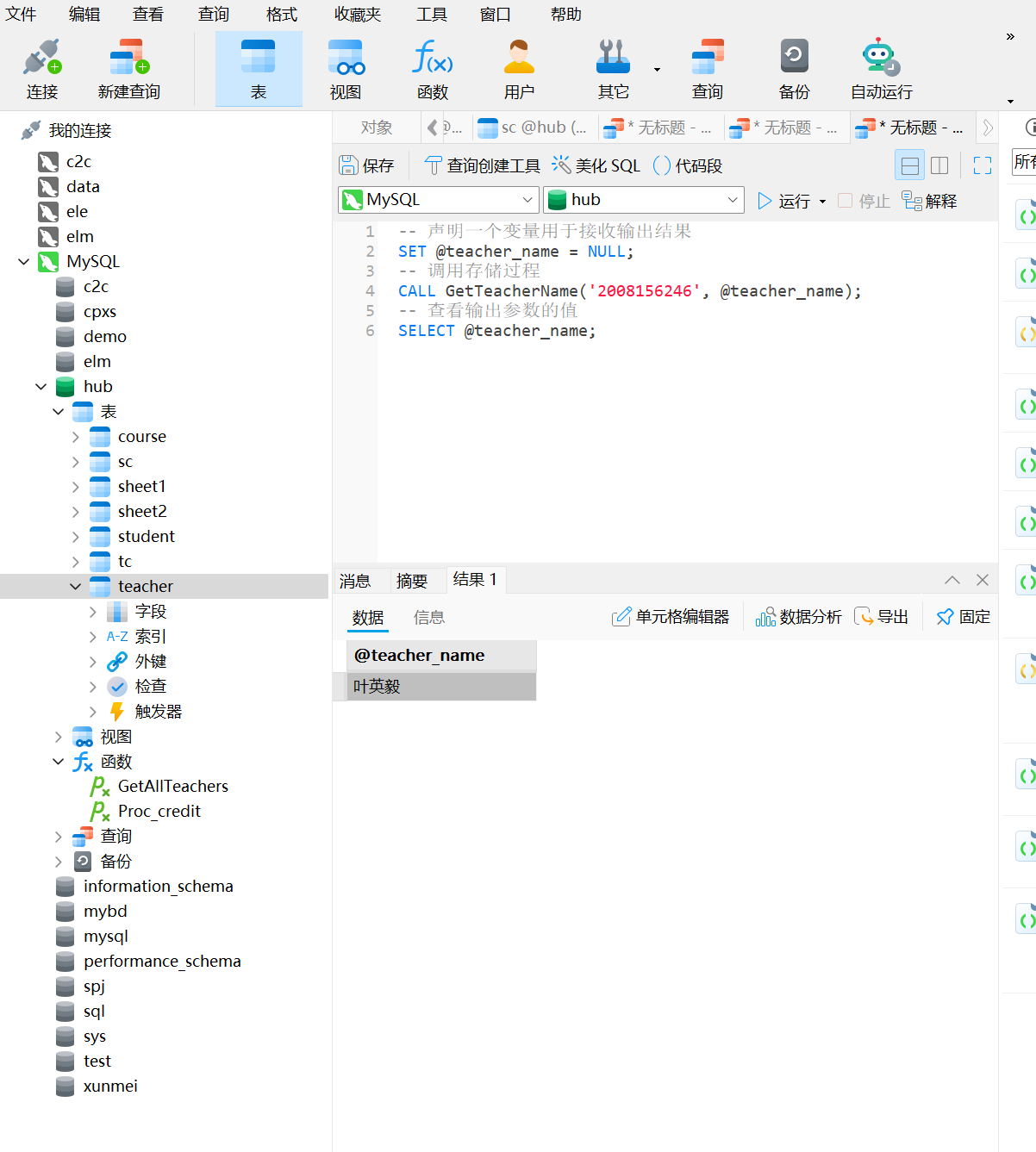
SET @teacher\_name = NULL;

-- 调用存储过程

CALL GetTeacherName('2008156246', @teacher\_name);

-- 查看输出参数的值

SELECT @teacher\_name;



2、了解REFCURSOR类型游标，阐述与普通游标有什么不同的地方。

1）定义与使用上的区别

**普通游标**：

需要在声明时指定查询语句（静态查询）。

游标与SQL查询绑定，无法动态改变。

只能在定义的存储过程中使用。

**REF CURSOR**：

是一个指向查询结果集的指针（动态游标）。

查询语句可以在运行时动态指定。

可以作为参数在存储过程或函数之间传递。

1. 静态与动态的区别

**普通游标**是静态的，查询语句在编译时就确定。

**REF CURSOR**是动态的，运行时可以动态构建查询并执行。

1. 可传递性

**普通游标：**

仅限在定义的PL/SQL块内使用，无法跨程序单元共享。

**REF CURSOR**：

可以作为参数传递给其他存储过程或函数，适合构建灵活的程序结构。

1. 可读性

**普通游标**：

是只读的，只能用于读取数据。

**REF CURSOR**：

可以是只读，也可以是可更新的（通过设置游标模式）。

5）内存管理

**普通游标**：

Oracle自动管理游标生命周期，超出作用域后会自动关闭。

**REF CURSOR**：

由用户显式打开和关闭，需要手动管理其生命周期，否则可能导致内存泄漏。

**实验总结：**

通过本次实验，我深入学习了PL/SQL编程语言的核心概念和实际应用，取得了以下收获：

**·熟练掌握了存储过程和自定义函数的设计与实现**：能够灵活使用存储过程和函数来封装数据库逻辑，提高了数据库操作的效率和复用性。

**·深入理解了游标的定义和使用**：通过实验，明确了普通游标和REF CURSOR的区别，熟悉了游标按行操作的具体实现，并能在实际场景中选择合适的游标类型。

**·对SQL按结果集操作和PL/SQL按行操作的区别有了清晰的认识**：理解了结果集的批量处理和逐行处理的适用场景，掌握了两者的联系和区别。

**·综合提高了数据库编程能力**：通过动手实践，加深了对PL/SQL的语法和特点的理解，为后续复杂数据库系统的开发奠定了坚实基础。

本次实验不仅提升了我的PL/SQL编程能力，也强化了对数据库系统设计与实现的整体理解。

**教师评语及成绩**：