数据库实验报告 实验十三 安全综合案例实践

姓名	学号	班级	课室
熊明	20305055	计科5班	D503

一、实验目的

通过完成一个综合案例的实验，加深对数据库安全性控制的理解。

二、实验环境

数据库：Mysql

图形化工具：Navicat Premium 16

三、实验内容

PostgreSQL数据库安全性控制简明教程

PostgreSQL是一款强大的开源关系型数据库管理系统，为了确保数据库的安全性，我们可以采取一系列措施来进行安全性控制。以下是一个简明的教程，涵盖了一些基本的安全性控制措施：

1. 访问控制

1.1 创建登录用户和密码

首先，创建数据库的登录用户，并为其分配密码：

```
CREATE USER your_user WITH PASSWORD 'your_password';
```

1.2 授予权限

根据需要，授予用户相应的权限，例如：

```
GRANT SELECT, INSERT, UPDATE, DELETE ON your_table TO your_user;
```

2. 角色管理

2.1 创建角色

使用角色进行权限管理是一种有效的方式。创建角色并授予权限：

```
CREATE ROLE data_admin;
GRANT data_admin TO your_user;
```

2.2 分配角色

将角色分配给用户，以便用户继承角色的权限：

```
GRANT data_admin TO your_user;
```

3. SSL加密

启用SSL加密以确保数据在传输过程中的安全：

```
ssl = on
```

4. 存储过程和函数的权限控制

4.1 创建存储过程

```
CREATE OR REPLACE PROCEDURE your_procedure()
AS
$$
BEGIN
    -- Your logic here （e.x. SELECT * FROM your_table WHERE condition;）
END;
$$
LANGUAGE plpgsql;
```

4.2 授予权限

为用户授予执行存储过程的权限：

```
GRANT EXECUTE ON PROCEDURE your_procedure() TO your_user;
```

5. 行级安全

使用行级安全策略限制用户对数据的访问：

```
ALTER TABLE your_table ENABLE ROW LEVEL SECURITY;
CREATE POLICY your_policy
    USING (your_condition)
    FOR ALL
    USING (true);
```

6. 审计日志

启用审计日志以跟踪数据库活动：

```
logging_collector = on
log_statement = 'all'
log_directory = '/var/log/postgresql/'
```

以上只是一个入门级的教程，实际上，数据库安全性控制涉及到更多方面，包括定期备份、更新数据库软件、监控异常活动等。在实际应用中，应根据具体需求和环境进行更详细的安全性配置。

四、课内实验

问题：赵老师当了2008级电子商务班的班主任，他要能查到全校的课程信息以及本班学生的选课信息，如何让他有权查到这些信息?
主要内容如下：

> 1. 登录管理
>
> 为新老师创建登录账号logzhao,验证该账号与数据库的连接访问是否正确?
>
> 2. 对用户授权
>
> 问题1:试解决赵老师能查询本年级学生的选课信息?
>
> 首先创建2008级学生选课信息的视图 scview,把访问该视图的权限授予赵老师，最后 验证赵老师能否访问该视图?
>
> 问题2:试解决让赵老师了解某课程的选课情况?
>
> 首先创建能查询指定课程选课信息的存储过程 scpro,把执行该存储过程的权限授予赵老师，最后验证赵老师能否执行存储过程?
>
> 补充内容：撤销赵老师查询某课程的选课情况，再验证赵老师能否执行存储过程?
>
> 3. 角色管理
>
> 问题：假如学校新增10个辅导员，都要在student表中添加、修改和删除学生，要个个 设置权限，方便吗?
>
> 可以考虑利用数据库的角色管理来实现：
>
> 首先创建辅导员角色m_role,然后对角色进行插入操作授权，再创建各个辅导员的登录以及对应的登录用户，使这些用户成为角色成员，再验证用户是否有插入操作的权限?
>
> 还可以考虑应用程序角色来实现：
>
> 创建应用程序角色，激活该角色，对其进行插入操作的授权，验证是否具有该操作的权限?

1. 创建赵老师的用户，将查询全校的课程信息的权限赋予该用户

```
-- 创建用户logzhao
CREATE USER 'logzhao'@'%' IDENTIFIED BY 'zhao';

-- 允许查到全校的课程信息以及本班学生的选课信息
GRANT SELECT ON school.courses TO 'logzhao'@'%';
```

	数据库	名	Grant Option	Index	Insert	Lock Tables	References	Select	Show View	Trigger	Update
▸	school	courses	☐	☐	☐	☐	☐	☑	☐	☐	☐

2. 创建2008级学生选课信息的视图 scview,把访问该视图的权限授予赵老师

因为数据集中没有2008级的学生查不出信息

```
SELECT *
FROM choices
WHERE sid IN(
SELECT sid
FROM students
WHERE grade = 2008)
```

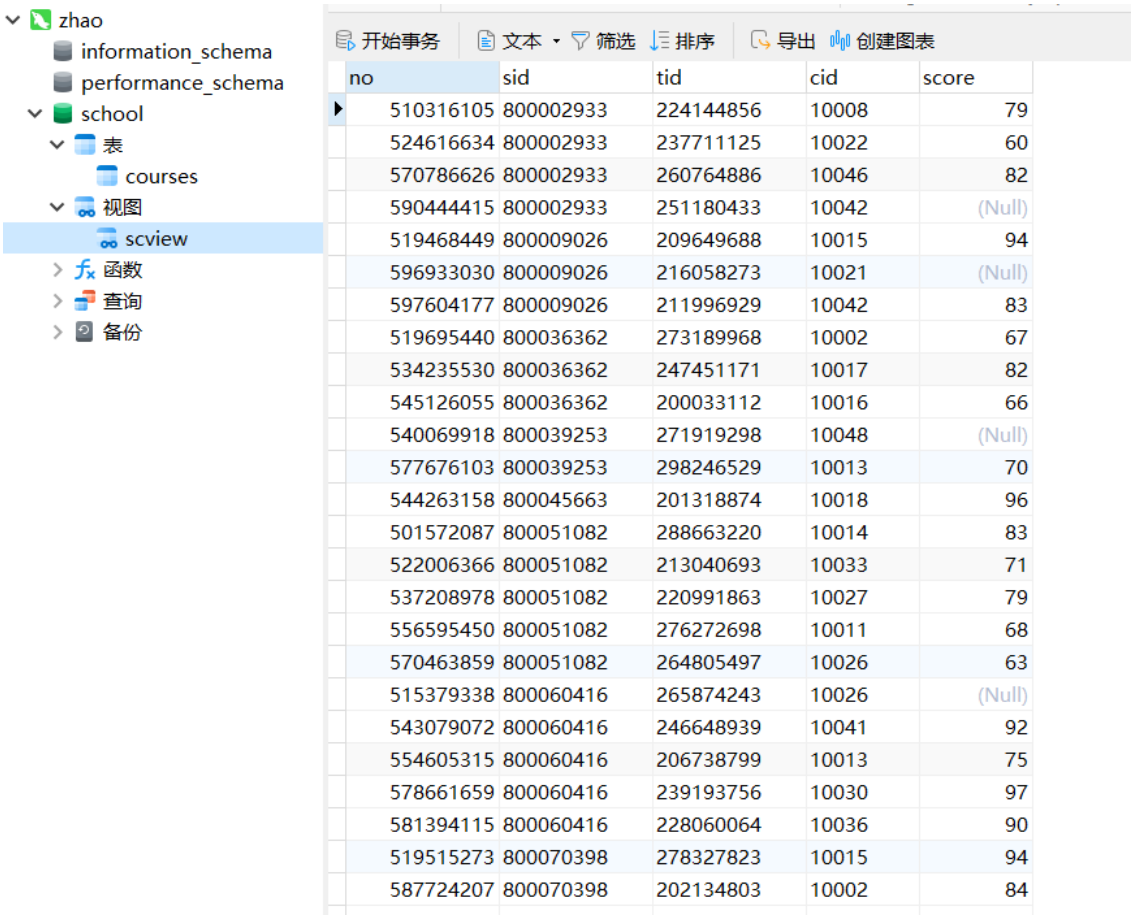
得到结果是空的：

no	sid	tid	cid	score
(N/A)	(N/A)	(N/A)	(N/A)	(N/A)

所以用2002级模拟有数据的2008级情况：

```
-- 创建2008级学生选课信息的视图 scview,把访问该视图的权限授予赵老师
CREATE OR REPLACE VIEW scview(no,sid,tid,cid,score) AS
    SELECT *
    FROM choices
    WHERE sid IN(
        SELECT sid
        FROM students
        WHERE grade = 2002
    )
with check option;
GRANT SELECT ON school.scview TO 'logzhao'@'%';
```

登录到赵老师账户，验证该权限



no	sid	tid	cid	score
510316105	800002933	224144856	10008	79
524616634	800002933	237711125	10022	60
570786626	800002933	260764886	10046	82
590444415	800002933	251180433	10042	(Null)
519468449	800009026	209649688	10015	94
596933030	800009026	216058273	10021	(Null)
597604177	800009026	211996929	10042	83
519695440	800036362	273189968	10002	67
534235530	800036362	247451171	10017	82
545126055	800036362	200033112	10016	66
540069918	800039253	271919298	10048	(Null)
577676103	800039253	298246529	10013	70
544263158	800045663	201318874	10018	96
501572087	800051082	288663220	10014	83
522006366	800051082	213040693	10033	71
537208978	800051082	220991863	10027	79
556595450	800051082	276272698	10011	68
570463859	800051082	264805497	10026	63
515379338	800060416	265874243	10026	(Null)
543079072	800060416	246648939	10041	92
554605315	800060416	206738799	10013	75
578661659	800060416	239193756	10030	97
581394115	800060416	228060064	10036	90
519515273	800070398	278327823	10015	94
587724207	800070398	202134803	10002	84

左上角的表示用`logzhao`账户登录到数据库，然后可以查看scview和courses表，说明权限赋予成功。

3. 创建能查询指定课程选课信息的存储过程 scpro，将执行该存储过程的权限授予赵老师

```
DELIMITER //

CREATE PROCEDURE scpro (IN sname_param VARCHAR(50))
BEGIN
    SELECT *
```

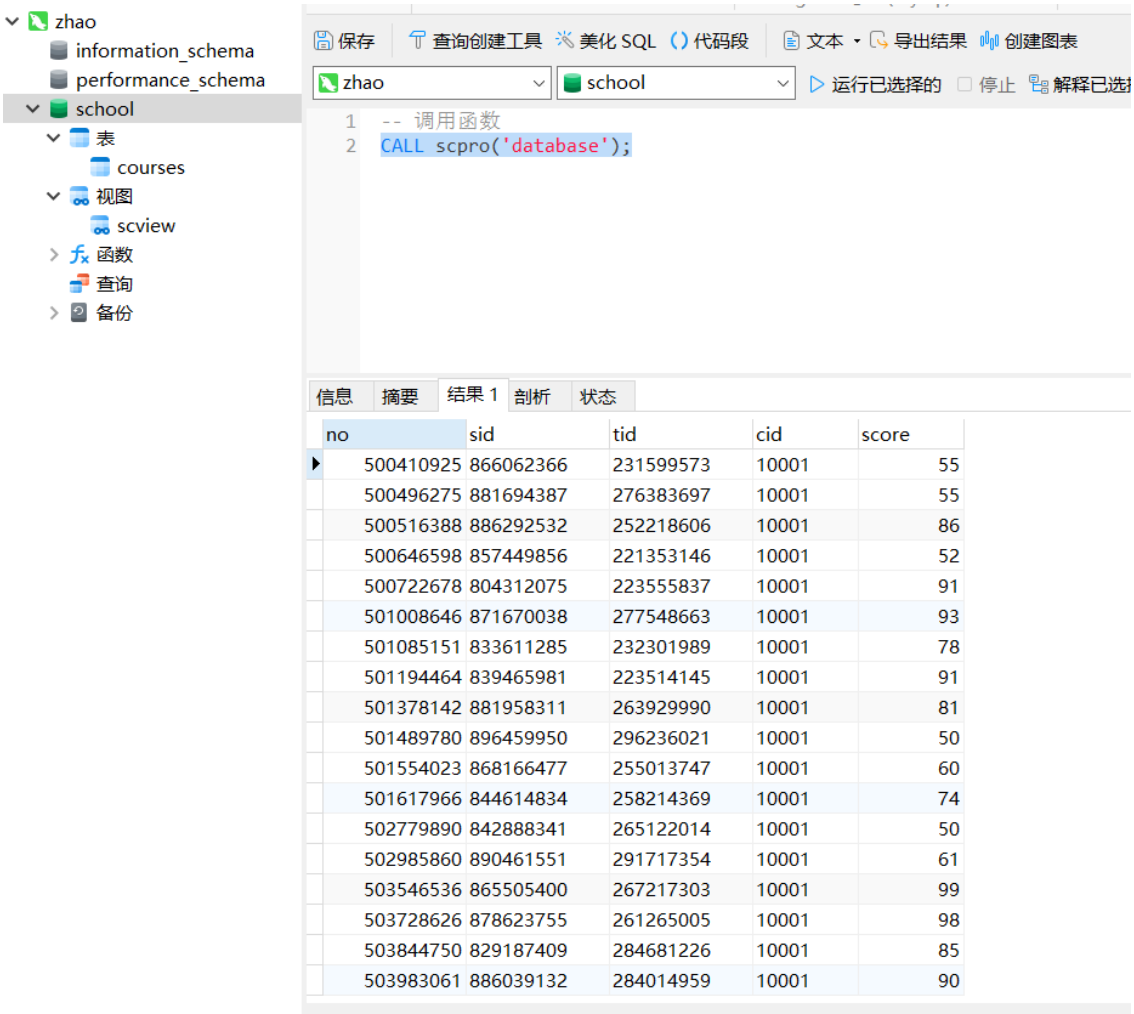
```
    FROM scview
    WHERE cid IN (
        SELECT cid
        FROM courses
        WHERE cname = sname_param
    );
END //

DELIMITER ;

-- CALL scpro('C++');
GRANT EXECUTE ON PROCEDURE school.scpro TO 'logzhao'@'%';
```

实现了一个存储过程`scpro`，输入课程名就能看到本班学生对于该课程的选课信息

在赵老师的账户下建立查询，尝试调用该存储过程：

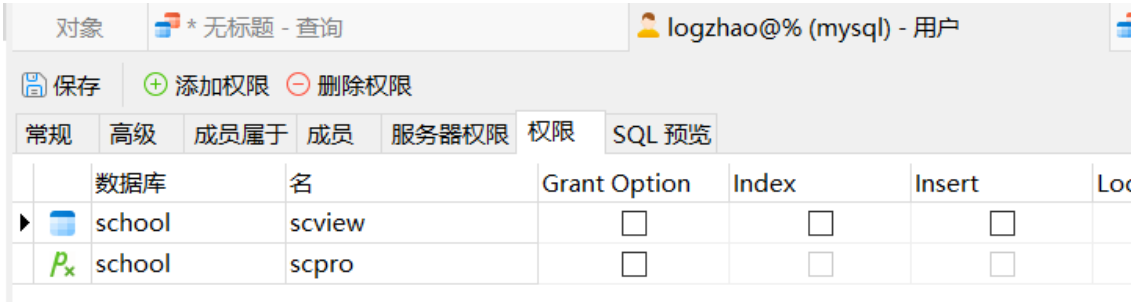


可以看到能成功查询

4. 撤销赵老师查询某课程的选课情况，再验证赵老师能否执行存储过程?

```
-- 撤销赵老师查询某课程的选课情况，再验证赵老师能否执行存储过程
REVOKE SELECT ON school.courses FROM 'logzhao'@'%';
```

然后查看用户权限：

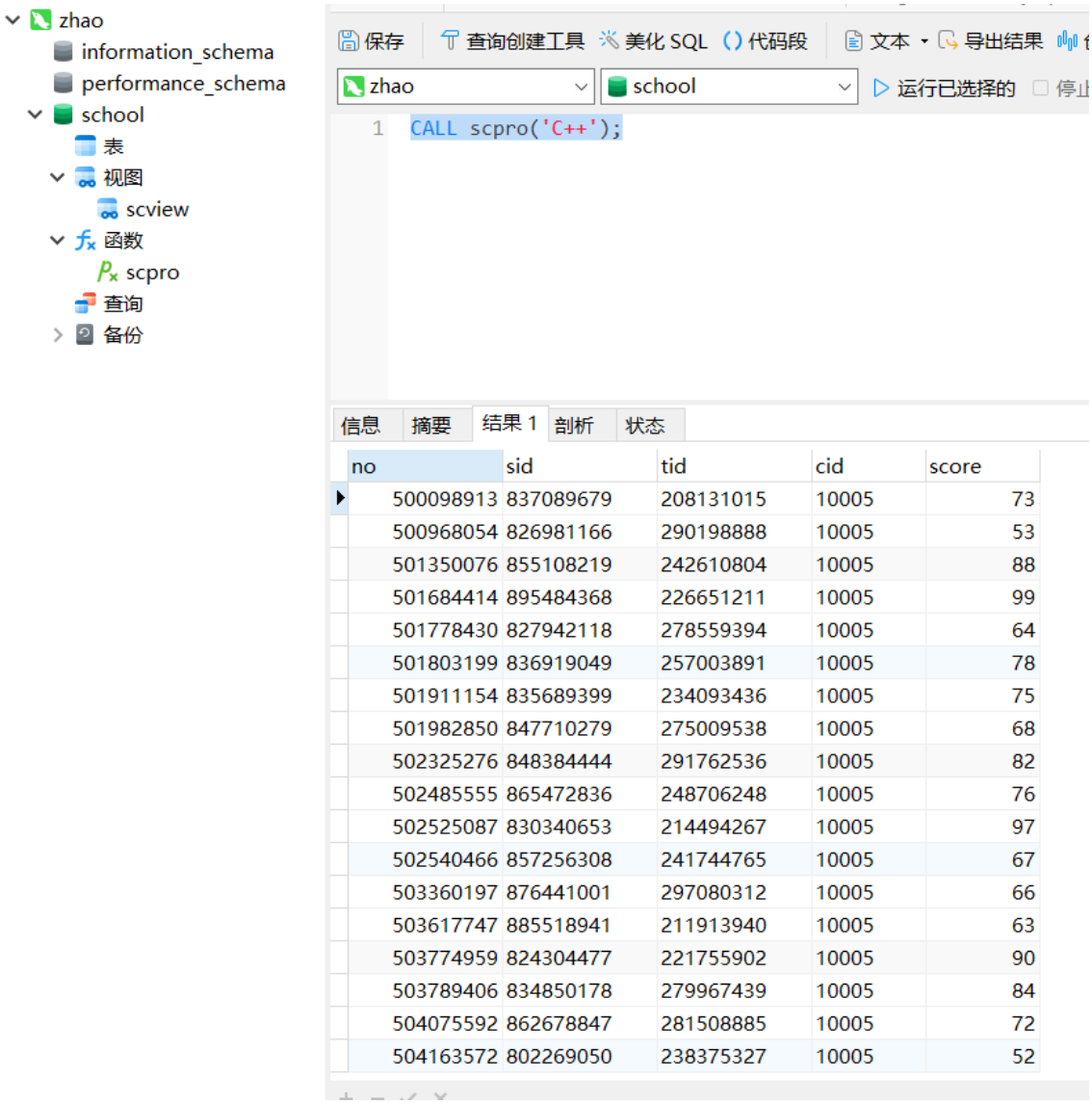


可以看到已经没有对courses的访问权限

登录赵老师用户，重新调用存储过程：

```
CALL scpro('C++');
```

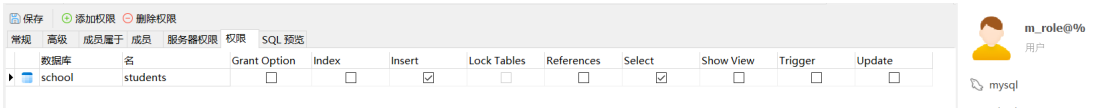
得到结果：



说明即使没有对courses的访问权限，依然能够调用存储过程。

5. 创建辅导员角色m_role,然后对角色进行插入操作授权

```
-- 创建辅导员角色m_role,然后对角色进行插入操作授权
CREATE ROLE m_role;
GRANT SELECT,INSERT ON school.students TO m_role;
```



`m_role`权限如图

6. 创建各个辅导员的登录以及对应的登录用户，使这些用户成为角色成员

   以一个为例，其余9个手动添加即可：

```
-- 创建用户
CREATE USER 'counselor1'@'%' IDENTIFIED BY '1';
-- 将用户添加为角色成员
GRANT m_role TO 'counselor1'@'%';
```

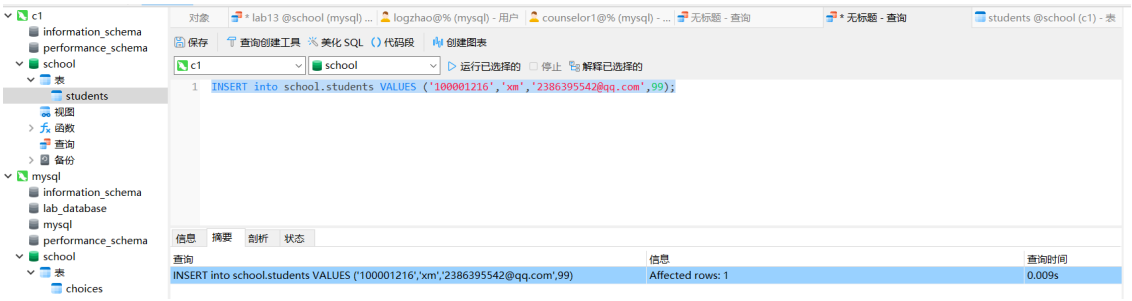
实验遇到问题，因为`mysql`需要手动开启角色自动激活，否则只会激活默认角色，开启命令如下：

```
-- 设置开启
set global activate_all_roles_on_login=ON;
-- 查看是否开启
show variables like 'activate_all_roles_on_login';
```

最后登入counselor1账户，插入一个student成员：

```
INSERT into school.students VALUES
('100001216','xm','2386395542@qq.com',99);
```

结果如下：



该查询在c1连接的权限下，成功插入一行记录