

Contents

数据库实验 10 - 安全综合案例	1
实验目的	1
PostgreSQL 数据库安全性控制简明教程	1
1. 访问控制	1
2. 角色管理	1
3. SSL 加密	1
4. 存储过程和函数的权限控制	2
5. 行级安全	2
6. 审计日志	2
课内实验	2
1. 登录管理	2
2. 对用户授权	3
3. 角色管理	10

数据库实验 10 - 安全综合案例

实验目的

通过完成一个综合案例的实验，加深对数据库安全控制的理解。

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。

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

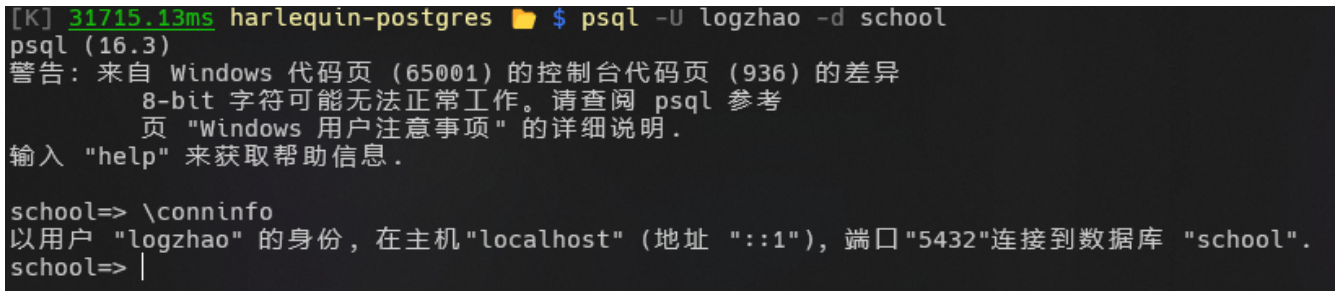
为新老师创建登录账号 logzhao，验证该账号与数据库的连接访问是否正确？

```
CREATE USER logzhao WITH PASSWORD 'password123';
```



验证连接：使用以下命令尝试连接到数据库：

```
psql -U logzhao -d school -h 127.0.0.1 -p 5432
```



可以看见连接成功。查看登录的账号信息，账号信息符合预期，说明账号创建正确。

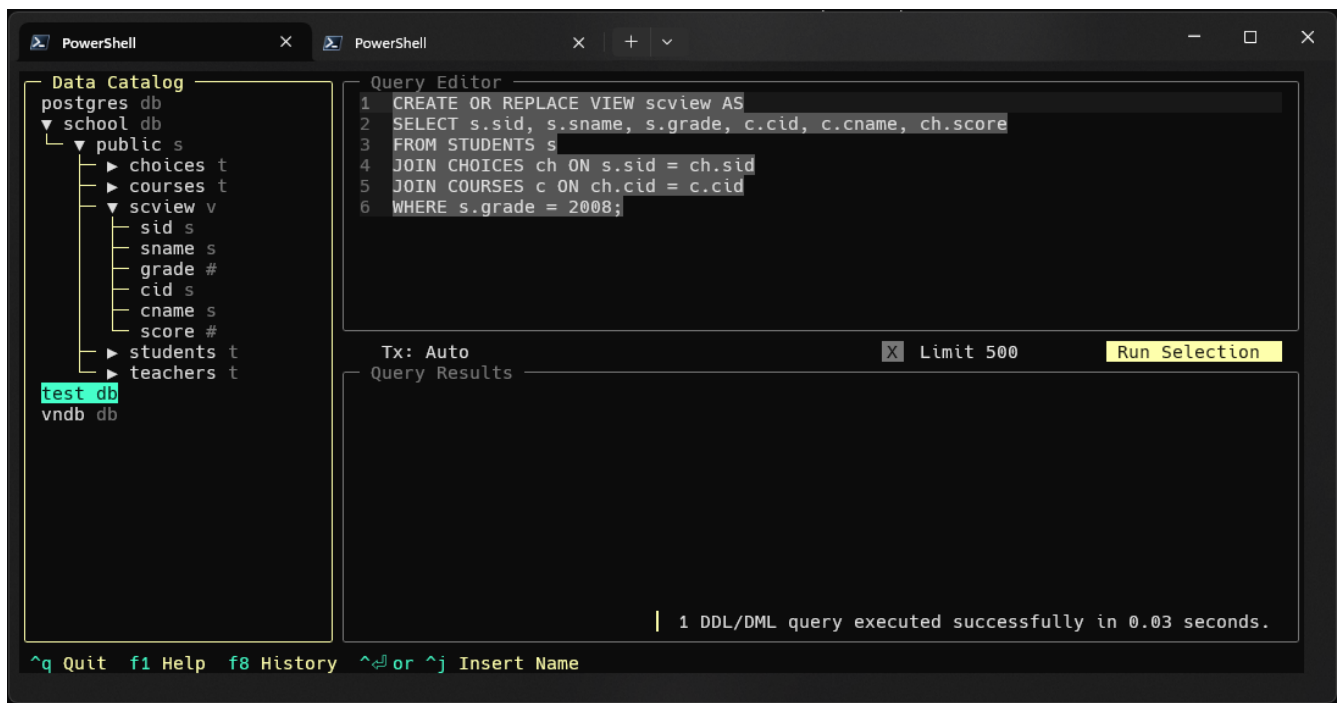
2. 对用户授权

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？

- 创建视图：

```
CREATE OR REPLACE VIEW scview AS
SELECT s.sid, s.sname, s.grade, c.cid, c.cname, ch.score
FROM STUDENTS s
JOIN CHOICES ch ON s.sid = ch.sid
JOIN COURSES c ON ch.cid = c.cid
WHERE s.grade = 2008;
```



- 把访问权限授予赵老师:

```
GRANT SELECT ON scview TO logzhao;
```

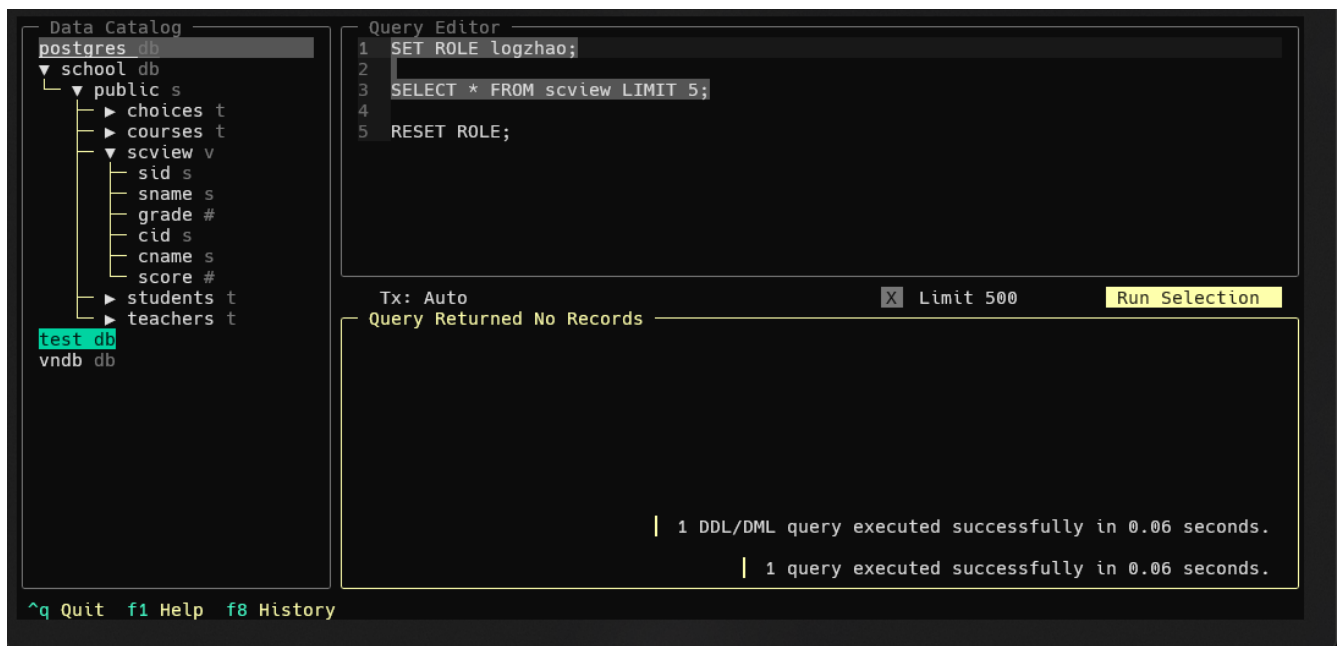


- 验证赵老师能否访问该视图:

```
SET ROLE logzhao;

SELECT * FROM scview LIMIT 5;

-- back to default role
RESET ROLE;
```



可见，访问语句执行正确。

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？

- 创建存储过程：

在一个 PROCEDURE 中只进行 select 而不对得到的值进行处理在 postgres 中是不被允许的，因此这里定义了一个临时的表，并将得到的结果插入到临时表中。

```
CREATE OR REPLACE PROCEDURE scpro(IN course_id VARCHAR)
LANGUAGE plpgsql
AS $$
BEGIN
    CREATE TEMPORARY TABLE IF NOT EXISTS temp_results (
        sid VARCHAR,
        sname VARCHAR,
        cname VARCHAR,
        score NUMERIC
    );

    DELETE FROM temp_results;

    INSERT INTO temp_results (sid, sname, cname, score)
    SELECT s.sid, s.sname, c.cname, ch.score
    FROM STUDENTS s
    JOIN CHOICES ch ON s.sid = ch.sid
    JOIN COURSES c ON ch.cid = c.cid
    WHERE c.cid = course_id;
END;
$$;
```

```
5
  Run | Copy
✓ 4 CREATE OR REPLACE PROCEDURE scpro(IN course_id VARCHAR)
3 LANGUAGE plpgsql
2 AS $$
1 BEGIN
21 CREATE TEMPORARY TABLE IF NOT EXISTS temp_results (
1   sid VARCHAR,
2   If you are an premium user, can show definition by hover
3   cname VARCHAR,
4   score NUMERIC
5 );
6
7 DELETE FROM temp_results;
8
9 INSERT INTO temp_results (sid, sname, cname, score)
10 SELECT s.sid, s.sname, c.cname, ch.score
11 FROM STUDENTS s
12 JOIN CHOICES ch ON s.sid = ch.sid
13 JOIN COURSES c ON ch.cid = c.cid
14 WHERE c.cid = course_id;
15 END;
16 $$; 288ms
17
18
  Run | New Tab
19 GRANT EXECUTE ON PROCEDURE scpro(VARCHAR) TO logzhao;
20
```

Result

Search Results

Cost: 288ms

Execution completed in 288ms

- 把执行权限授予赵老师:

```
GRANT EXECUTE ON PROCEDURE scpro(VARCHAR) TO logzhao;
```

```
1  Run | New Tab
✓ 40 GRANT EXECUTE ON PROCEDURE scpro(VARCHAR) TO logzhao; 3ms
1
2  CALL scpro('10001')
```

If you are an premium user, can show definition by hover

Result

Search Results

Export

Cost: 3m

Execution completed in 3ms

- 验证赵老师能否执行存储过程:

```
SET ROLE logzhao;

CALL scpro('10001');

RESET ROLE;
```

直接执行，发现赵老师没有访问 temp_results 表的权限，因此无法执行存储过程。

```
1 SET ROLE logzhao; 1ms
2 CALL scpro('10001') 4ms permission denied for table temp_results
```

Result

Search Results

Cost: 4ms

permission denied for table temp_results

赋予权限后，发现没有权限访问 school 中的几个表格还是不够：

```
1
43 RESET ROLE; 0ms
1 grant select, insert, update, delete on temp_results to logzhao; 2ms
2 SET ROLE logzhao; 2ms
3 CALL scpro('10001') 5ms permission denied for table students
```

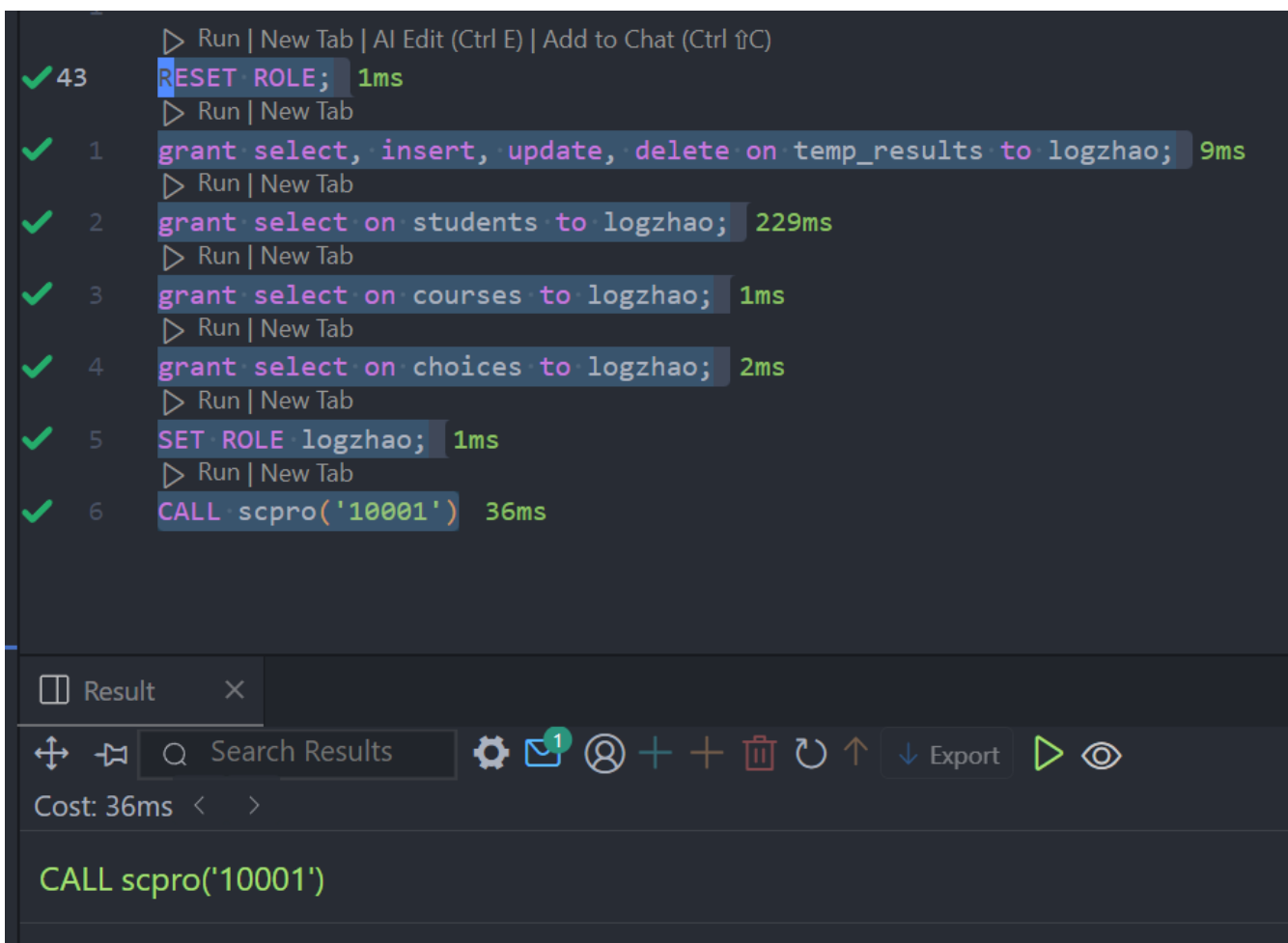
Result

Search Results

Cost: 5ms

permission denied for table students

全部赋予权限后，正常执行：



```
43 RESET ROLE; 1ms
1 grant select, insert, update, delete on temp_results to logzhao; 9ms
2 grant select on students to logzhao; 229ms
3 grant select on courses to logzhao; 1ms
4 grant select on choices to logzhao; 2ms
5 SET ROLE logzhao; 1ms
6 CALL scpro('10001') 36ms
```

Result

Search Results

Cost: 36ms

CALL scpro('10001')

补充内容：撤销赵老师查询某课程的选课情况的权限，再验证赵老师能否执行存储过程？

```
RESET ROLE;
REVOKE EXECUTE ON PROCEDURE scpro FROM logzhao;
SET ROLE logzhao;
CALL scpro('10001');
```

执行后，发现撤销权限后，赵老师依旧能够执行存储过程：


```
END LOOP;
END $$;
```

```

1  > Run | New Tab | AI Edit (Ctrl E) | Add to Chat (Ctrl ⌘C)
19 CREATE ROLE m_role;
   > Run | New Tab
1  GRANT INSERT, UPDATE, DELETE ON students TO m_role;
2
3  > Run | Copy
3  DO $$
4  BEGIN
5  FOR i IN 1..10 LOOP
6  EXECUTE format('CREATE USER counselor%s WITH PASSWORD 'password%s'', i, i);
7  EXECUTE format('GRANT m_role TO counselor%s', i);
8  END LOOP;
9  END $$;

```

Result

Search Results

Cost: 114ms

Execution completed in 114ms

- 验证插入操作权限

```
SET ROLE counselor1;

INSERT INTO STUDENTS (sid, sname, email, grade) VALUES ('S1001', 'New Student', 'new@example.com', 2023);

RESET ROLE;
```

```

1  > Run | New Tab
✓30 SET ROLE counselor1; 1ms
   > Run | New Tab
✓ 1 INSERT INTO STUDENTS (sid, sname, email, grade) VALUES ('S1001', 'New Student', 'new@example.com',

```

Result

Search Results

Cost: 3ms

INSERT INTO STUDENTS (sid, sname, email, grade) VALUES ('S1001', 'New Student', 'new@example.com', 2023)

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？

- 创建应用程序角色，并授予权限：

```
CREATE ROLE app_role;
GRANT INSERT ON students TO app_role;
```

```
2
  ▷ Run | New Tab
1 CREATE ROLE app_role;
  ▷ Run | New Tab
✓ 35 GRANT INSERT ON students TO app_role; 13ms
```

Result

Search Results

Execution completed in 13ms

- 验证插入操作权限:

```
SET ROLE app_role;
INSERT INTO STUDENTS (sid, sname, email, grade) VALUES ('S1002', 'App Student', 'app@example.com', 2023);
RESET ROLE;
```

```
▷ Run | New Tab | AI Edit (Ctrl E) | Add to Chat (Ctrl ⌘C)
✓ 34 SET ROLE app_role; 1ms
  ▷ Run | New Tab
✓ 1 INSERT INTO STUDENTS (sid, sname, email, grade) VALUES ('S1002', 'App Student', 'app@example.com',
  ▷ Run | New Tab
✓ 2 RESET ROLE; 1ms
3
```

Result

Search Results

Cost: 1ms

Execution completed in 1ms