

浙江大学



《数据库系统》 实验报告

| | |
|--------|-----------------------|
| 作业名称 : | SQL 安全性 |
| 姓 名 : | 王晓宇 |
| 学 号 : | 3220104364 |
| 电子邮箱 : | 3220104364@zju.edu.cn |
| 联系电话 : | 19550222634 |
| 授课教师 : | 孙建伶 |

2024 年 3 月 26 日

实验名称 SQL 安全性

1 实验目的

熟悉通过SQL进行安全性控制的方法。

2 实验环境

1. 操作系统: Windows 11 23H2
2. 数据库管理系统: MySQL 8.0.36
3. 工具: MySQL workbench 8.0

3 实验流程

3.1 新建数据库和用户

3.1.1 新建数据库

首先新建数据库 lab4,之后生成新表插入数据:

```
use lab4;

# 课程表
CREATE TABLE course(
  c_id VARCHAR(20),
  c_name VARCHAR(20) ,
  PRIMARY KEY(c_id)
);

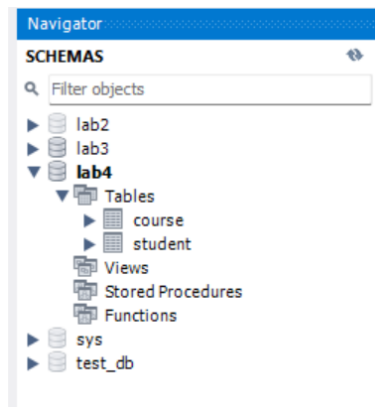
# 学生表
CREATE TABLE student(
  s_id VARCHAR(20),
  s_name VARCHAR(20),
  s_birth VARCHAR(20),
  s_sex VARCHAR(10) ,
  PRIMARY KEY(s_id),
  check (s_sex in ('男','女'))
);
```

插入学生表测试数据

```
insert into student values('01' , '赵雷' , '1990-01-01' , '男');
insert into student values('02' , '钱电' , '1990-12-21' , '男');
insert into student values('03' , '孙凤' , '1990-05-20' , '男');
insert into student values('04' , '李云' , '1990-08-06' , '男');
insert into student values('05' , '周梅' , '1991-12-01' , '女');
insert into student values('06' , '吴兰' , '1992-03-01' , '女');
insert into student values('07' , '郑竹' , '1989-07-01' , '女');
insert into student values('08' , '王菊' , '1990-01-20' , '女');
## insert into student values('08' , 'Ironman' , '1990-01-20' , '男');
```

课程表测试数据

```
insert into course values('01' , '语文' );
insert into course values('02' , '数学' );
insert into course values('03' , '英语' );
```



3.1.2 新建用户

root用户即超级管理员用户，拥有数据库的全部权限，而普通用户，由root创建，普通用户只拥有root所分配的权限。

新建普通用户的语法：

```
CREATE USER <user_name>@<host_name> identified BY <password>;
```

example

```
CREATE USER 'star0228'@'%' identified BY 'mypass'; -- 主机名为"%", 即对所有主机开放权限
```

```
CREATE USER 'testuser1'@'localhost' IDENTIFIED BY '123456'; -- 对本地主机开放，密码为123456
```

本次新建数据库 lab4 ,新建用户 testuser1 连接到本地主机,密码123456.(注意是在Root 权限下新建)

```
CREATE USER 'testuser1'@'localhost' IDENTIFIED BY '123456';
```

```
35
36 • CREATE USER 'testuser1'@'localhost' IDENTIFIED BY '123456';
37
15 12:31:40 Insert into course values(03 , 英语 )
16 20:03:28 CREATE USER 'testuser1'@'localhost' IDENTIFIED BY '123456'
```

Setup New Connection

Connection Name: connect1 Type a name for the connection

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: localhost Port: 3300 Name or IP address of the server host - and TCP/IP port.

Username: testuser1 Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

Configure Server Management... Test Connection Cancel OK

Setup New Connection

Connection Name: connect1 Type a name for the connection

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: localhost

Username: testuser1

Password: Store in Vault ...

Default Schema:

MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

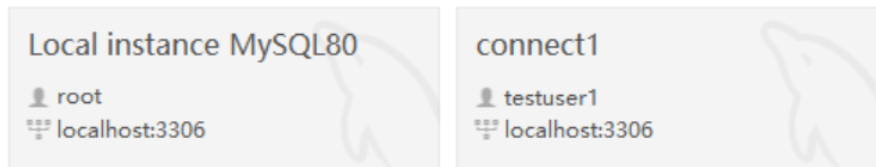
Host: localhost
Port: 3306
User: testuser1
SSL: enabled with TLS_AES_128_GCM_SHA256

A successful MySQL connection was made with the parameters defined for this connection.

OK

Configure Server Management... Test Connection Cancel OK

MySQL Connections ⊕ ↻

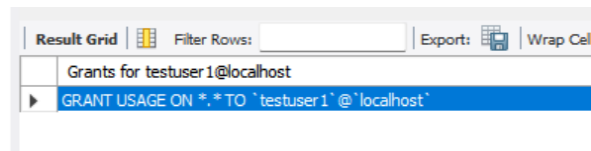


3.2 建立表

考察表的生成者拥有该表的哪些权限

现在我们查看一下testuser1@localhost拥有的权限：

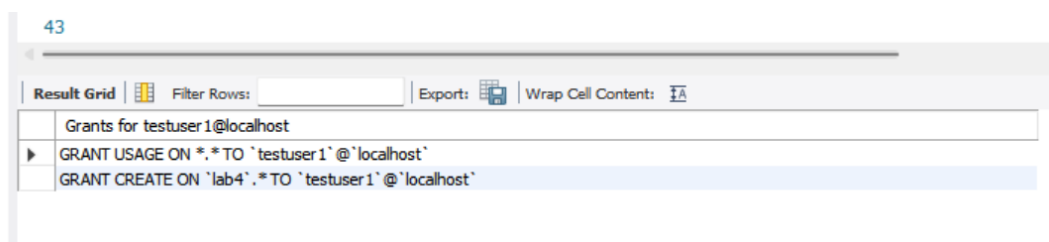
```
show grants for 'testuser1'@'localhost';
```



为了以普通用户身份建表，需要在root身份为testuser1授予在lab4上建表的权限：

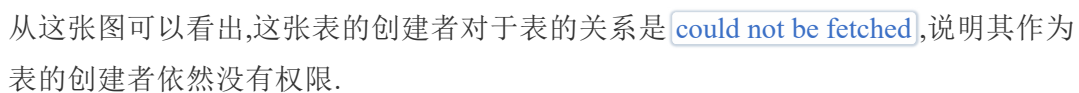
```
GRANT create ON lab4.* TO 'testuser1'@'localhost';
```

再次通过show命令查看得到的授权：



现在让此测试用户建立新表,并查看他对这个表的权限如何：

```
CREATE TABLE score(  
  s_id VARCHAR(20),  
  c_id VARCHAR(20),  
  s_score INT(3),  
  PRIMARY KEY(s_id,c_id),  
  check (s_score >0)  
);
```



考察grant和revoke命令相应的作用

现在 `testuser1` 没有任何权限,我们利用 `root` 给他授权在 `score` 表上的 `SELECT` 和 `INSERT` 权限.

查看一下哪些用户有哪些在表score上的权限:

```
39
40 • GRANT select ON lab4.score TO 'testuser1'@'localhost';
41 • GRANT insert ON lab4.score TO 'testuser1'@'localhost';
42 • select * from mysql.tables_priv where table_name='score';
43
```

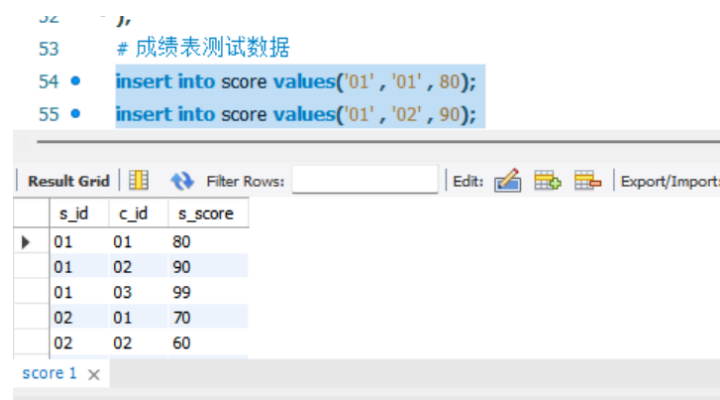
[illegible]

可以看到执行完上述三条指令后, `testuser1` 对 `score` 表有了 `insert`, `select` 两种权限, 现在尝试一下插入数据和查看指令:

成绩表测试数据

```
insert into score values('01' , '01' , 80);
insert into score values('01' , '02' , 90);
insert into score values('01' , '03' , 99);
insert into score values('02' , '01' , 70);
insert into score values('02' , '02' , 60);
insert into score values('02' , '03' , 80);
insert into score values('03' , '01' , 80);
insert into score values('03' , '02' , 80);
insert into score values('03' , '03' , 80);
insert into score values('04' , '01' , 50);
insert into score values('04' , '02' , 30);
insert into score values('04' , '03' , 20);
insert into score values('05' , '01' , 76);
insert into score values('05' , '02' , 87);
insert into score values('06' , '01' , 31);
insert into score values('06' , '03' , 34);
insert into score values('07' , '02' , 89);
insert into score values('07' , '03' , 98);
```

```
select * from score;
```



The screenshot shows a SQL IDE interface. At the top, there is a comment line: `# 成绩表测试数据`. Below it, two SQL commands are listed: `insert into score values('01' , '01' , 80);` and `insert into score values('01' , '02' , 90);`. Below the commands, there is a "Result Grid" section. It contains a table with three columns: `s_id`, `c_id`, and `s_score`. The table displays the first five rows of data inserted: (01, 01, 80), (01, 02, 90), (01, 03, 99), (02, 01, 70), and (02, 02, 60). The table is titled "score 1" with a close button.

| | s_id | c_id | s_score |
|---|------|------|---------|
| ▶ | 01 | 01 | 80 |
| | 01 | 02 | 90 |
| | 01 | 03 | 99 |
| | 02 | 01 | 70 |
| | 02 | 02 | 60 |

可以看到插入成功且显示出了数据.

3.3.2 revoke 命令

回收权限的语法:

```
REVOKE <priv_type> ON [object_type] FROM <user>
```

以root身份,把表score的select权限回收:

```
REVOKE select ON lab4.score FROM 'testuser1'@'localhost';
```

此后再回到root查询对表score拥有的权限。

```
GRANT create ON lab4.* TO 'testuser1'@'localhost';
```

```
38 • show grants for 'testuser1'@'localhost';
39
40 • GRANT select ON lab4.score TO 'testuser1'@'localhost';
41 • GRANT insert ON lab4.score TO 'testuser1'@'localhost';
42 • select * from mysql.tables_priv where table_name='score';
43
```

| | Host | Db | User | Table_name | Grantor | Timestamp | Table_priv | Column_priv |
|---|-----------|------|-----------|------------|----------------|---------------------|------------|-------------|
| ▶ | localhost | lab4 | testuser1 | score | root@localhost | 2024-03-26 20:57:56 | Insert | |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

可以看到testuser1的select权限被收回,此时在testuser1端去查询score表,发现拒绝访问。

```
35 20:53:46 18 row(s) returned select * from score LIMIT
36 20:59:43 Error Code: 1142. SELECT command denied to user 'testuser1'@'localhost' for table 'score' select * from score LIMIT
```

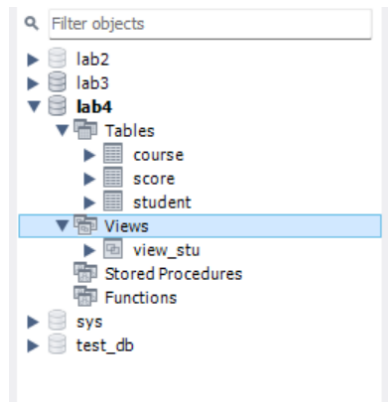
说明权限已被收回,指令有效。

3.4 建立视图,并把该视图的查询权限授予其他用户

考察通过视图进行权限控制的作用

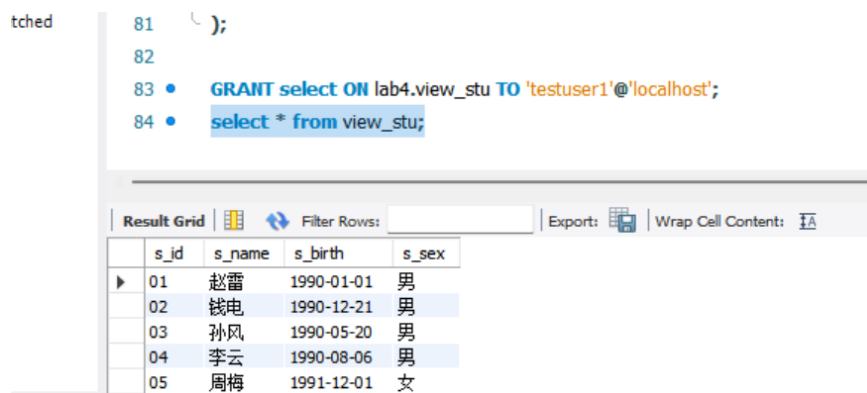
首先通过Root去生成一个视图:包含student的全部信息的视图。

```
create view view_stu as(
    select *
    from student
);
```

现在给 `testuser1` 在视图 `view_stu` 上的 `select` 权限,并选择 `testuser1` 去查看视图:

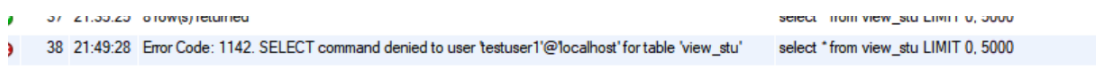
```
GRANT select ON lab4.view_stu TO 'testuser1'@'localhost';
select * from view_stu;
```



可以看到用户拥有了对视图的select权限,此时view的地位是和table接近的,所以指令也很接近.

此时再收回权限,预期产生报错:

```
REVOKE select ON lab4.view_stu FROM 'testuser1'@'localhost';
select * from view_stu;
```



确实.

4 遇到的问题及解决方法

- 新创建用户拥有create权限之后并不能创建带有外键指向没权限的表

这个错误倒也是正常的,因为一个只拥有创建表权限的用户在创建表之后如果要指明外键的话,必须要有对被指向表的访问权限,要不然指不到,解决方案是root给这个用户表的访问权限。

5 总结

本次实验是对sql权限的学习,因为理论课上讲过SQL注入,所以SQL的安全性需要被重视,通过 `revoke`, `grant` 的权限调用,可以将修改表的权限限制,可能会对防御SQL注入有一定效果。

另外,这节课依然提到了视图的概念,视图本身和权限等概念有相似之处,都是为了保护表中数据做的功能划分,更加重了我们对开发系统安全性的重视。

下周开始做图书管理系统,预祝成功。