实验4 SQL安全性

熊子宇 3200105278

1 实验目的

熟悉通过 SQL 进行安全性控制的方法。

2 实验平台

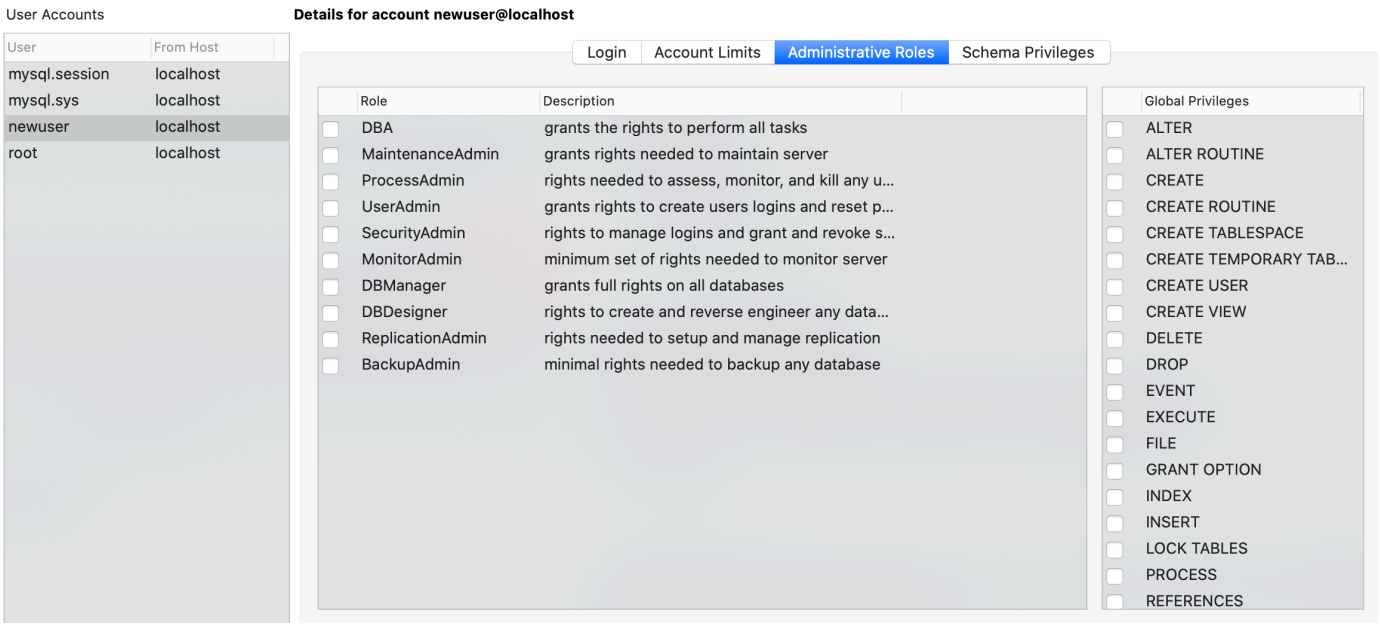
1. 操作系统: MacOS
2. 数据库管理系统: MySQL 5.7.28
3. 数据库图形界面: MySQL Workbench 6.3.10

3 实验内容和要求

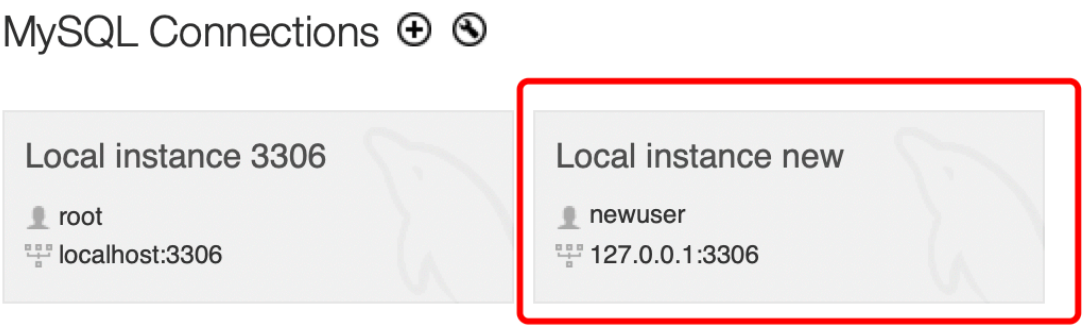
3.1 建立表，考察表的生成者拥有该表的哪些权限

在3.1节，我们先使用图形化的方法来创建用户（作用等价于`create user`命令）

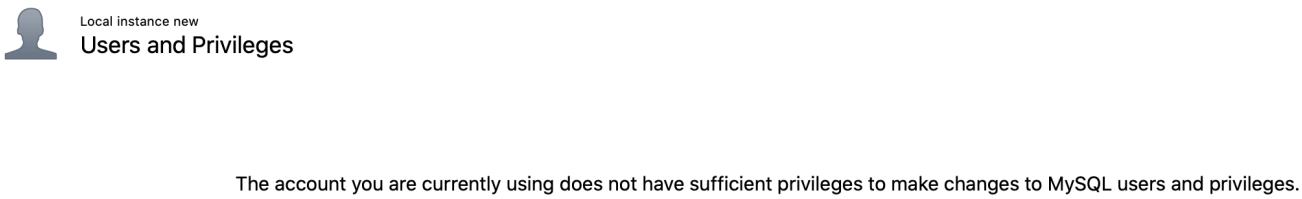
在MySQL Workbench Mangement中的Users and Privileges板块新建一个User `newuser`，不赋予任何权限。



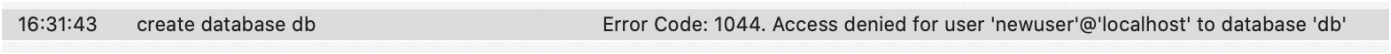
在主页新建`newuser`的Connection，如下



在`Local instance new`连接中，点击Users and Privileges，会告知该用户无任何权限。



尝试`create database db;`语句后，会报错无权限创建dabatase。



在root用户中执行`grant create on Library.* to 'newuser'@'localhost';`语句，授予newuser在Library数据库中创建表的权限。

使用`show grants for 'newuser'@'localhost'`语句查询newuser的权限：

Grants for newuser@localhost
GRANT USAGE ON *.* TO 'newuser'@'localhost'
GRANT CREATE ON `library`.* TO 'newuser'@'localhost'

使用`create table test(test int);`插入表test，插入成功：

Tables_in_library
book
book_stock
borrow
borrower_book
card
test

如果尝试修改表`insert into book values();`，则会报错：`Error Code: 1142. INSERT command denied to user 'newuser'@'localhost' for table 'book'`

因此表的生成者其实只有`create table`的权限。

3.2 使用 SQL 的 grant 和 revoke 命令对其他用户进行授权和权力回收，考察相应的作用

在3.1节中，我们已经尝试了使用`grant`命令让root给newuser授予创建表的权限。现在使用`revoke`命令将该权限收回。

```
revoke create on Library.* from 'newuser'@'localhost';
flush privileges;
```

再次查询newuser的权限，发现权限已更新：

Grants for newuser@localhost
GRANT USAGE ON *.* TO 'newuser'@'localhost'

在newuser账户下访问数据库Library，会报错：

	Time	Action	Response
1	17:02:30	use Library	Error Code: 1044. Access denied for user 'newuser'@'localhost' to database 'library'

实际上，MySQL数据库使用**六张表**来控制操作权限。操作权限一共有六个级别。

- user表

user表列出可以连接服务器的用户及其口令，并且它指定他们有哪种全局（超级用户）权限。在user表启用的任何权限均是全局权限，并适用于所有数据库。

- db表

db表列出数据库，而用户有权限访问它们。在这里指定的权限适用于一个数据库中的所有表。

- tables_priv表

tables_priv表指定表级权限，在这里指定的一个权限适用于一个表的所有列。

- columns_priv表

columns_priv表指定列级权限。这里指定的权限适用于一个表的特定列。

- proce_priv表

columns_priv表指定存储过程权限。这里代表允许使用某个存储过程的权限。

- proxies_priv表

利用 MySQL proxies_priv（模拟角色）实现类似用户组管理。角色(Role)可以用来批量管理用户，同一个角色下的用户，拥有相同的权限。

授权格式：

```
grant [权限1,权限2,权限3] on *.* to user@'host' identified by 'password'
```

例如，给"newuser" 用户管理员权限，并且允许该用户继续给别的用户赋权限：

```
grant all privileges on *.* to 'newuser'@'192.168.1.%' with grant option;
```

- **all privileges**：表示将所有权限授予给用户。也可以指定具体权限：**create**、**drop**、**select**、**insert**、**delete**、**update**
- **on**：表示这些权限对哪些数据库和表生效，格式：数据库名.表名，这里写“*”表示所有数据库，所有表。如果我要指定将权限应用到test库的user表中，可以这么写：test.user
- **to**：将权限授予哪个用户。格式：”用户名”@”登录IP或域名”。%表示没有限制，在任何主机都可以登录。
- **with grant option**：通过在grant语句的最后使用该子句，就允许被授权的用户把得到的权限继续授给其它用户

注：使用GRANT添加权限，权限会自动叠加，不会覆盖之前授予的权限，比如你先给用户添加一个SELECT权限，后来又给用户添加了一个UPDATE权限，那么该用户就同时拥有了SELECT和UPDATE权限。

3.3 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

在之前的实验中，我们已经在Library数据库中创建了`book_stock`视图。现在将该视图的查询权限(`select`)授权给newuser：

```
grant select on Library.book_stock to 'newuser'@'localhost';
```

视图本质上是虚表，在某种程度上可以当作表来对待。因此授权视图的查询权限和授权表的权限十分类似。

切换至用户newuser，依次执行三条语句：

```
use Library;
show tables;
select * from book_stock;
```

成功查询到该视图内容：

bno	stock
0000000001	4
0000000009	5

而如果尝试其他操作，如删除视图，则没有该权限：

	Time	Action	Response
1	18:11:20	use Library	0 row(s) affected
2	18:11:27	show tables	1 row(s) returned
3	18:11:36	select * from book_stock LIMIT 0, 1000	2 row(s) returned
4	18:12:47	drop view book_stock	Error Code: 1142. DROP command denied to user 'newuser'@'localhost' for table 'book_stock'

4 实验心得

1. MySQL中的权限级别和命令语句

MySQL中的权限和语句都与课上略有不同，需要在网上学习相关资料才能正确使用。

2. 权限更新可能有延迟

在3.2节中，当使用`revoke`命令收回newuser的`create table`权限，并且`flush privileges;`以后，我发现仍然可以在newuser用户下继续成功创建表。只有当把connection断开重连时，实际上权限才会变更。这可能是MySQL的一个漏洞。

3. 安全性是由数据库控制还是由外部软件控制?

做完本实验，我想到图书管理系统中至少要分管理员和普通用户两个角色。有至少两种方式控制权限。一种是像本实验一样，在数据库内部定义多个用户并由DBA授予相应权限。不同种类用户登录后以不同身份连接至数据库，对于数据库拥有不同的访问权限。另一种方式是在ODBC中借由C++代码来约束。

我初步认为使用数据库内在的安全性控制要好于外部软件。因为数据库内部已经集成好这个功能，较为完备且简单。如果在外部软件中定义可能会造成疏漏。