实验4 SQL安全性

姓名：段皞一

学号：3190105359

专业：计算机科学与技术

一 实验目的

- 熟悉通过SQL进行数据完整性控制的方法。

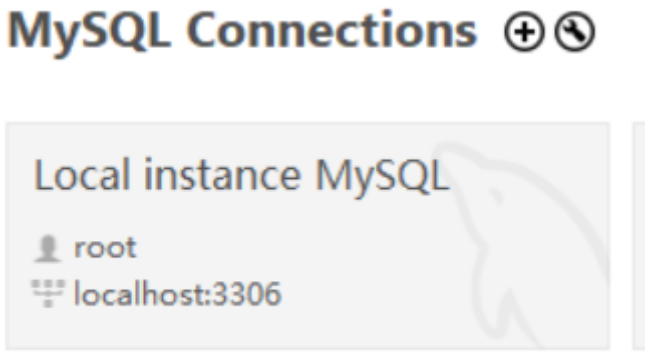
二 实验平台

- 数据库管理系统：SQL Server 或MySQL

三 实验内容和要求

1.**建立表，考察表的生成者拥有该表的哪些权限**

- **用*root*身份登录*MySQL*：**



*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> idetified BY <password>;
```

例如：

```
CREATE USER 'mike'@'%'identified BY '123456';
```

主机名为'%'，即对所有主机开放权限。

而另外一个例子：

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

是常规的普通用户创建方法。

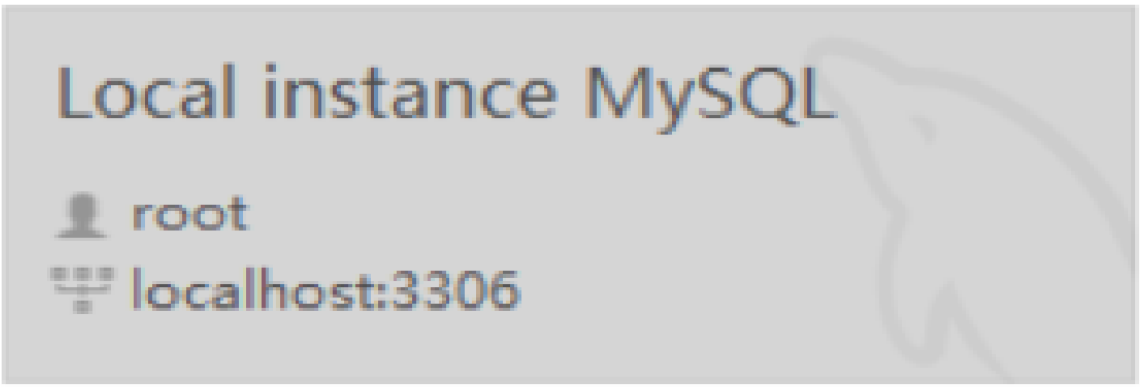
- **新建查询执行这条语句，新建用户：**

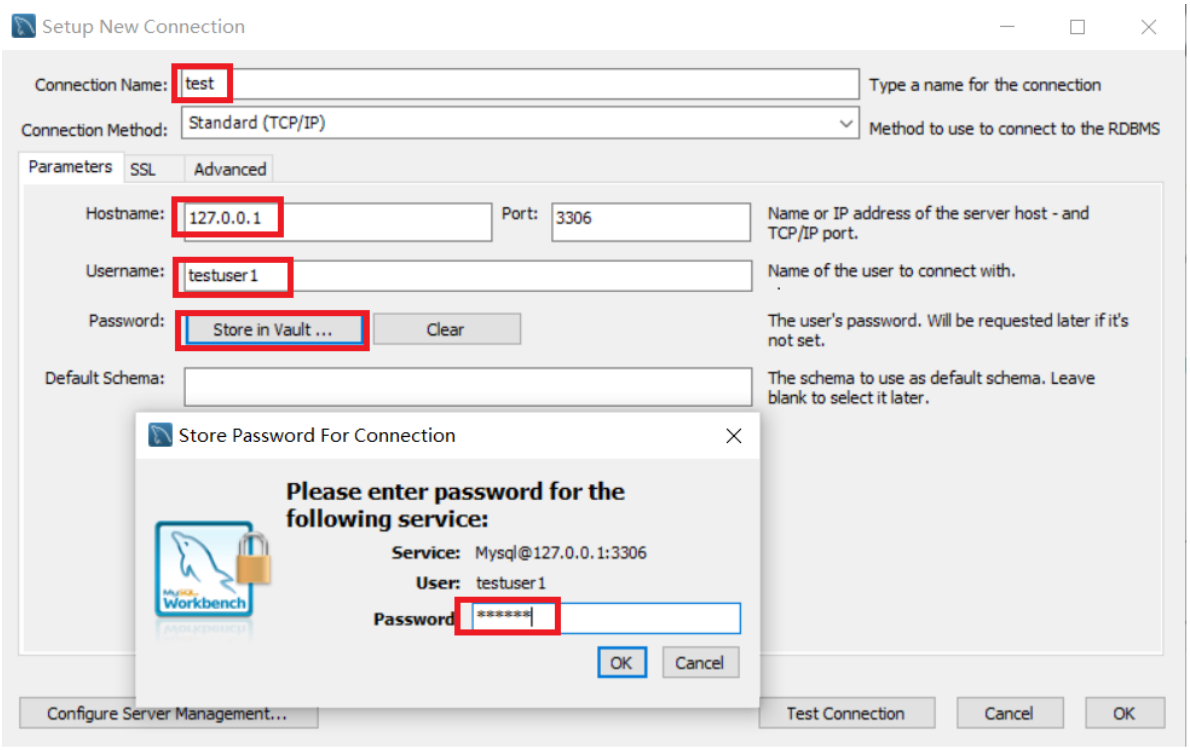
```
1    CREATE USER 'testuser1'@'localhost' identified BY '123456';
```

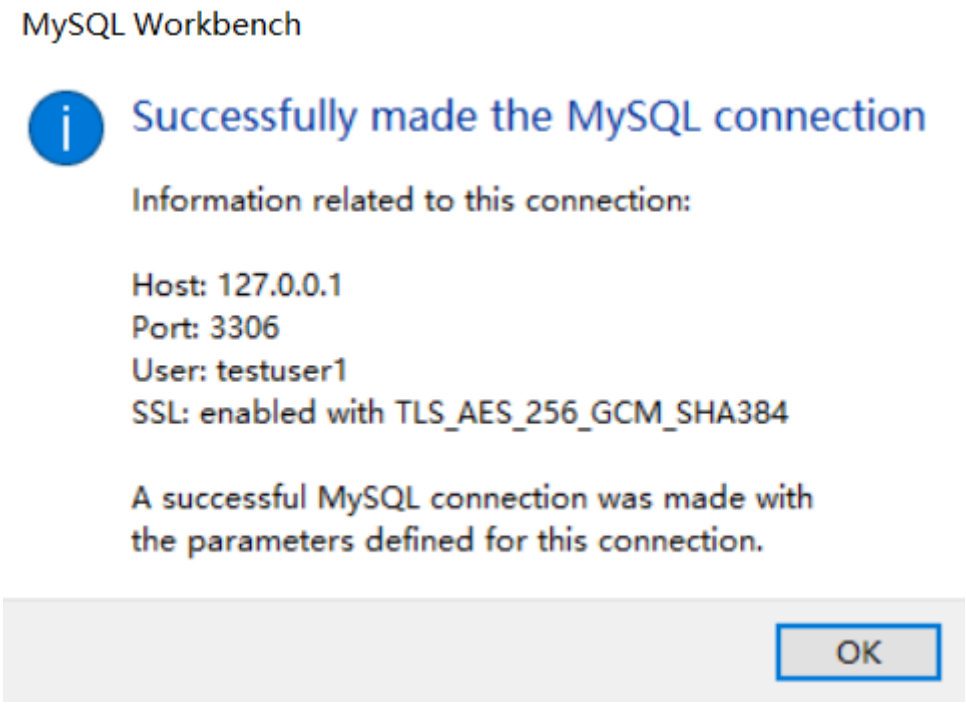
可以看到任务栏中出现新建普通用户*'testuser1'*成功的消息：

```
2  12:57:55  CREATE USER 'testuser1'@'localhost' identified BY '123456'  0 row(s) affected  0.032 sec
```

- **现在用这个新的用户身份登录：**







- **用这个新建的用户不能直接访问root用户建立的表：**

我们可以在新建用户的目录下新建查询，输入：

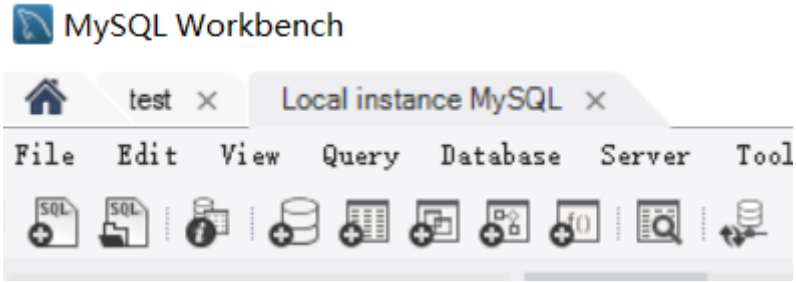
```
select * from sys.company;
```

会报出如下错误：

18:58:20 select * from newtest.company LIMIT 0, 1000 Error Code: 1142. SELECT command denied to user 'testuser1'@'localhost' for table 'account' 0.000 sec

- **对新建用户进行授权：**

首先回到*root*身份，为该用户进行访问表的授权。



授权语句的语法：

```
grant priv_type on [object_type] to user;
```

*MySQL*中由多种权限，授予的权限也可以分为多个层级。

现在为我们新建的用户授予数据库*newtest*下的表*account*的*select*权限：

```
grant select on newtest.account to 'testuser1'@'localhost';
```

现在再回到新建的用户进行数据库的表查询，不再报错：

	name	city
▶	dhy	Kunming

可见授权成功。

- **查看用户的权限：**

为了查看用户testuser1@localhost的权限，应该输入如下的代码：

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

	Grants for testuser1@localhost
	GRANT USAGE ON *.* TO `testuser1`@`localh...
▶	GRANT SELECT ON `sys`.`company` TO `test...

接下来在*root*下查看一下哪些用户有在表*company*上的权限：

```
select * from mysql.tables_priv where table_name = 'company';
```

得到以下的结果：

	Host	Db	User	Table_name	Grantor	Timestamp	Table_priv	Column_priv
	localhost	sys	testuser1	company	root@localhost	0000-00-00 00:00:00	Select	
▶*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- **建立表，考察用户的权限：**

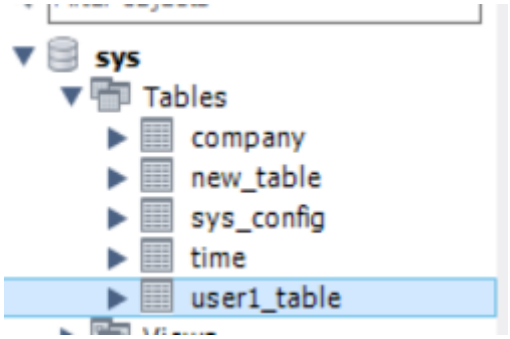
为了以普通用户身份建立表，首先需要在*root*身份为*testuser1*授权在某个数据库上建立表的权限，例如：

```
grant create on sys.* to 'testuser1'@'localhost';
```

现在以*testuser1*的身份建立表*user1_table*：

```
use sys;
create table user1_table
(
    id char(10),
    name char(15),
    primary key (id)
);
```

在*root*身份刷新数据库，看到表*user1_table*确实建立成功。

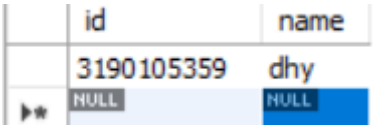


此时，可以观察到，*testuser1*作为表的创建者对*user1_table*没有其他的权限。在*testuser1*下执行以下的对表*user1_table*进行操作的语句，均无法成功：

```
select * from sys.user1_table;

insert into sys.user1_table
values ('3190105359', 'dhy');
```

而*root*则拥有所有的权限。同样在*root*下执行上述的语句，是可以成功的。



和*SQL server*的权限设计进行比较，发现*MySQL*并没有像*SQL server*划分*db_owner*、*db_datareader*、*db_datawriter*等粗粒度的权限模式。我们可以使用*GRANT*命令来对*user1_table*授予其他相关的权限（*UPDATE*、*INSERT*、*SELECT*）。

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**

- ***grant*授予权限**

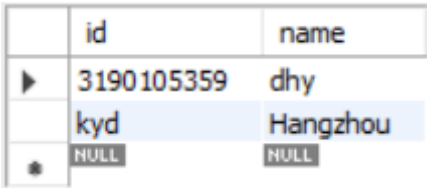
在*root*下授予用户*testuser1*查询和插入的权限：

```
grant select on sys.* to 'testuser1'@'localhost';
grant insert on sys.* to 'testuser1'@'localhost';
```

之后就可以在*testuser1*下进行字段的插入了：

```
insert into sys.user1_table
values ('kyd', 'Hangzhou');
```

重新回到*root*，刷新后可以看见表*user1_table*下新增了刚刚在*testuser1*下加入的字段。授权成功。



- ***revoke*收回权限**

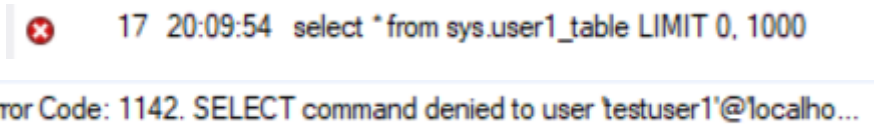
如果我们想收回权限，语法如下所示：

```
revoke priv_type on [object_type] from user;
```

现在回到*root*身份，把表*user1_table*的*select*权限回收：

```
revoke select on sys.* from 'testuser1'@'localhost';
```

此后再次回到*testuser1*下查询表*user1_table*，发现访问被拒绝了。



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**

- **新建视图**

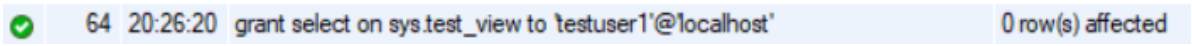
在*root*下新建查询输入，新建视图的*SQL*语句：

```
create view test_view(our_name) as
(
select name from
user1_table
);
```

- **授权将视图查询权限**

现在把该视图的查询权限授予*testuser1*：

```
grant select on sys.test_view to 'testuser1'@'localhost';
```

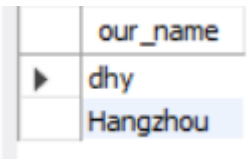


授权成功后，回到*testuser1*用户下。用该用户身份查询视图*test_view*：

```
use sys;

select * from test_view;
```

可以看到如下结果，可见授权成功了。



- **通过视图进行权限控制的作用**

在数据库的管理过程中，有些时候，我们对用户进行授权时，需要对用户的读取权限进行设置，比如：某用户只能获取*user*表的*name*和*age*数据，不能获取*sex*数据。

这种情况下，可以创建视图限制用户的权限：

```
create view other as select a.name, a.age from user as a;
```

这样，用户就不能够访问到*user*表中的*sex*数据了。