

- 实验目的：
- 数据库的管理系统：MySQL
- 实验内容和要求：
- 实验过程：
 - 1. 建立表，考察表的生成者拥有该表的哪些权限。
 - 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。
 - 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。
- 小结

实验4：SQL安全性

实验目的：

1. 熟悉通过SQL进行数据完整性控制的方法。

数据库的管理系统：MySQL

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。
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。
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。

实验过程：

1. 建立表，考察表的生成者拥有该表的哪些权限。
 - 这里我们试图模拟成绩数据库的建立，以观察权限的运作方式。

- 首先，我们利用如下命令建立一个数据库及表，然后查询root（即表的建立者）对

```
create database examming;
```

```
use examming;
```

```
create table realscore(  
    stuid int,  
    stuname varchar(40),  
    stuscore int  
);
```

```
use examming;
```

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

数据库的权限：

- 效果如下：

| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
|---------------------------|---|---------|--------------------|
| Grants for root@localhost | | | |
| ▶ | GRANT SELECT, INSERT, UPDATE, DELETE, CR... | | |
| | GRANT ALLOW_NONEXISTENT_DEFINER, APPLI... | | |
| | GRANT PROXY ON ``@`` TO `root`@`localho... | | |

可见表的生成者基本具备一切对数据库的权限。

- 我们再去探究下一级的user建表时对表的权限：

```
create user 'student1'@'localhost' identified by '654321';
```

- 连接这个user:

Setup New Connection

Connection Name: Type a name for the connection

Connection Method: Method to use to connect to the RDBMS

Parameters SSL Advanced

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

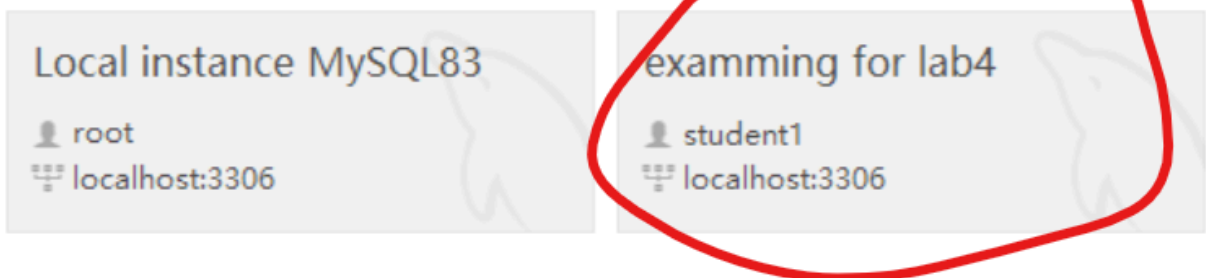
Username: Name of the user to connect with.

Password: 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.

- 最终连接成功效果如下：

MySQL Connections + ⓘ



在未授权之前，以正常方式访问examming库都是不被允许的（哪怕是use examming）：

14 14:48:39 use examming Error Code: 1044. Access denied for user 'student1'@'localhost' to dat...

- 于是我们需要下放权限：

```
grant create on examming.* to 'student1'@'localhost';
```

（意为：令student1在examming下属表中能够使用create语句）

- 我们再次尝试建表发现成功了：

| | | | | |
|---|---|----------|--|-------------------|
| ✓ | 6 | 14:25:57 | use examming | 0 row(s) affected |
| ✓ | 7 | 14:25:57 | create table fakescore(stuid int, stuname varchar(40), stuscore int) | 0 row(s) affected |

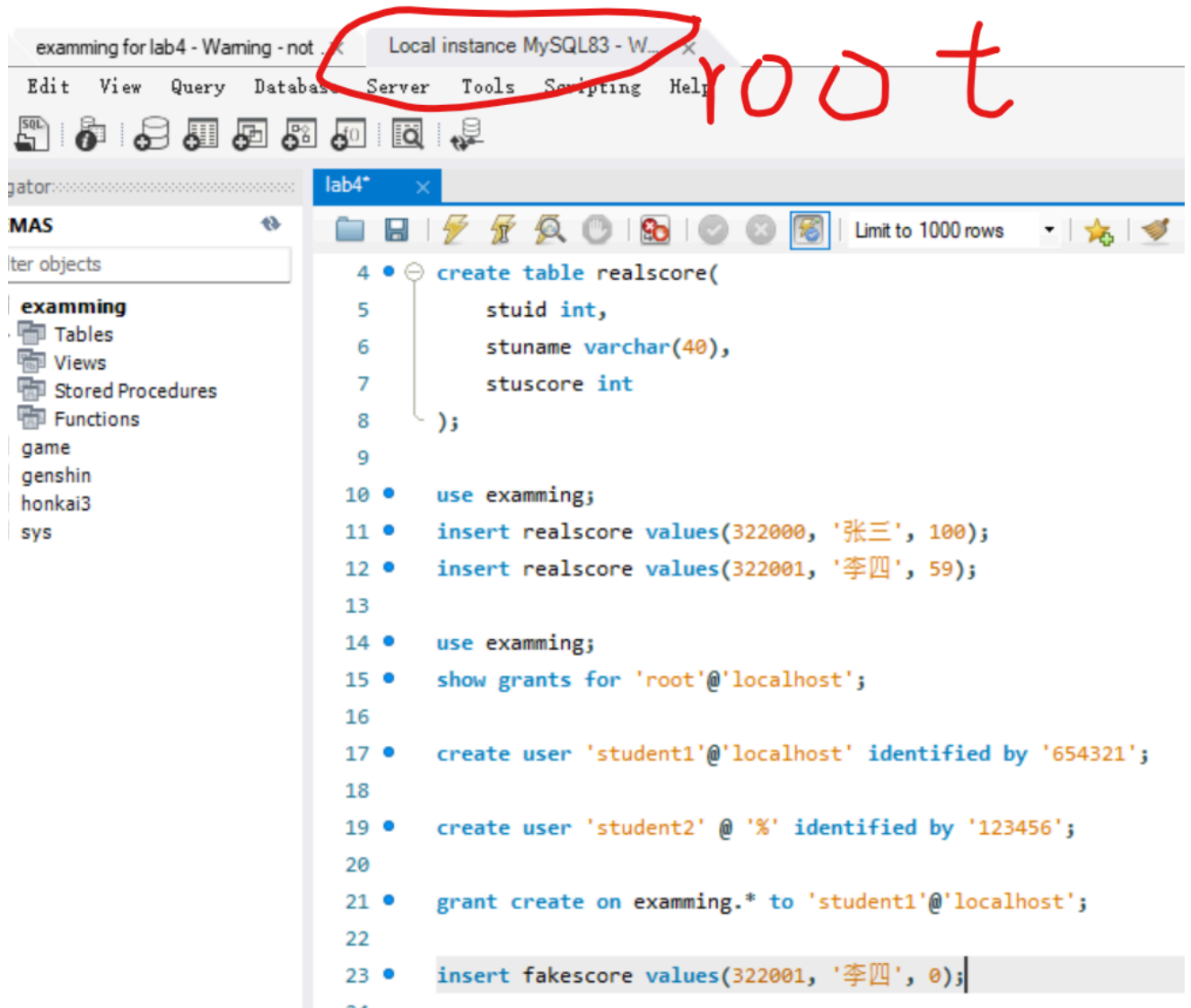
- 假设这个人是李四，他喜出望外想要改成绩：

```
insert fakescore values(322001, '李四', 100);
```

- 结果发现还是不行，因为insert权限仍然没有下放：

```
8 14:27:50 insert fakescore values(322001, '李四', 100) Error Code: 1142. INSERT command denied to user 'student1'@'local...
```

- 但root用户，可以顺利地insert李四创建的表，并把他的成绩改成0分：



- 显然成功了：

| | | | | |
|---|---|----------|--|-------------------|
| ✓ | 8 | 14:29:16 | insert fakescore values(322001, '李四', 0) | 1 row(s) affected |
|---|---|----------|--|-------------------|

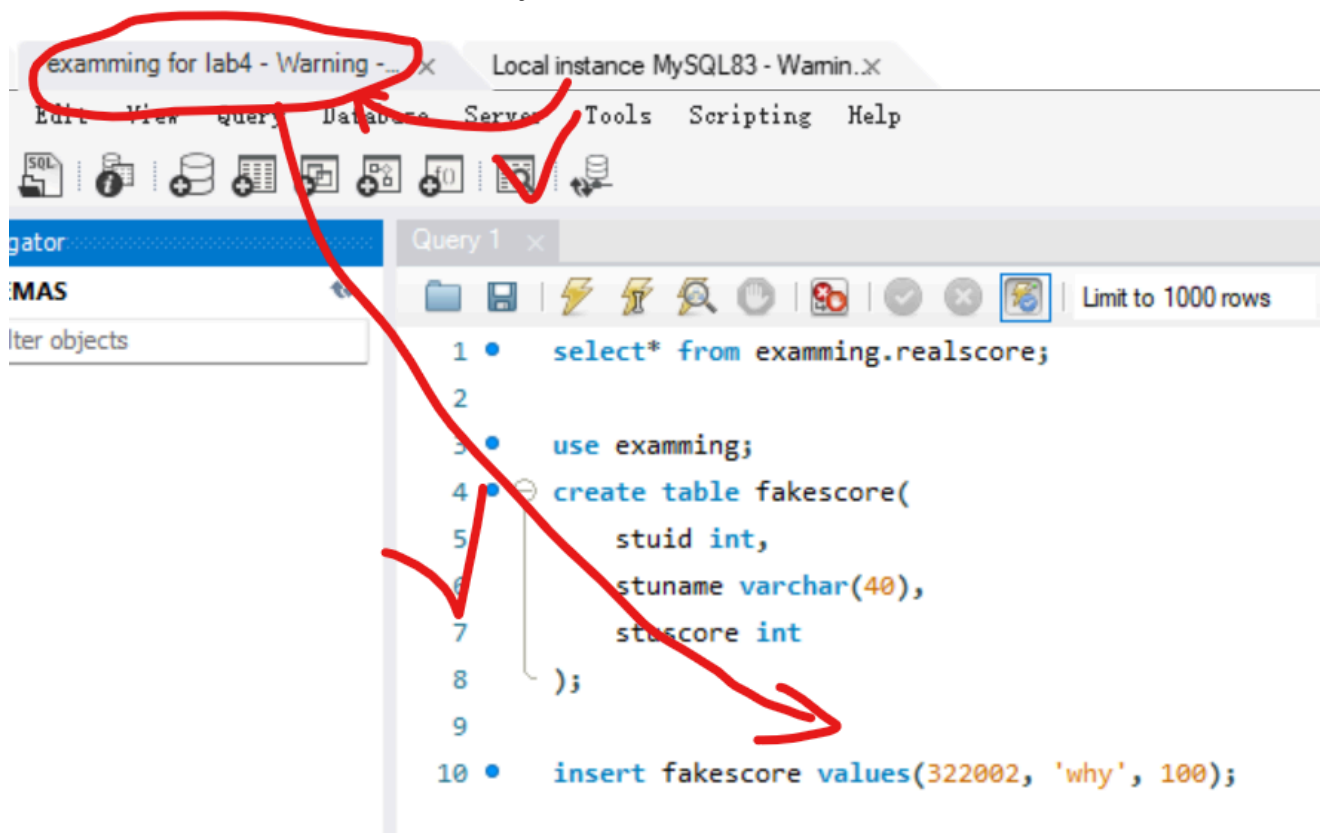
2. 使用SQL的grant和revoke命令对其他用户进行授权和权力回收，考察相应的作用。

事实上上面那步已经操作过了这部分的内容，但这里还是更进一步的做一些操作：

- 我们给student1下放insert的权限：

```
grant insert on fakescore to 'student1'@'localhost';
```

- 接下来，在student1视角下给why同学insert100分：



| Result Grid | | | |
|-------------|--------|---------|----------|
| | stuid | stuname | stuscore |
| ▶ | 322001 | 李四 | 0 |
| | 322002 | why | 100 |

- 显然这成功了：
- 我们再尝试收回insert的权限：

```
revoke insert on fakescore from 'student1'@'localhost';
```

- 再在student1视角下给赵五同学insert1000分：

```
insert fakescore values(322003, '赵五', 1000);
```

- 发现报错了，说明权限确实被收回了：

11 14:37:46 insert fakescore values(322003, '赵五', 1000)

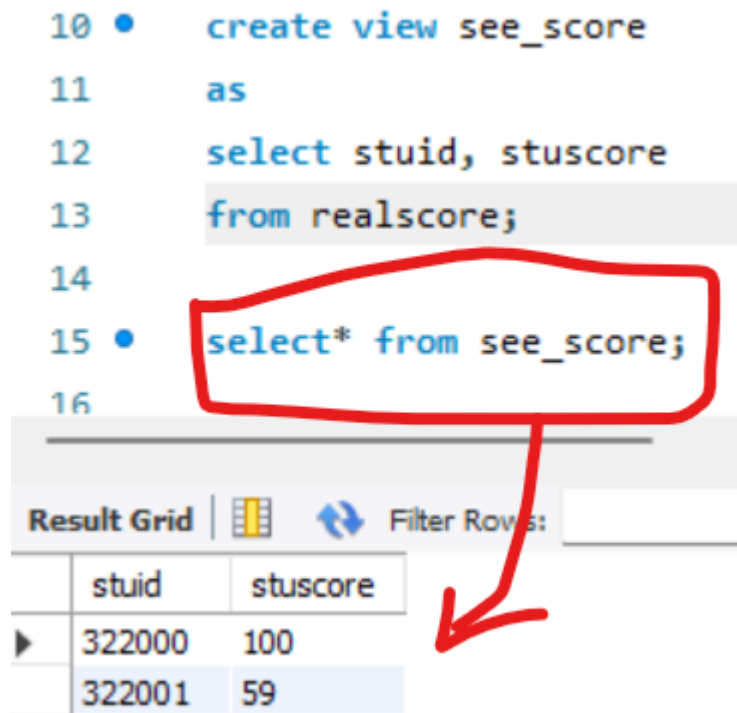
Error Code: 1142. INSERT command denied to user 'student1'@'localhost' for table 'fakescore'



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。

（假设成绩发放的日子到了，教务网可以把成绩视图公布给同学们）

- root先建立一个realscore的成绩视图:

```
10 • create view see_score
11 as
12 select stuid, stuscore
13 from realscore;
14
15 • select* from see_score;
16
```



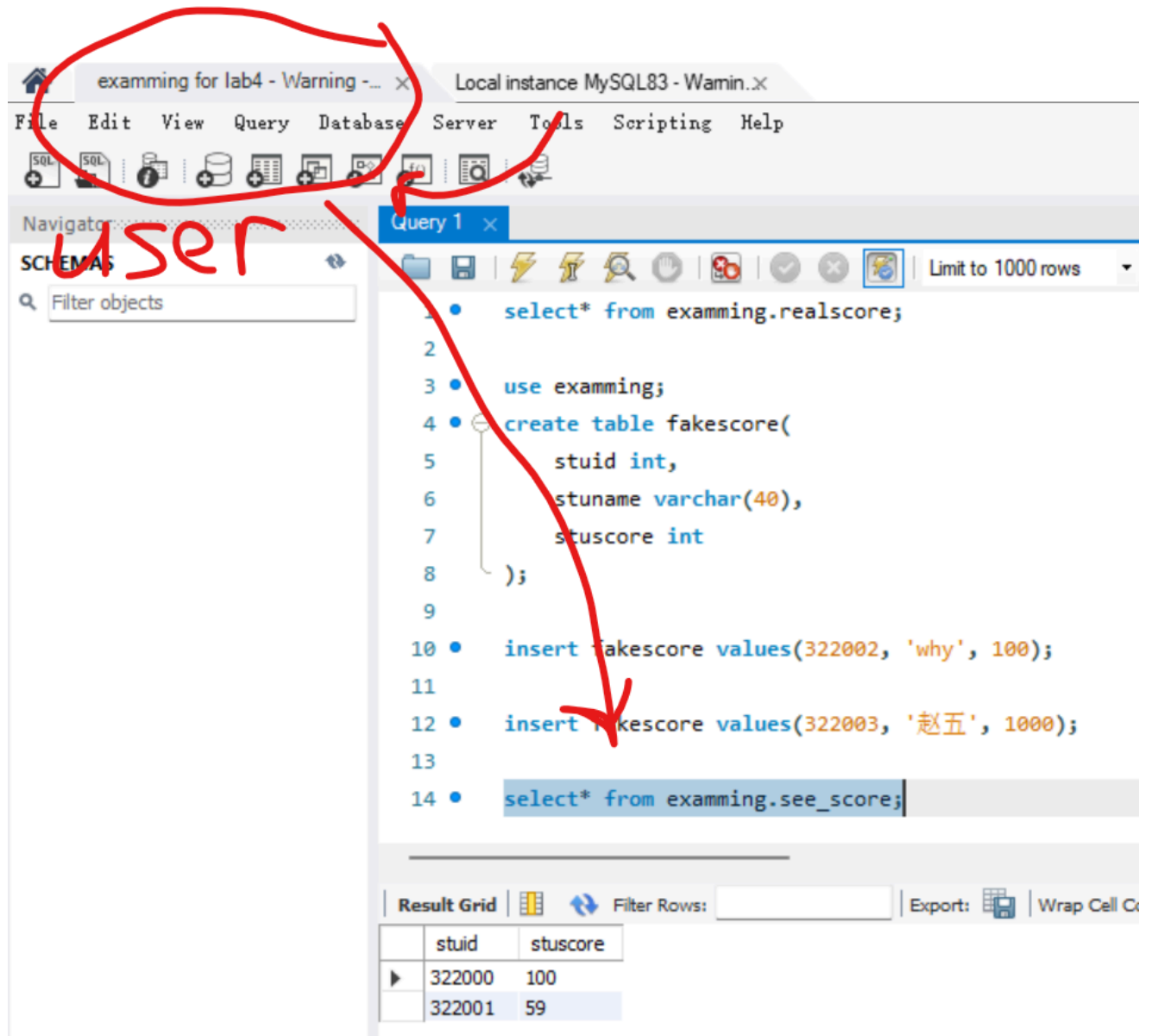
Result Grid |   Filter Rows:

| | stuid | stuscore |
|---|--------|----------|
| ▶ | 322000 | 100 |
| | 322001 | 59 |

- 接下来, root把视图的select权限给到student1:

```
grant select on examming.see_score to 'student1'@'localhost';
```

- 再在student1视角下调用select，结果发现自己挂科的事实没有改变（悲：



小结

经过这个实验我们不难发现，整个sql系统的权限管理是系统化且严格化的，几乎每一个操作的权限都被独立封装，这也让数据管理者能够更好地保护数据的安全。