

哈尔滨工业大学（深圳）

# 数据库实验指导书

实验二 高级 SQL 语言实验

2022 秋

## 目录

1	实验目的 .....	3
2	实验环境 .....	3
3	实验步骤 .....	3
3.1	观察并回答问题 .....	3
3.1.1	关于视图 .....	4
3.1.2	关于触发器 .....	4
3.1.3	关于约束 .....	5
3.2	创建新用户并分配权限 .....	5
3.3	设计并实现 .....	9
3.4	思考题 .....	10
附录 1	GRANT 命令 .....	11

# 1 实验目的

- 1、理解视图、触发器、约束的基本概念，掌握它们的用法；
- 2、能结合实例设计合理的视图、触发器和约束；
- 3、结合实验加深对数据库完整性和安全性的理解。

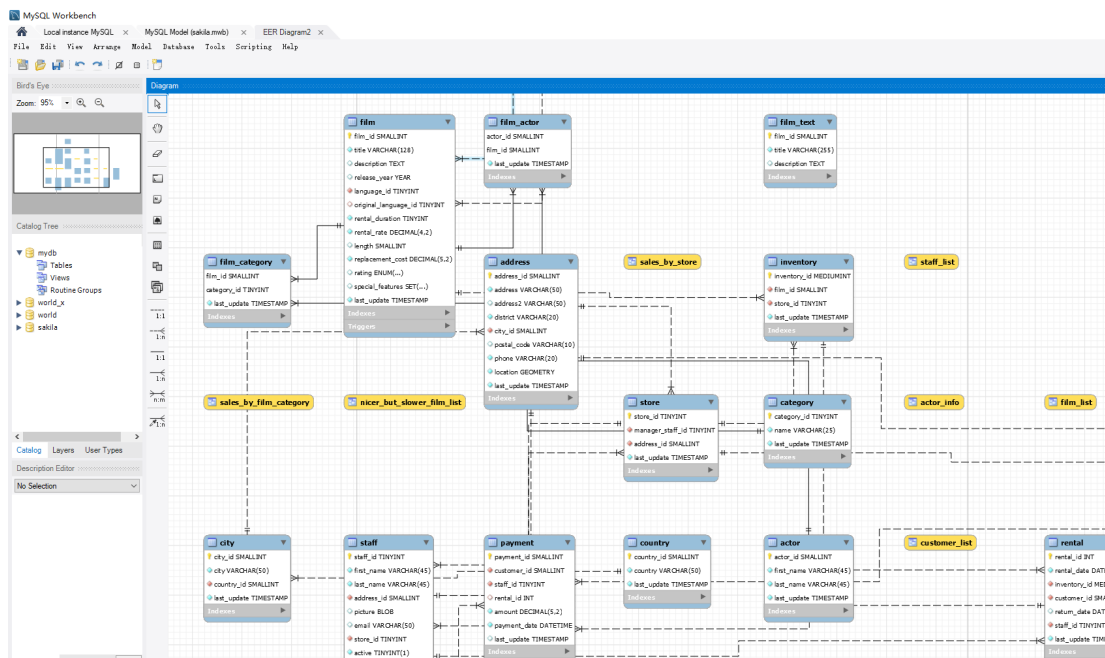
## 2 实验环境

Windows 10 操作系统、MySQL8.0。

## 3 实验步骤

### 3.1 观察并回答问题

- 1、打开官方文档：  
<https://dev.mysql.com/doc/sakila/en/sakila-installation.html>
- 2、用 MySQL Workbench 打开 sakila.mwb 文件：



注意：如果打开时报错 mysql Error unserializing GRT data string too long, 重启 MySQL Workbench 再打开就可以了。

以下问题的答案均写到实验报告中。

### 3.1.1关于视图

#### 1、回答问题:

- (1) sakila.mwb 模型图中共有几个 view?
- (2) 分析以下 3 个视图, 回答以下问题:

视图名	关联表	作用
actor_info		
film_list		
sales_by_store		

- (3) 分别执行以下 2 句 SQL 语句:

```
update staff_list set `zip code` = '518055' where ID = '1';
```

```
update film_list set price = 1.99 where FID = '1';
```

截图执行结果, 并分析一下视图在什么情况下可以进行 update 操作, 什么情况下不能?

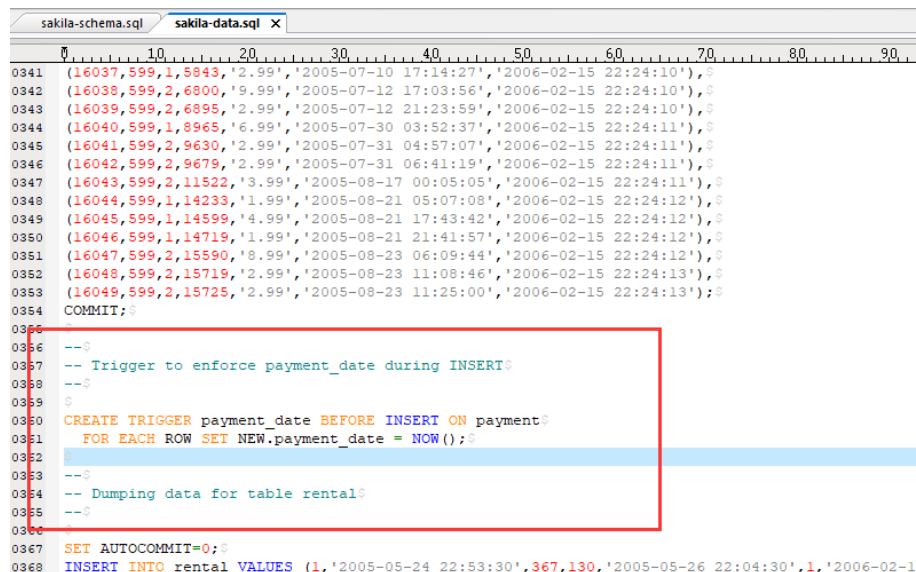
- (4) 执行以下命令查询 sakila 数据库中的视图是否可更新:

```
SELECT table_name, is_updatable FROM information_schema.views  
WHERE table_schema = 'sakila';
```

### 3.1.2关于触发器

#### 1、回答问题:

- (1) 触发器 customer\_create\_date 建在哪个表上? 这个触发器实现什么功能?
- (2) 在这个表上新增一条数据, 验证一下触发器是否生效。(截图语句和执行结果)
- (3) 我们可以看到 sakila-schema.sql 里的语句是用于创建数据库的结构, 包括表、视图、触发器等, 而 sakila-data.sql 主要是用于往表写入数据。但 sakila-data.sql 里有这样一个建立触发器的语句:



```
sakila-schema.sql sakila-data.sql X  
0341 (16037,599,1,5843,'2.99','2005-07-10 17:14:27','2006-02-15 22:24:10'),  
0342 (16038,599,2,6800,'9.99','2005-07-12 17:03:56','2006-02-15 22:24:10'),  
0343 (16039,599,2,6895,'2.99','2005-07-12 21:23:59','2006-02-15 22:24:10'),  
0344 (16040,599,1,8965,'6.99','2005-07-30 03:52:37','2006-02-15 22:24:11'),  
0345 (16041,599,2,9630,'2.99','2005-07-31 04:57:07','2006-02-15 22:24:11'),  
0346 (16042,599,2,9679,'2.99','2005-07-31 06:41:19','2006-02-15 22:24:11'),  
0347 (16043,599,2,11522,'3.99','2005-08-17 00:05:05','2006-02-15 22:24:11'),  
0348 (16044,599,1,14233,'1.99','2005-08-21 05:07:08','2006-02-15 22:24:12'),  
0349 (16045,599,1,14599,'4.99','2005-08-21 17:43:42','2006-02-15 22:24:12'),  
0350 (16046,599,1,14719,'1.99','2005-08-21 21:41:57','2006-02-15 22:24:12'),  
0351 (16047,599,2,15590,'8.99','2005-08-23 06:09:44','2006-02-15 22:24:12'),  
0352 (16048,599,2,15719,'2.99','2005-08-23 11:08:46','2006-02-15 22:24:13'),  
0353 (16049,599,2,15725,'2.99','2005-08-23 11:25:00','2006-02-15 22:24:13');  
0354 COMMIT;  
0355  
0356 --  
0357 -- Trigger to enforce payment_date during INSERT  
0358 --  
0359 $  
0360 CREATE TRIGGER payment_date BEFORE INSERT ON payment  
0361 FOR EACH ROW SET NEW.payment_date = NOW();  
0362  
0363 --  
0364 -- Dumping data for table rental  
0365 --  
0366 $  
0367 SET AUTOCOMMIT=0;  
0368 INSERT INTO rental VALUES (1,'2005-05-24 22:53:30',367,130,'2005-05-26 22:04:30',1,'2006-02-1
```

请同学们思考, 这个触发器是否可以移到 sakila-schema.sql 里去执行? 为什么?

### 3.1.3关于约束

观察 sakila-schema.sql 里面的 create table store 一段:

```
335 CREATE TABLE store (
336     store_id TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
337     manager_staff_id TINYINT UNSIGNED NOT NULL,
338     address_id SMALLINT UNSIGNED NOT NULL,
339     last_update TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
340     PRIMARY KEY (store_id),
341     UNIQUE KEY idx_unique_manager (manager_staff_id),
342     KEY idx_fk_address_id (address_id),
343     CONSTRAINT fk_store_staff FOREIGN KEY (manager_staff_id) REFERENCES staff (staff_id) ON DELETE RESTRICT ON UPDATE CASCADE,
344     CONSTRAINT fk_store_address FOREIGN KEY (address_id) REFERENCES address (address_id) ON DELETE RESTRICT ON UPDATE CASCADE
345 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
346
```

回答问题:

(1) store 表上建了哪几种约束? 这些约束分别实现什么功能?

约束类型	功能

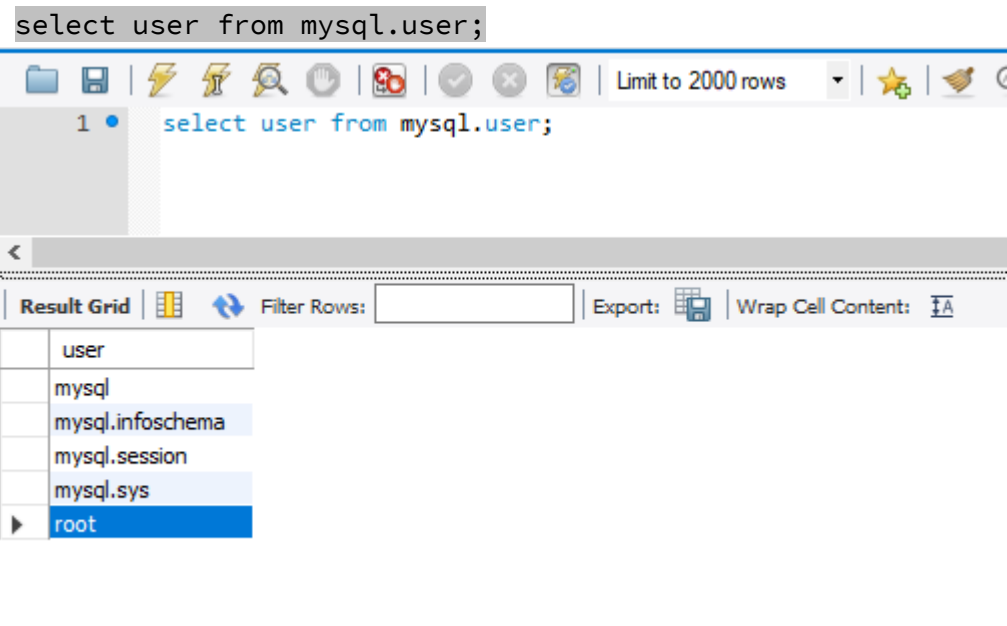
(2) 图中第 343 行的 ON DELETE RESTRICT 和 ON UPDATE CASCADE 是什么意思?

### 3.2 创建新用户并分配权限

(请把过程截图填写到实验报告中)

#### 1、创建新用户

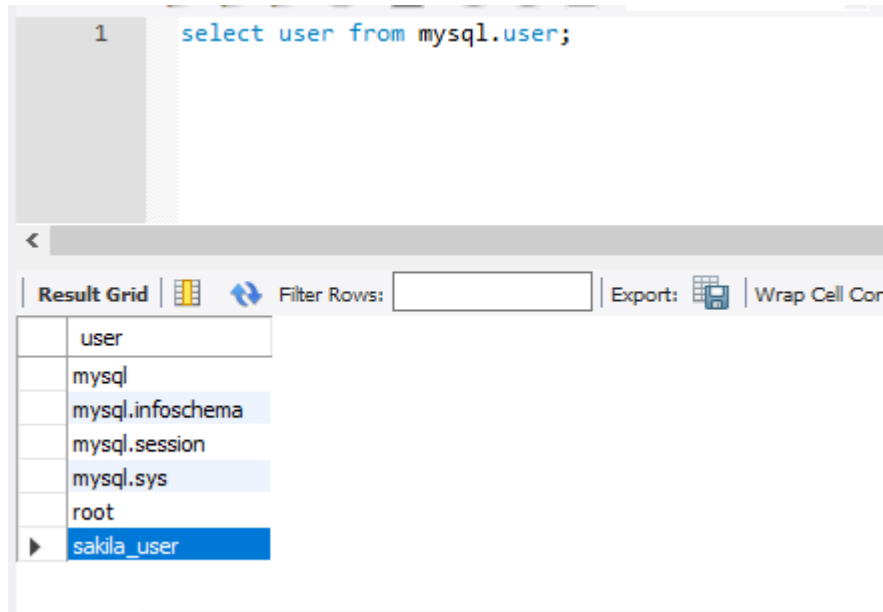
(1) 查看当前已有用户



(2) 新建 sakila\_user 用户 (密码 123456)

```
create user 'sakila_user'@'localhost' identified by '123456';
```

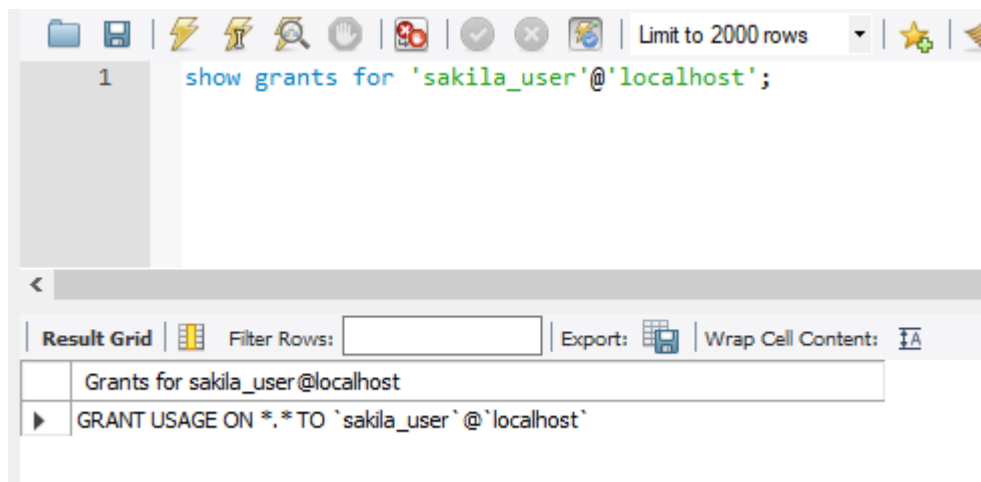
(3) 再查看用户就可以看到新用户：



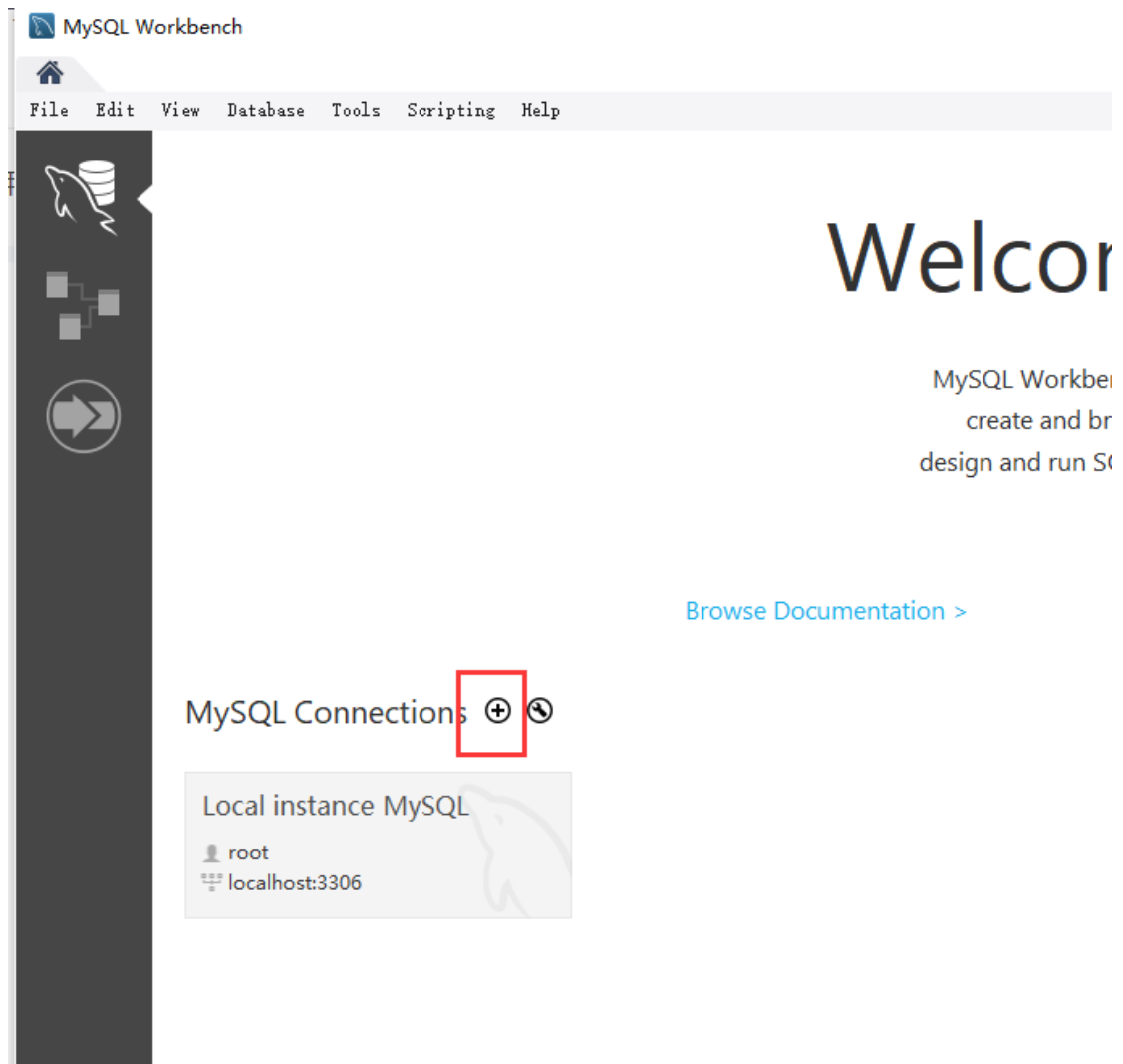
## 2、将查询 film\_list 视图的权限分配给这个新用户

(1) 先查看新用户当前的权限

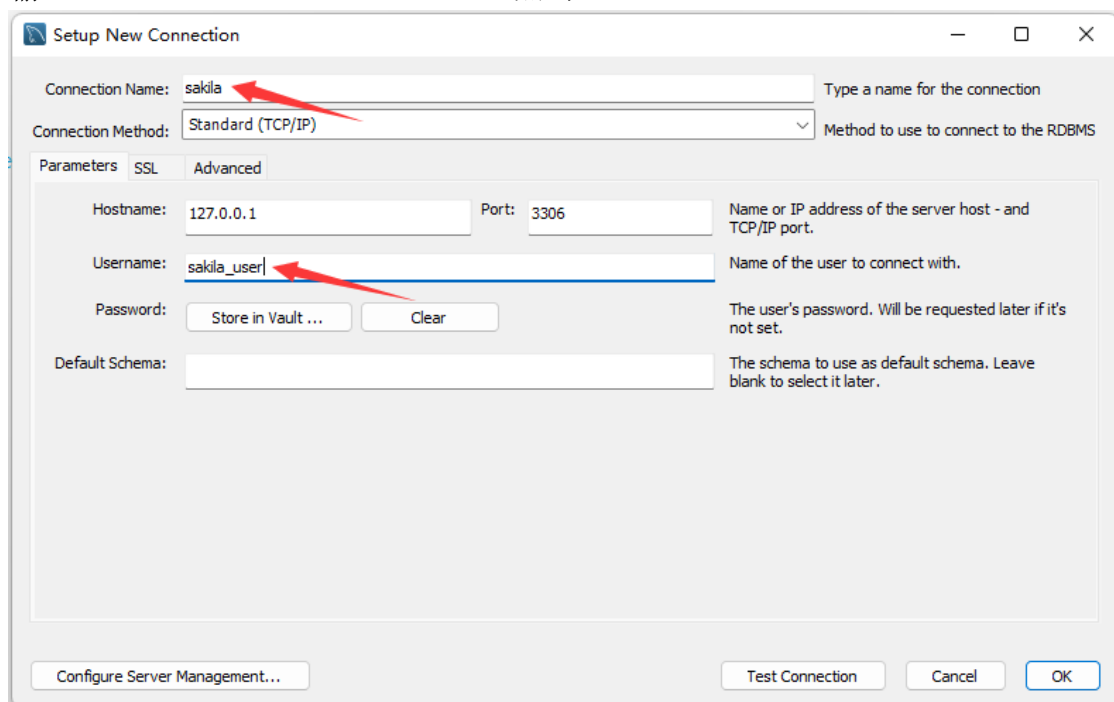
```
show grants for 'sakila_user'@'localhost';
```



(2) 看看当前的权限可以做什么  
用新用户连接数据库：

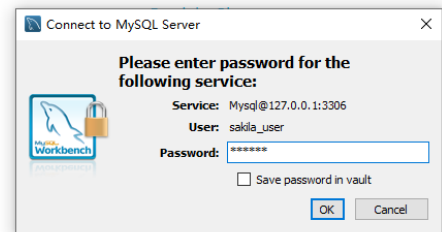
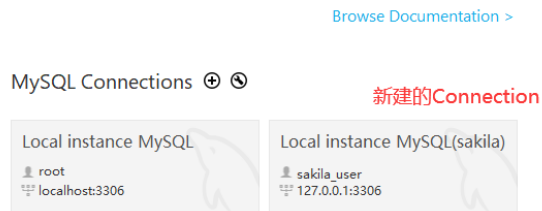


输入 Connection Name、Username，点击“OK”

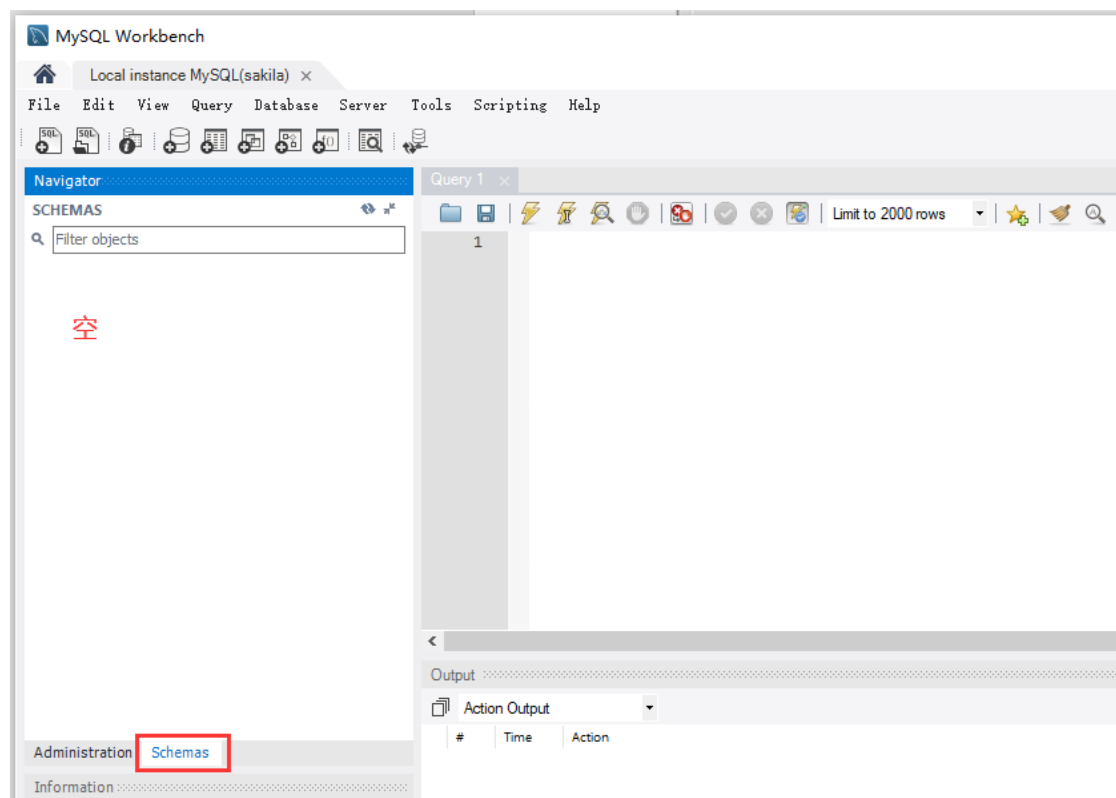


输入密码，登录：

create and browse your database schemas, work with database objects and insert, design and run SQL queries to work with stored data. You can also migrate schemas at database vendors to your MySQL database.



这时看不到原来的数据库：



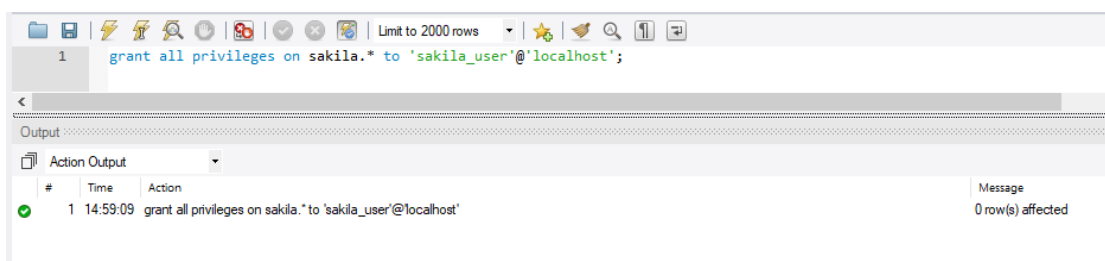
你可以用新用户新建库、表等。

(3) 把原来的 sakila 数据库的访问权限赋予 sakila\_user 用户

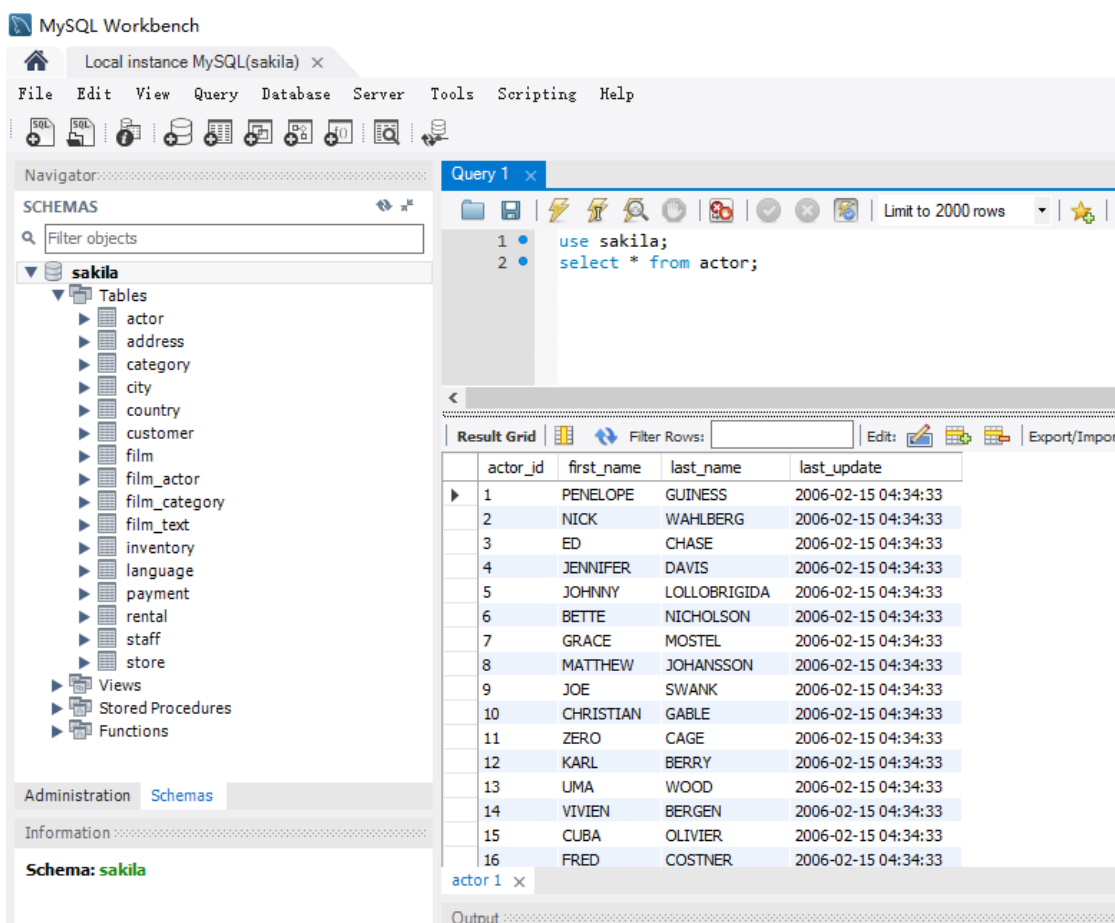
注意：要先切换回 root 用户执行下面的 grant 操作

```
grant all privileges on sakila.* to 'sakila_user'@'localhost';
```





切换到 sakila\_user 用户，这时可以看到 sakila 数据库，可以进行增删查改等操作：



更多 grant 的用法参考附录 1。

### 3.3 设计并实现

根据应用场景，为 Sakila 数据库合理地设计并实现：

1. 设计 1 个视图，至少关联 2 个表；
2. 设计 1 个触发器，需要在报告里体现触发器生效。

注意：请将创建语句、执行结果截图记录到实验报告里。

### 3.4 思考题

*（这部分不是必做题，供有兴趣的同学思考）*

在阿里开发规范里有一条“**【强制】不得使用外键与级联，一切外键概念必须在应用层解决。**”请分析一下原因。外键是否没有存在的必要？

**注意：**请将答案写到实验报告里。

## 附录 1 GRANT 命令

一、**grant** 普通数据用户，查询、插入、更新、删除 数据库中所有表数据的权利。

```
grant select on testdb.* to common_user@'%'  
grant insert on testdb.* to common_user@'%'  
grant update on testdb.* to common_user@'%'  
grant delete on testdb.* to common_user@'%'
```

或者，用一条 MySQL 命令来替代：

```
grant select, insert, update, delete on testdb.* to common_user@'%'
```

二、**grant** 数据库开发人员，创建表、索引、视图、存储过程、函数。。。等权限。

grant 创建、修改、删除 MySQL 数据表结构权限。

```
grant create on testdb.* to developer@'192.168.0.%';  
grant alter on testdb.* to developer@'192.168.0.%';  
grant drop on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 外键权限。

```
grant references on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 临时表权限。

```
grant create temporary tables on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 索引权限。

```
grant index on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 视图、查看视图源代码 权限。

```
grant create view on testdb.* to developer@'192.168.0.%';  
grant show view on testdb.* to developer@'192.168.0.%';
```

grant 操作 MySQL 存储过程、函数 权限。

```
grant create routine on testdb.* to developer@'192.168.0.%'; -- now, can show procedure status  
grant alter routine on testdb.* to developer@'192.168.0.%'; -- now, you can drop a procedure
```

```
grant execute on testdb.* to developer@'192.168.0.%';
```

### 三、grant 普通 DBA 管理某个 MySQL 数据库的权限。

```
grant all privileges on testdb to dba@'localhost'
```

其中，关键字 **privileges** 可以省略。

### 四、grant 高级 DBA 管理 MySQL 中所有数据库的权限。

```
grant all on *.* to dba@'localhost'
```

### 五、MySQL grant 权限，分别可以作用在多个层次上。

#### 1. grant 作用在整个 MySQL 服务器上：

```
grant select on *.* to dba@localhost; -- dba 可以查询 MySQL 中所有数据库中的表。  
grant all on *.* to dba@localhost; -- dba 可以管理 MySQL 中的所有数据库
```

#### 2. grant 作用在单个数据库上：

```
grant select on testdb.* to dba@localhost; -- dba 可以查询 testdb 中的表。
```

#### 3. grant 作用在单个数据表上：

```
grant select, insert, update, delete on testdb.orders to dba@localhost;
```

这里在给一个用户授权多张表时，可以多次执行以上语句。例如：

```
grant select(user_id,username) on smp.users to mo_user@'%' identified by '123345';  
grant select on smp.mo_sms to mo_user@'%' identified by '123345';
```

#### 4. grant 作用在表中的列上：

```
grant select(id, se, rank) on testdb.apache_log to dba@localhost;
```

#### 5. grant 作用在存储过程、函数上：

```
grant execute on procedure testdb.pr_add to 'dba'@'localhost'  
grant execute on function testdb.fn_add to 'dba'@'localhost'
```

### 六、查看 MySQL 用户权限

查看当前用户（自己）权限：

```
show grants;
```

查看其他 MySQL 用户权限：

```
show grants for dba@localhost;
```

七、撤销已经赋予给 **MySQL** 用户权限的权限。

revoke 跟 grant 的语法差不多，只需要把关键字 to 换成 from 即可：

```
grant all on *.* to dba@localhost;  
revoke all on *.* from dba@localhost;
```

八、**MySQL grant、revoke** 用户权限注意事项

1. grant, revoke 用户权限后，该用户只有重新连接 **MySQL** 数据库，权限才能生效。
2. 如果想让授权的用户，也可以将这些权限 grant 给其他用户，需要选项 **grant option**

```
grant select on testdb.* to dba@localhost with grant option;
```

这个特性一般用不到。实际中，数据库权限最好由 **DBA** 来统一管理。

注意：创建完成后需要执行 **FLUSH PRIVILEGES** 语句。

（参考链接 [MySQL Grant 命令 \(runoob.com\)](http://runoob.com)）