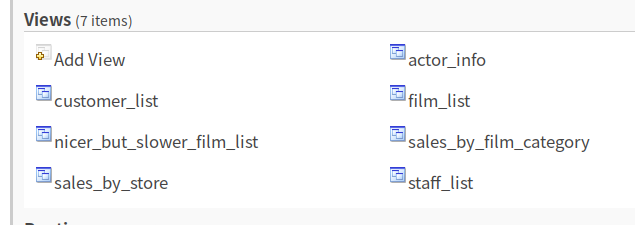
实验二报告

## 观察并回答问题

### 关于视图

1. sakila.mwb模型图中共有几个View？

一共有7个View：



1. 分析以下3个视图，回答以下问题：

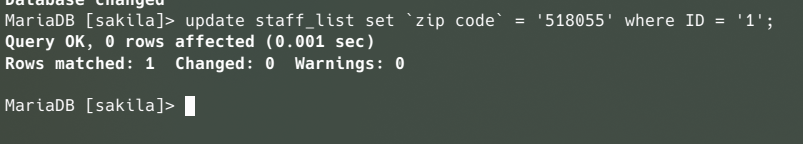
|  |  |  |
| --- | --- | --- |
| 视图名 | 关联表 | 作用 |
| actor\_info | film,film\_category,film\_actor | 检索演员的id、姓名以及出演的电影的类型和名，称使用逗号分隔。 |
| film\_list | film,film\_catagory,film\_actor, actor, | 检索电影的信息列表，含有id、标题、描述、分类、价格、长度、评分、演员。 |
| sales\_by\_store | city,country,rental, inventory,store,address, city,country,staff | 按照总销售量排序销售员和商店信息。 |

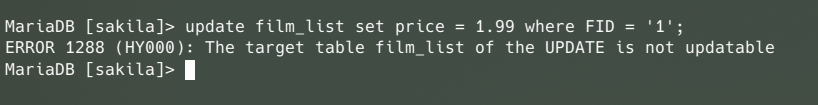
1. 分别执行以下2句SQL语句：

update staff\_list set `zip code` = '518055' where ID = '1';

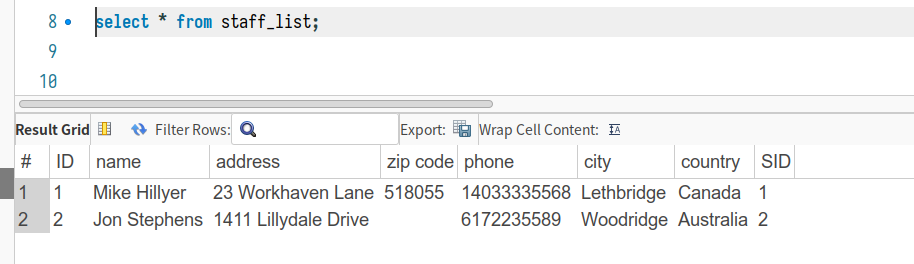
update film\_list set price = 1.99 where FID = '1';

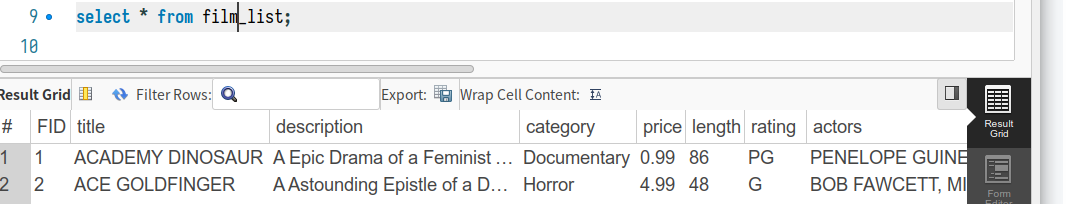
截图执行结果，并分析一下视图在什么情况下可以进行update操作，什么情况下不能？



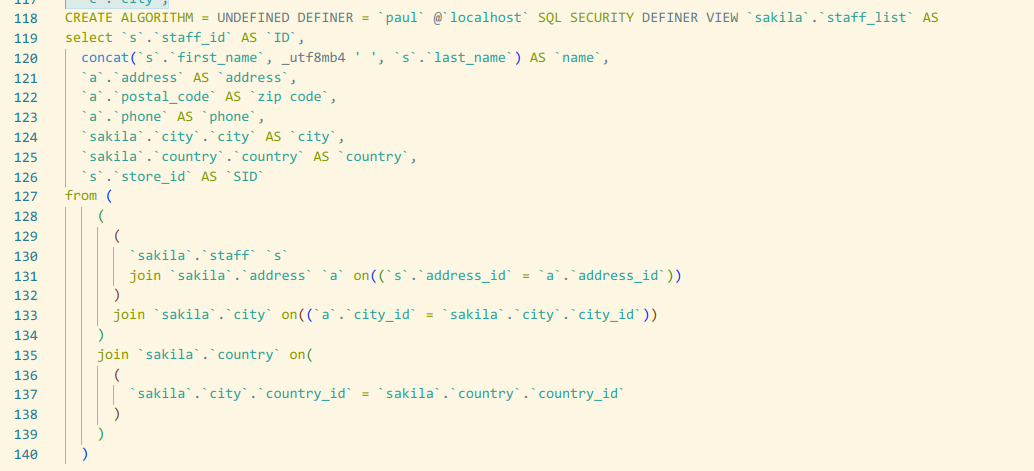


观察Select得到的值，





以及对应的代码：



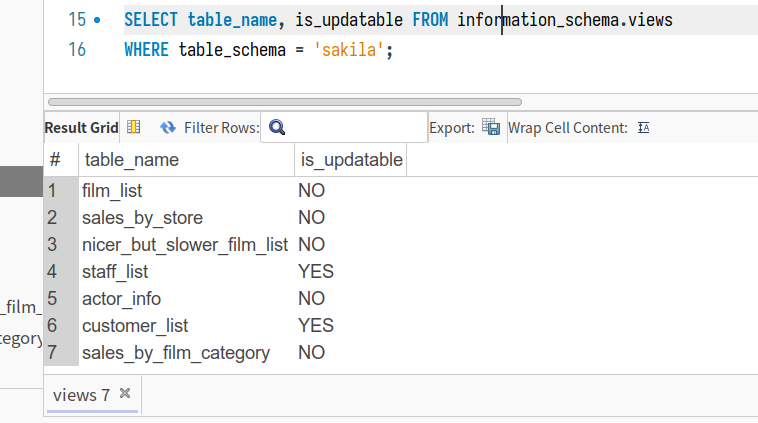


可以发现，staff\_list的select过程只有join，而film\_list的select过程包括join也包括group by。group by 即将得到的数据按照某一列的值分作小类并做一些运算得到select结果，这其中有分类再处理的过程，得到的结果与原表中并不是一一对应的关系，所以数据是只读不可写的，这样的视图的update操作是不被允许的。

1. 执行以下命令查询sakila数据库中的视图是否可更新，截图执行结果：

SELECT table\_name, is\_updatable FROM information\_schema.views

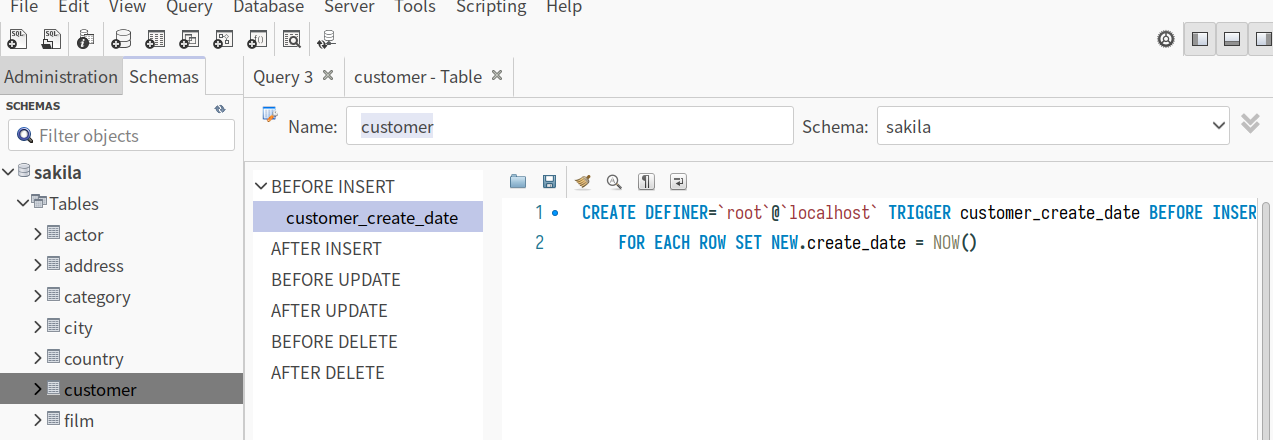
WHERE table\_schema = 'sakila';



### 关于触发器

1. 触发器customer\_create\_date建在哪个表上？这个触发器实现什么功能？

在表customer上：



实现的功能：

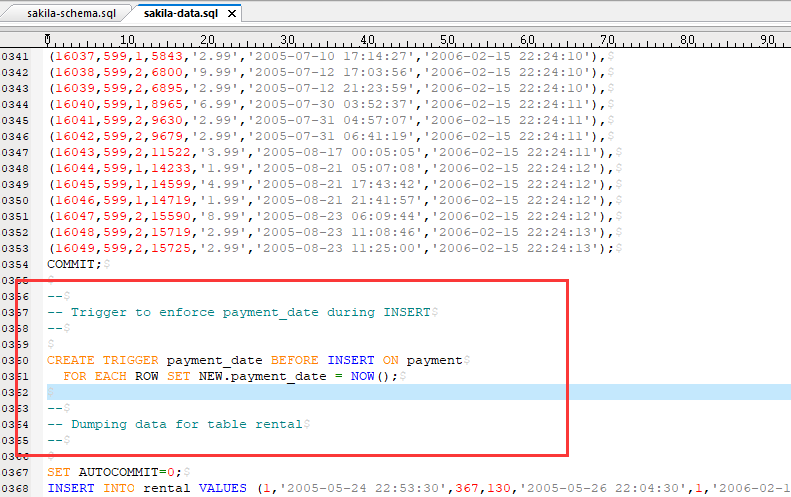
当插入customer表的条目的时候，自动设置创建条目的create\_date字段值为当前时间。

1. 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）



触发器正确生效了。

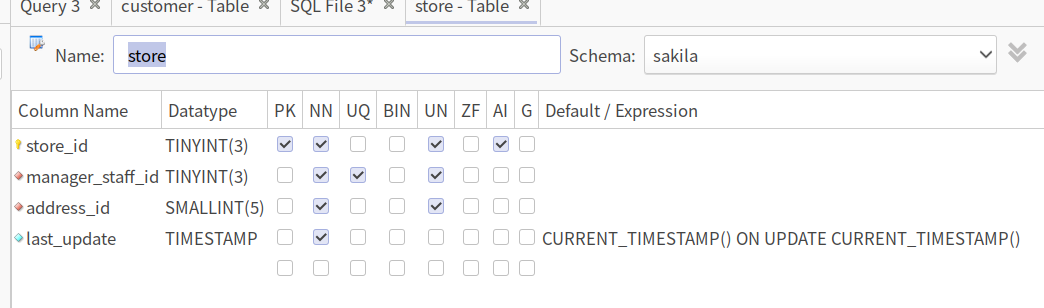
1. 我们可以看到sakila-schema.sql里的语句是用于创建数据库的结构，包括表、视图、触发器等，而sakila-data.sql主要是用于往表写入数据。但sakila-data.sql里有这样一个建立触发器payment\_date的语句，这个触发器是否可以移到sakila-schema.sql里去执行？为什么？



不可以，这条触发器是在插入数据的时候生效的，而导入数据库数据执行sql语句的过程也是插入数据的过程，如果在插入数据之前设置好那么在插入数据的时候就会生成新的数据，和原本期望导入的数据就有很大差别了。

### 关于约束

1. store表上建了哪几种约束？这些约束分别实现什么功能？（至少写3个）



CREATE TABLE store (

store\_id TINYINT UNSIGNED NOT NULL AUTO\_INCREMENT,

manager\_staff\_id TINYINT UNSIGNED NOT NULL,

address\_id SMALLINT UNSIGNED NOT NULL,

last\_update TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (store\_id),

UNIQUE KEY idx\_unique\_manager (manager\_staff\_id),

KEY idx\_fk\_address\_id (address\_id),

**CONSTRAINT** fk\_store\_staff **FOREIGN KEY** (manager\_staff\_id) **REFERENCES** staff (staff\_id) **ON DELETE** RESTRICT ON UPDATE CASCADE,

**CONSTRAINT** fk\_store\_address **FOREIGN KEY** (address\_id) **REFERENCES** address (address\_id) **ON DELETE** RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

|  |  |
| --- | --- |
| 约束类型 | 功能 |
| 主键 PRIMARY KEY | 将当前键设定为主键 |
| 非空键 NOT NULL | 当前键值不能设置为 NULL |
| 唯一值 UNIQUE | 当前列的当前值在本表中只能出现一次 |

1. 图中第343行的ON DELETE RESTRICT和ON UPDATE CASCADE是什么意思？

ON DELETE RESTRICT：

1. ON DELETE：在删除时执行操作
2. RESTRICT：在父表中删除记录的时候，首先检查该记录是否有对应外键，如果有则拒绝删除

ON UPDATE CASCADE：

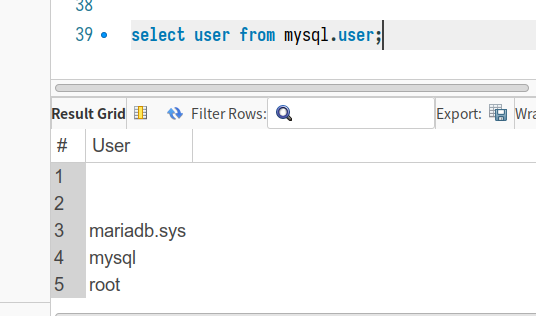
1. ON UPDATE：在更新时执行操作
2. CASCADE：当在父表中更新对应记录的时候，首先检查该记录是否有对应的外键，如果有则同时更新外键在字表中的记录

## 创建新用户并分配权限

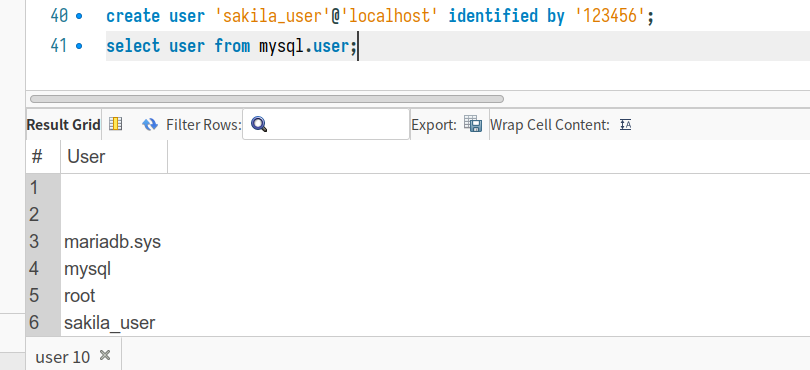
（截图语句和执行结果）

1. 执行命令新建sakila\_user用户（密码123456）；

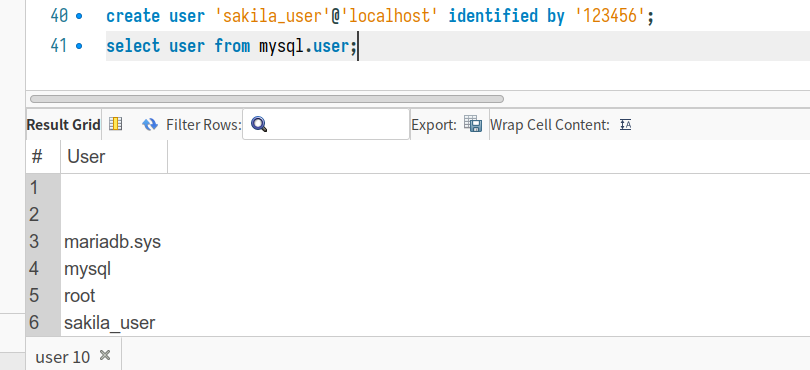
查看当前用户：



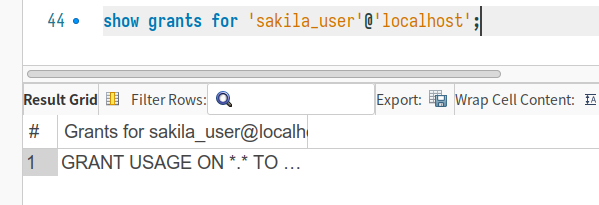
创建新用户：

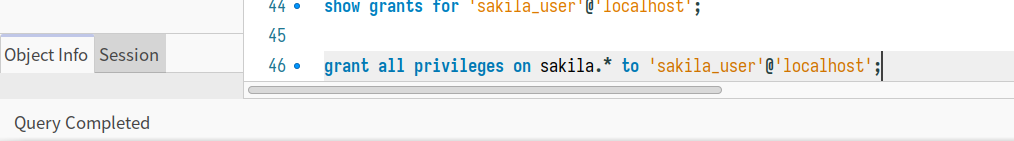


1. 执行命令查看当前已有用户；

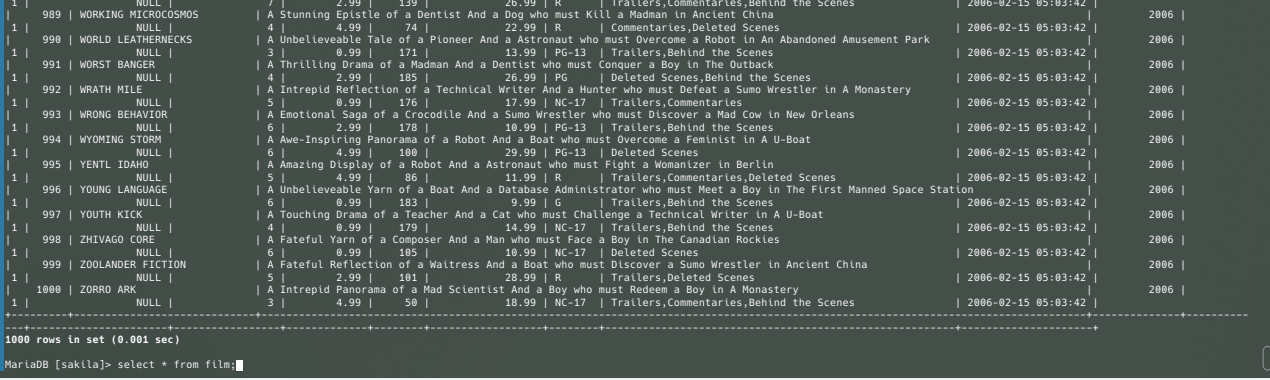


1. 执行命令把sakila数据库的访问权限赋予sakila\_user用户；





1. 切换到sakila\_user用户，执行select \* from film操作。



## 设计并实现

根据应用场景，为Sakila数据库合理地设计并实现：

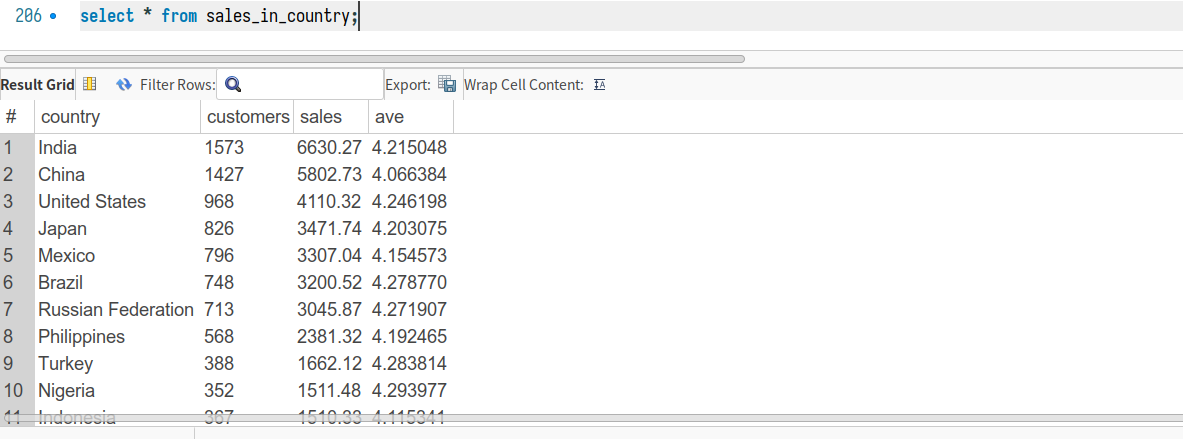
（截图语句和执行结果）

1. 设计1个视图，至少关联2个表；
2. 执行新建视图的语句，并截图SQL和执行结果：

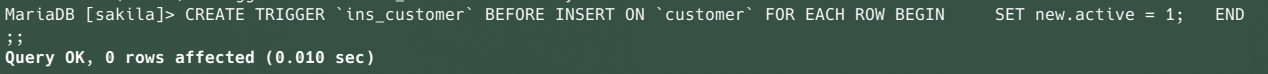
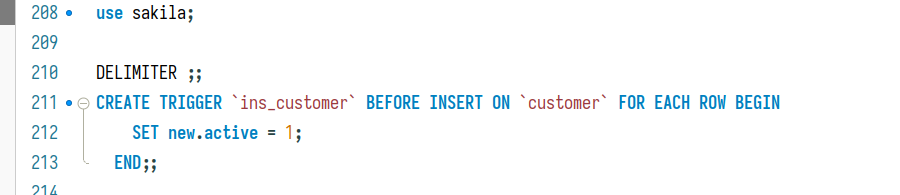


作用：按国家统计总销售金额，按从大到小排序，并且统计居住与各个国家的顾客数量。

1. 执行select \* from [视图名]，截图执行结果：

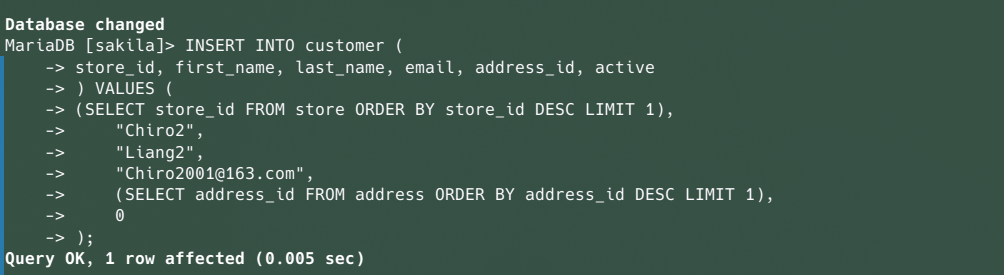


1. 设计1个触发器，需要体现触发器生效。
2. 执行新建触发器的语句，并截图SQL和执行结果：

本触发器用于强制使插入行的 active 字段设置为 1。

1. 验证触发器是否生效，截图验证过程：

插入一条数据，设置插入时 active=0：

检索这条数据：

可以看到 active = 1，触发器已经生效。

## 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。你认为外键是否没有存在的必要？

我认为外键在存在上是必要的，但是使用上不是必要的。

外键能够保证数据库自身的数据一致性和完整性，当外部程序并不100%可靠的时候，数据库内部的外键能够维护数据的逻辑，减少因为程序错误导致的数据出错，也有助于数据的恢复。同时，如果不使用外键，会导致数据库内的数据冗余，系统性能会受到数据库 IO 速度瓶颈限制。

但是使用外键，会导致更大的性能问题，查表的时候会联系查找到更多的表；同时会造成维护的繁琐，需要额外考虑多个表之间的逻辑和数据关系，更加难以修改多个表之间的链接逻辑。

综上，外键在数据库比较简单、不需要后期维护的时候，能够更好地组织数据库的数据逻辑；但是在较大的数据库项目中，它并不能减少系统的复杂性，反而引入更多的限制因素，从而增加了系统的复杂性和维护难度。所以外键在合适的项目中可以存在，但是在公司、大项目中使用并不推荐。