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？

7

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 | actor、film、film\_actor、category、film\_category、 | 展示演员的id、姓、名、演过影片的影片信息(包含电影类型和主题组成的字符串)  (影片类型1:影片主题1,影片主题2;) |
| film\_list | film、category、actor、film\_category、film\_actor | 展示电影id、主题、描述、类型、价格、时长、包含的演员 |
| sales\_by\_store | payment、rental、inventory、store、address、city、country、staff | 展示商店的位置(城市,国家)、商店管理者姓名、总折扣销售额 |

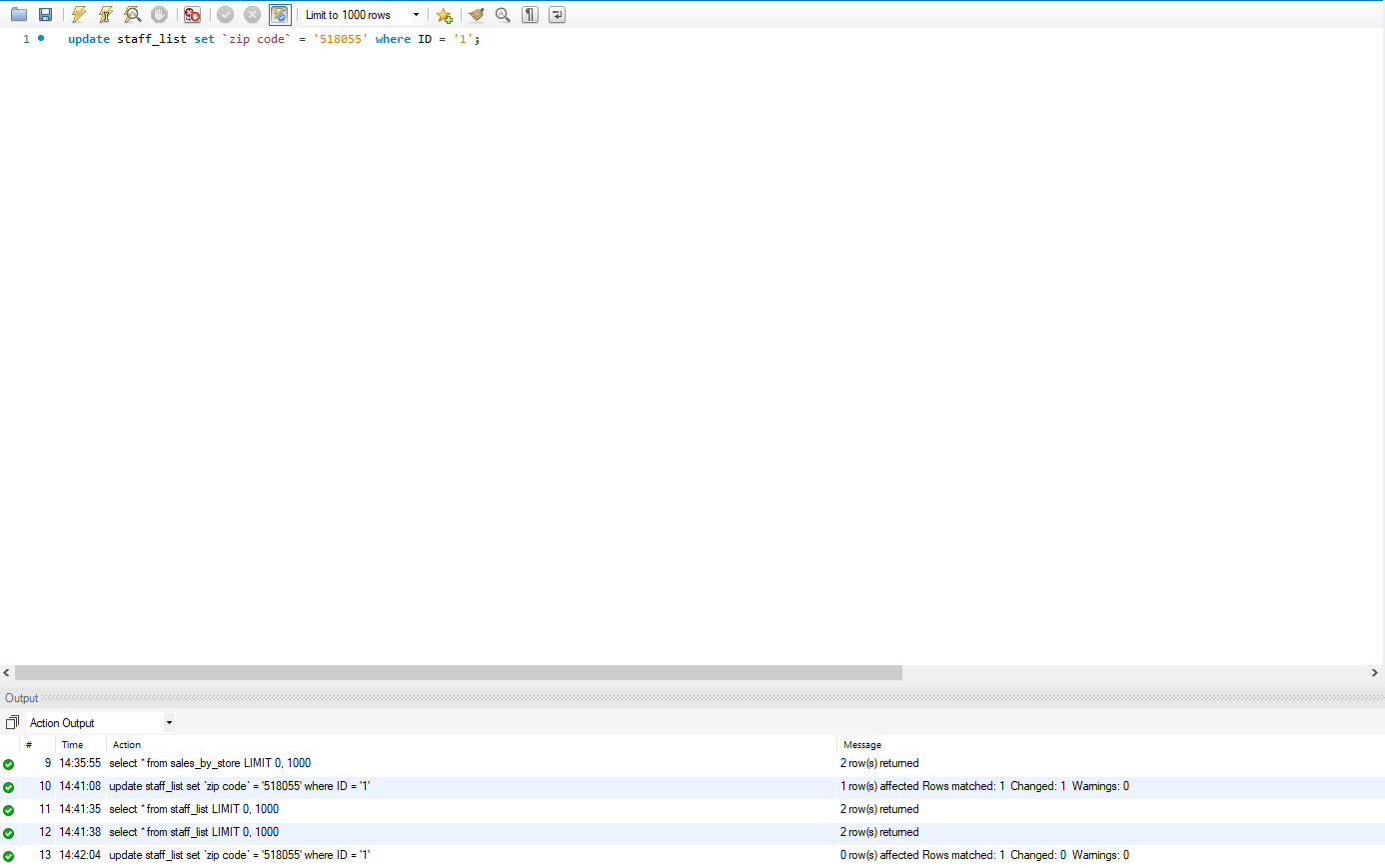
1. 分别执行以下2句SQL语句：

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？

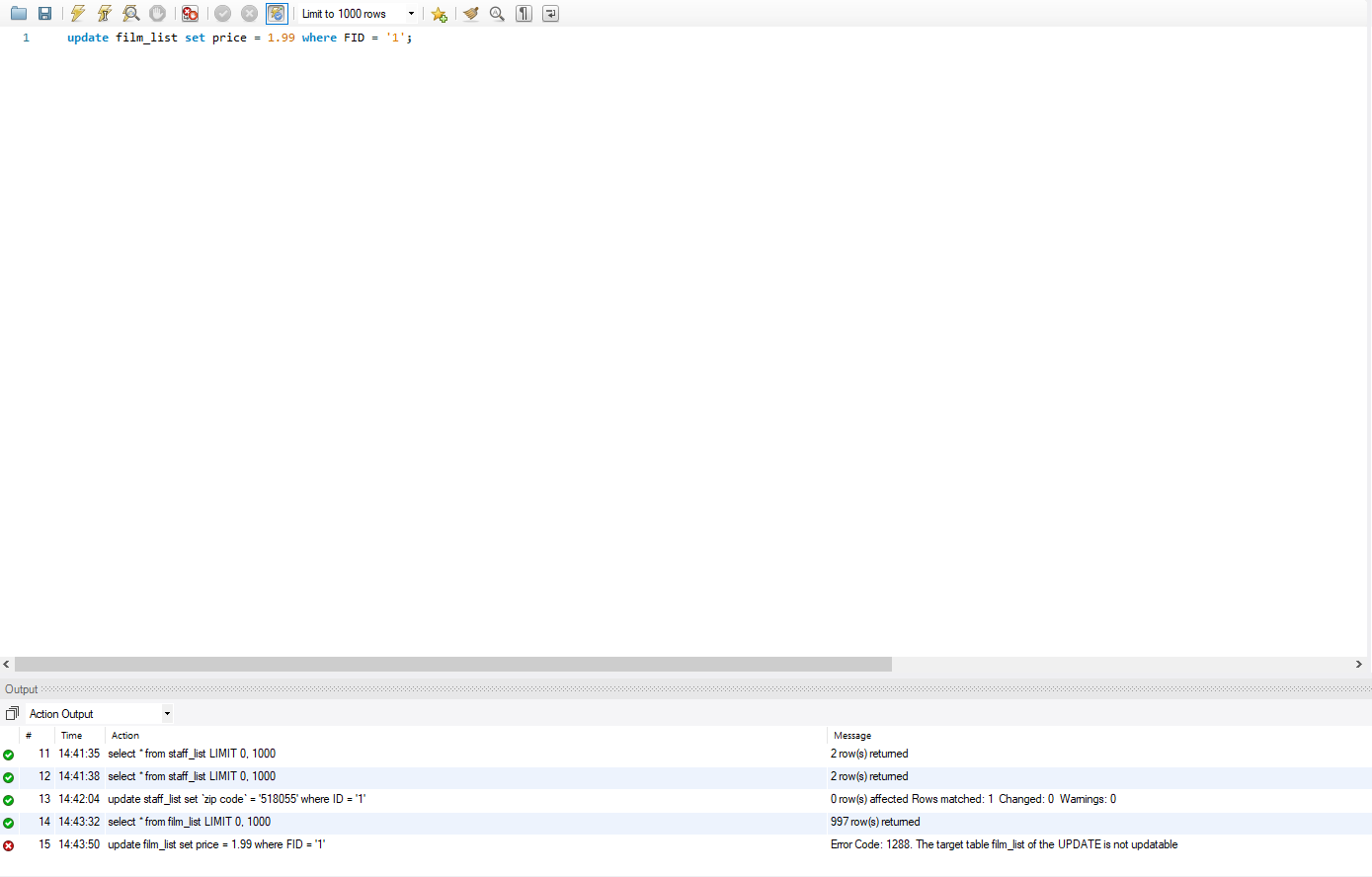
update staff\_list set `zip code` = '518055' where ID = '1';



分析：

第一条update语句修改成功，查看staff\_list视图，发现修改的zip code来源于address的pental\_code而pental\_code是只存在于address基表的数据，而address的主键address\_id作为staff的外键与staff相连，所以修改只影响了address基本表，而且该视图的构建没有聚合函数、派生列、分组函数，修改成功。

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



分析：

第二条update语句修改失败，查看film\_list视图，发现修改的price来源于film的rental\_rate，该视图的构建运用了分组函数，因此不能update，修改失败。

视图update修改条件：

1、select语句在选择列表中没有聚合函数，也不包含TOP,GROUP BY,UNION(除非视图是分区视图)或DISTINCT子句。聚合函数可以用在FROM子句的子查询中，只要不修改函数返回的值。

2、select语句的选择列表中没有派生列。派生列是由任何非简单列表达式(使用函数、加法或减法运算符等)所构成的结果集列。

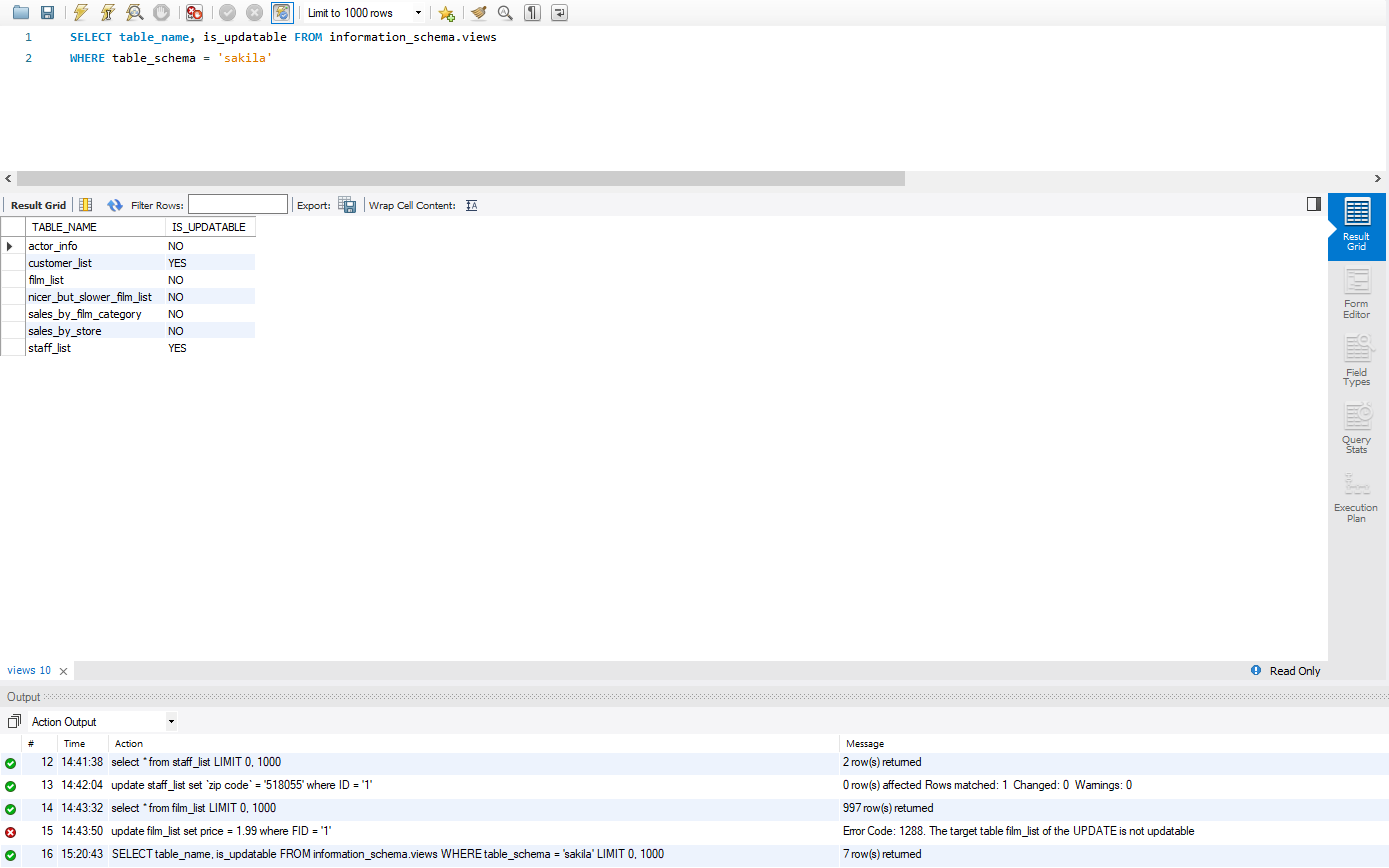
3、select语句中的FROM子句至少引用一个表。select语句不能只包含非表格格式的表达式(即不是从表派生出的表达式)。

4、INSERT,UPDATE和DELETE语句在引用可更新视图之前，也必须如上述条件指定的那样满足某些限制条件。只有当视图可更新，并且所编写的UPDATE或INSERT语句只修改视图的FROM子句引用的一个基表中的数据时，UPDATE和INSERT语句才能引用视图。

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';

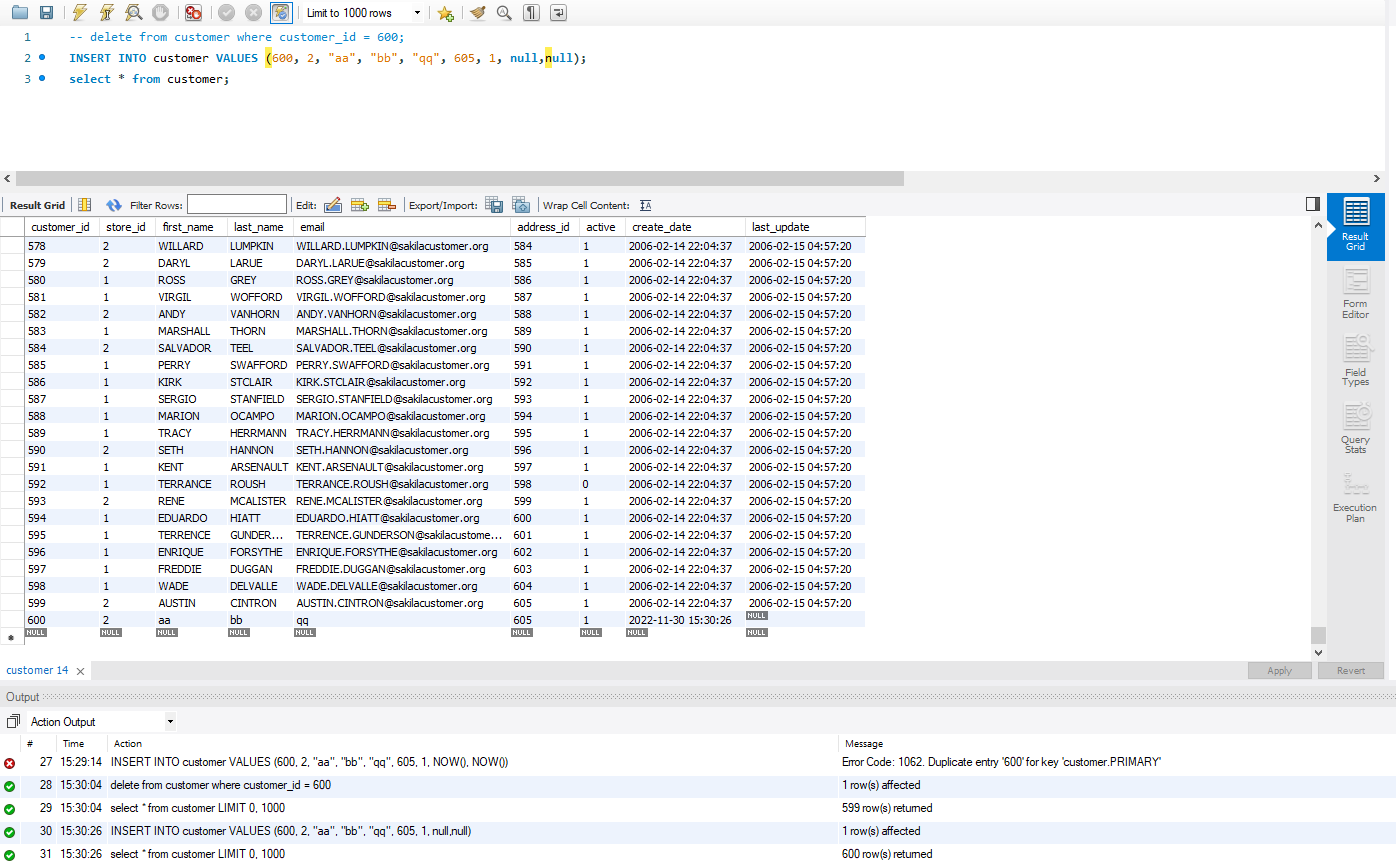


### 关于触发器

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

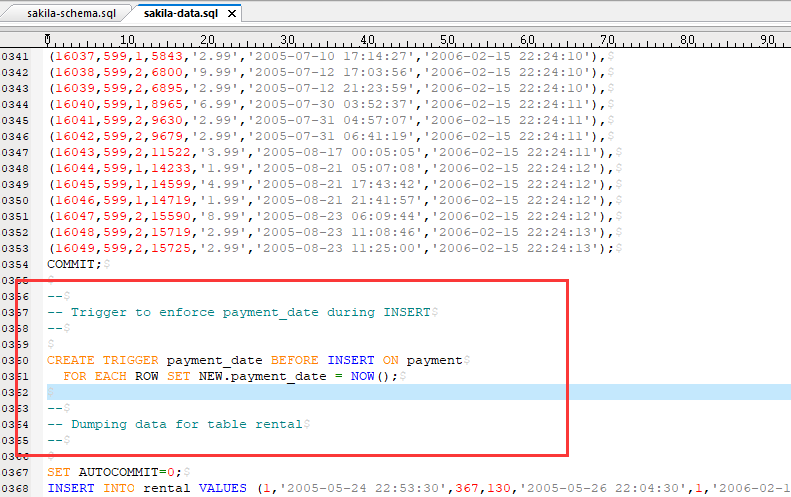
customer表上，在customer表的插入操作中，每次插入操作都自动将插入信息元组中的create\_date赋值为当前时间NOW()

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）



发现customer\_id为600的数据中(新插入的数据)create\_date为当前时间

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？



不能，因为数据库先执行sakila-schema.sql里的语句是用于创建数据库的结构，再执行sakila-data.sql里的语句将初始数据插入数据库中，payment\_date触发器用于将之后每次插入payment的数据的payment\_date赋值为当前时间，若是移动到sakila-schema.sql里执行，则sakila-data.sql里的payment的初始数据插入操作将把已经存在于payment的数据的payment\_date都改为当前时间，错误。

### 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）

|  |  |
| --- | --- |
| 约束类型 | 功能 |
| 主键约束PK = PRIMARY KEY | 主键约束每个表只有一个，表示该属性是主键，用以确定数据的唯一性和标识该行数据 |
| 唯一约束UQ = UNIQUE | 唯一约束，表示该属性的值是唯一的，用以确定数据的唯一性 |
| 非空约束NN = NOT NULL | 非空约束，表示该属性的值不能为空 |

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

ON DELETE RESTRICT：当在父表(即外键的来源表)中删除对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除。

ON UPDATE CASCADE：当在父表(即外键的来源表)中更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许更新。

## 创建新用户并分配权限

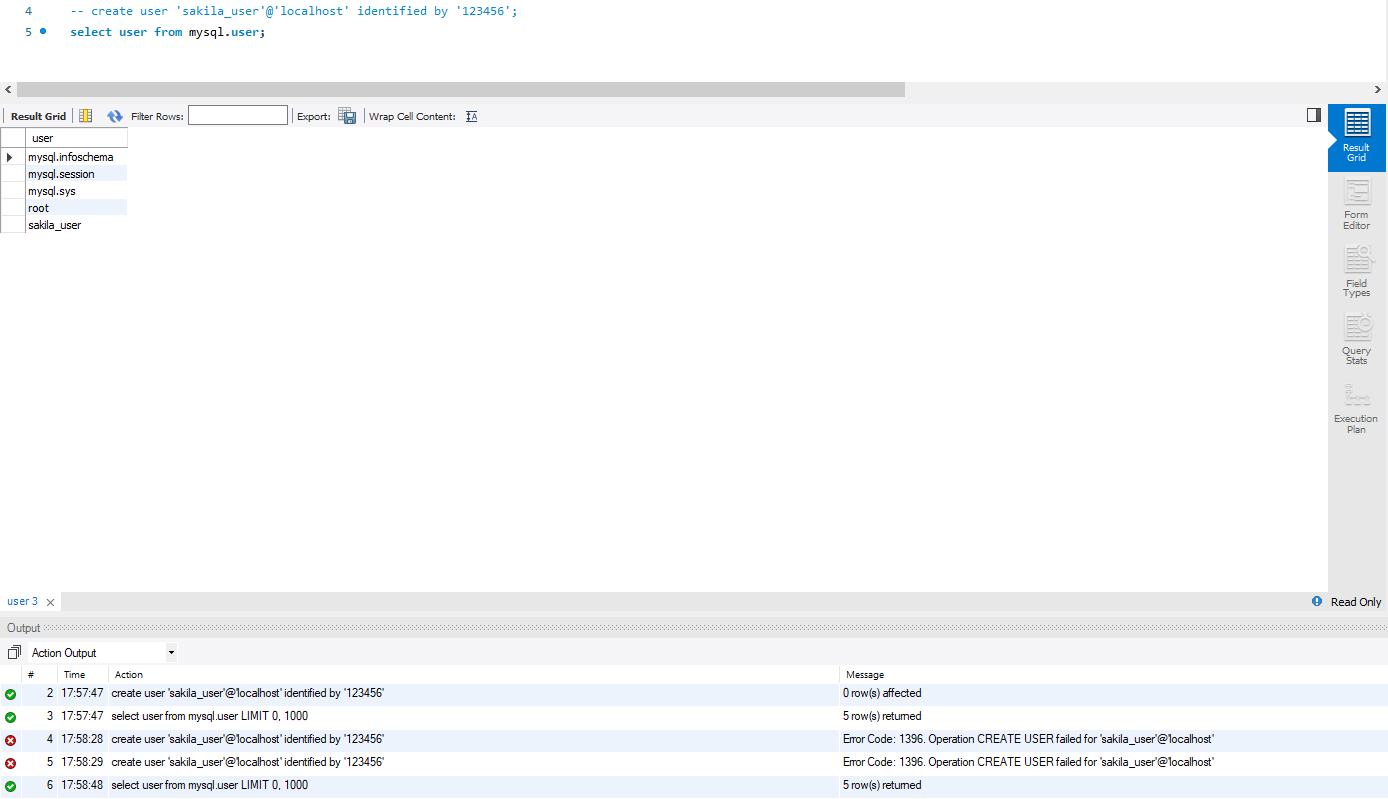
（截图语句和执行结果）

1. 执行命令新建sakila\_user用户（密码123456）；

create user 'sakila\_user'@'localhost' identified by '123456';



1. 执行命令查看当前已有用户；

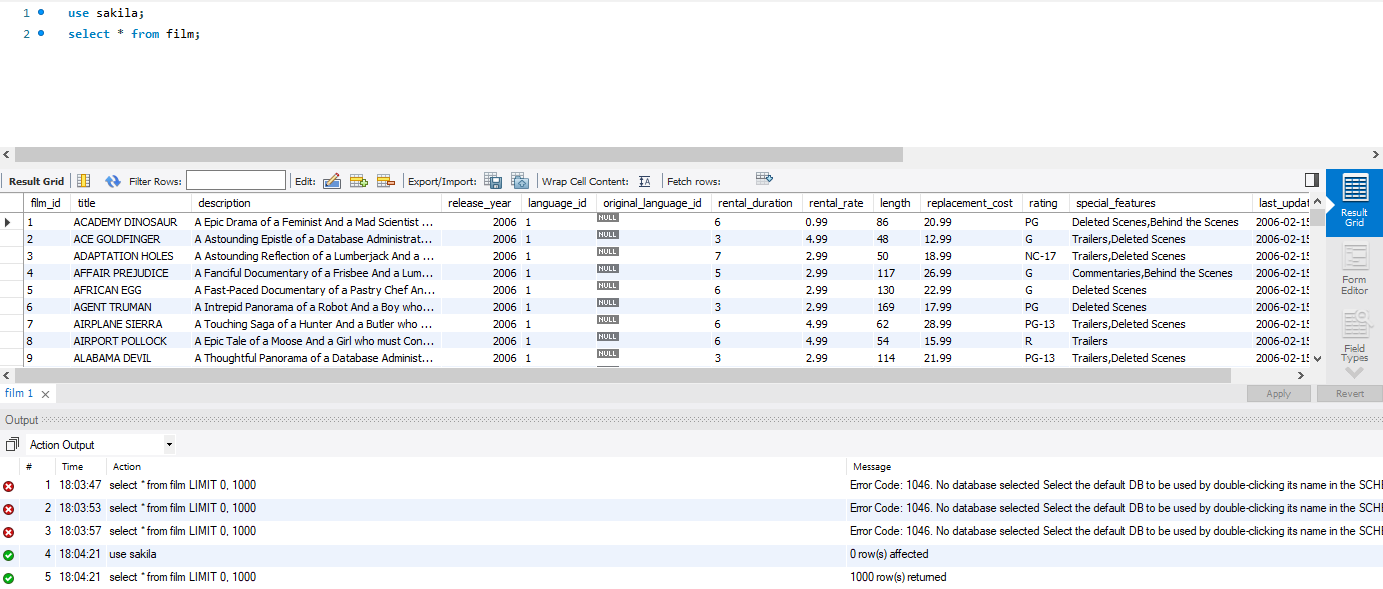


1. 执行命令把sakila数据库的访问权限赋予sakila\_user用户；

grant all privileges on sakila.\* to 'sakila\_user'@'localhost';



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：

use sakila;

CREATE VIEW category\_list

AS

SELECT distinct category.category\_id as id, category.name as name ,

(SELECT GROUP\_CONCAT(film.title ORDER BY film.title SEPARATOR ', ')

FROM film

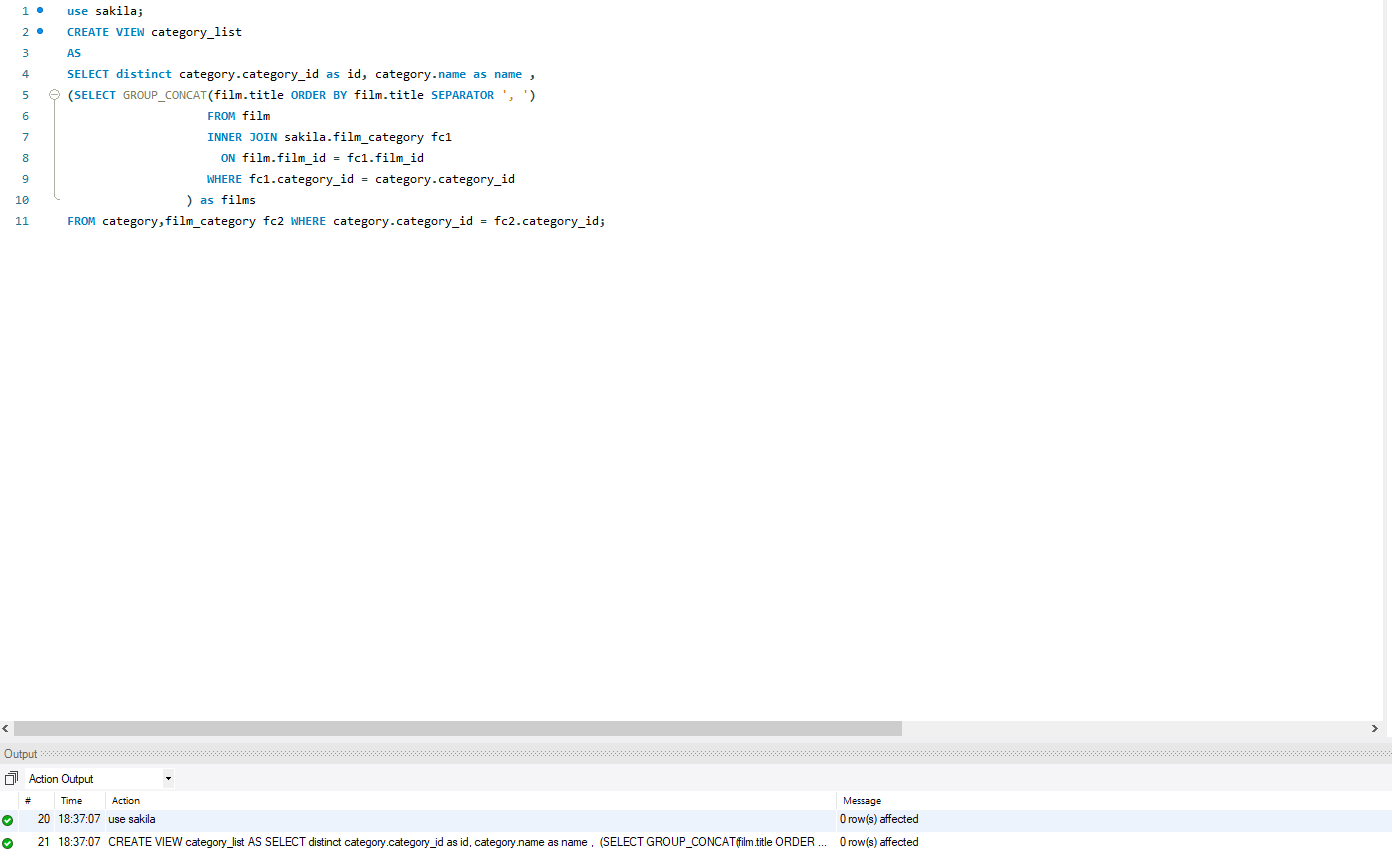
INNER JOIN sakila.film\_category fc1

ON film.film\_id = fc1.film\_id

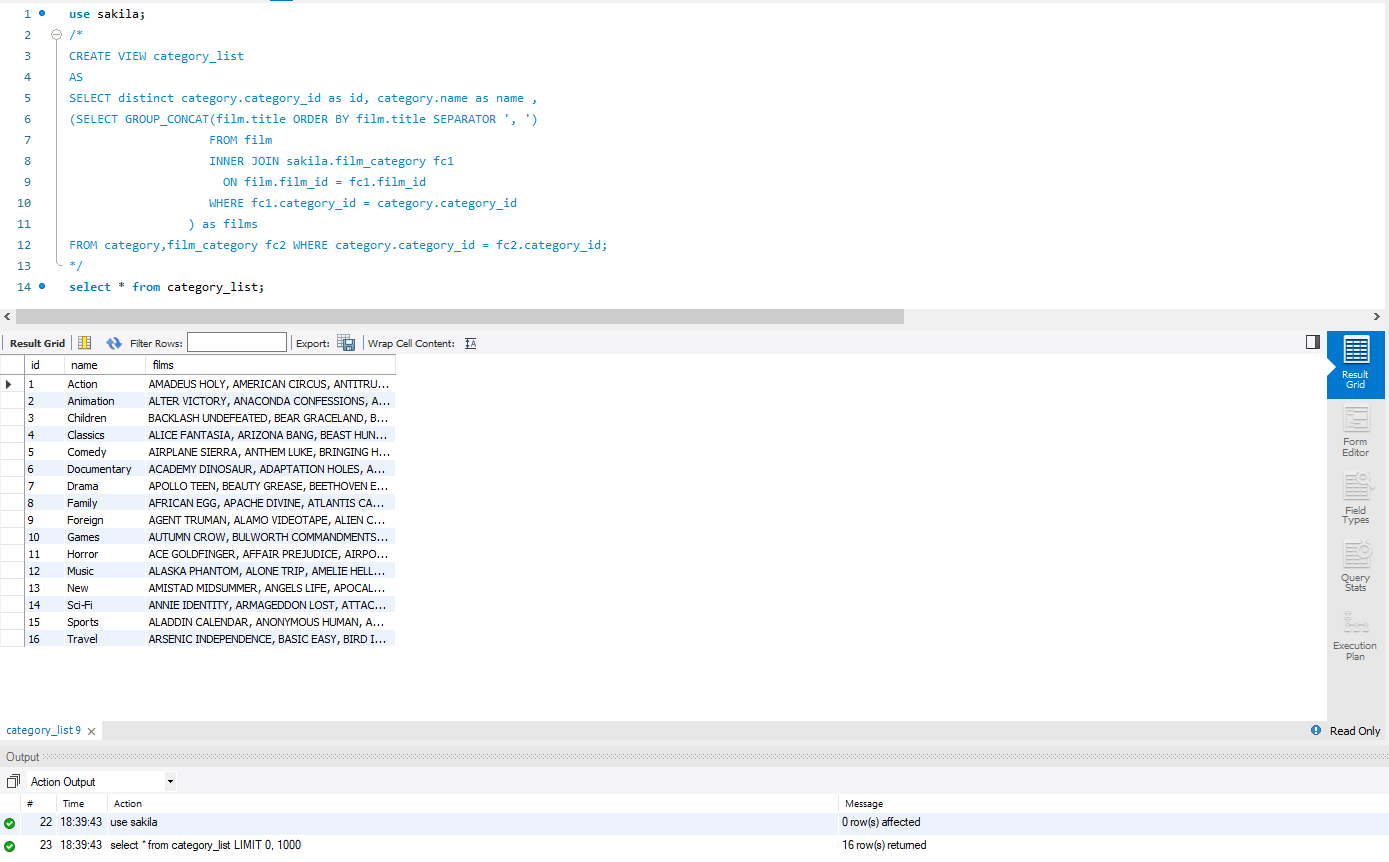
WHERE fc1.category\_id = category.category\_id

) as films

FROM category,film\_category fc2 WHERE category.category\_id = fc2.category\_id;



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：

CREATE TRIGGER customer\_update\_change\_last\_update BEFORE UPDATE ON customer

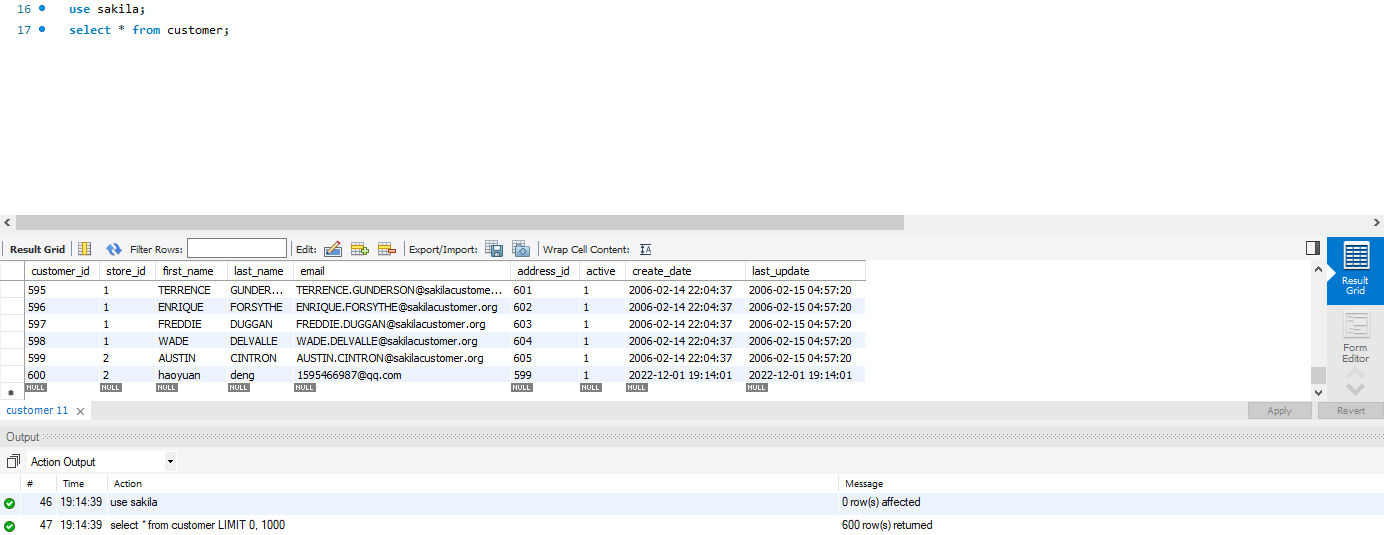
FOR EACH ROW SET new.last\_update = NOW();

该触发器用以在每次修改customer表中数据时自动更新修改时间

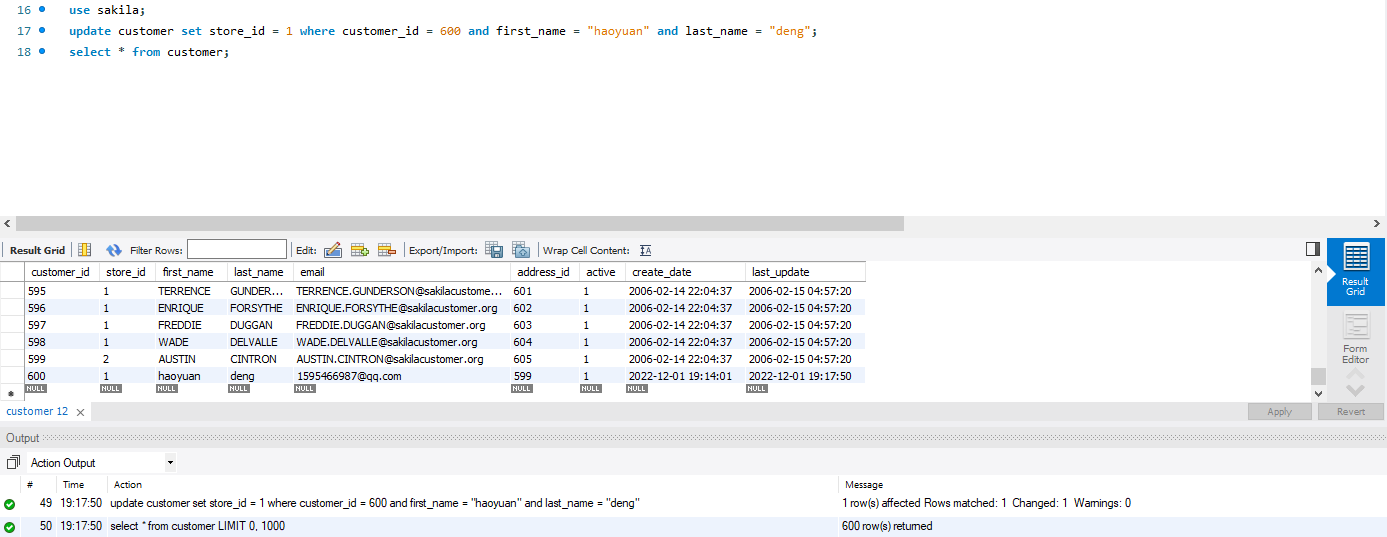


1. 验证触发器是否生效，截图验证过程：

先插入一条自己的数据(customer\_id = 600)



进行自己数据的更新之后，查看last\_update时间是否发生变化



发现update\_time产生了改变，变成了当时时间，触发器生效。

## 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？

原因：太多外键和级联的使用会大大增加表与表之间的耦合度和复杂度，让错误修改表的某个属性可能造成整个数据库的错误。

有必要，虽然外键的存在很容易让表之间的耦合性大大提升，但可以保持数据一致性，完整性，主要目的是控制存储在外键表中的数据。