

实验二报告

一、 观察并回答问题

1. 关于视图

(1) sakila.mwb 模型图中共有几个 View?

7

(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	actor,film_actor,film_category,category	得到关于每个演员，其姓名（first_name+last_name, ）以及演过的各类型电视剧的名称
film_list	actor,category,film_category,,film,film_actor	得到影片的 ID，名字，类型，描述，售价，长度，等级，以及其中所有演员的名字
sales_by_store	Payment,rental,inventory,store,address,city,country,staff	得到两个国家的商店，其经理名字以及租借得到的报酬总和

(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？

```
sakila> update staff_list set `zip code` = '518055' where ID = '1'
[2022-11-30 14:51:21] 1 row affected in 25 ms
```

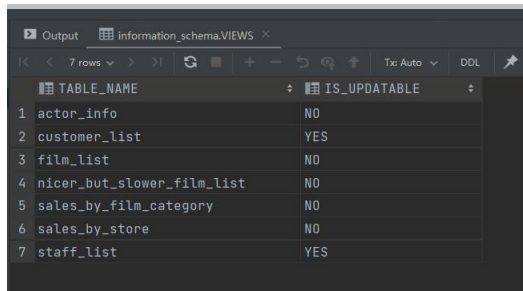
```
sakila> update film_list set price = 1.99 where FID = '1'
[2022-11-30 14:52:45] [HY000][1288] The target table film_list of the UPDATE is not updatable
[2022-11-30 14:52:45] [HY000][1288] The target table film_list of the UPDATE is not updatable
```

若视图定义中含有 GROUP BY 子句，则此视图不允许更新。

(4) 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';



TABLE_NAME	IS_UPDATABLE
actor_info	NO
customer_list	YES
film_list	NO
nicer_but_slower_film_list	NO
sales_by_film_category	NO
sales_by_store	NO
staff_list	YES

2. 关于触发器

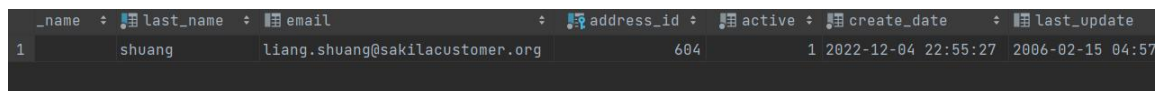
- (1) 触发器 customer_create_date 建在哪个表上？这个触发器实现什么功能？

customer 表，功能是，每当插入一条新的 customer 数据时，create_date 自动更新为插入的当前时间

- (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）

```
insert into customer values (600,1,'liang','shuang','liang.shuang@sakilacustomer.org',604,1,'2006-02-14 22:04:36');
```

```
sakila> insert into customer values (600,1,'liang','shuang','liang.shuang@sakilacustomer.org',604,1,'2006-02-14 22:04:36',
[2022-12-04 22:55:27] 1 row affected in 25 ms
sakila> select * from customer where customer_id=600
[2022-12-04 22:55:38] 1 row retrieved starting from 1 in 123 ms (execution: 9 ms, fetching: 114 ms)
```



_name	last_name	email	address_id	active	create_date	last_update
1	shuang	liang.shuang@sakilacustomer.org	604	1	2022-12-04 22:55:27	2006-02-15 04:57

- (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 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？

```
sakila-schema.sql sakila-data.sql x
0 10 20 30 40 50 60 70 80 90
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
```

不可以，触发器的定义需要放在创建表之后，否则在我们插入初始数据的时候，其 payment_date 全部会定义成现在的时间，和我们预期插入的数据是不符合的

3. 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, store_id	唯一标识商店的编号
非空约束, manager_staff_id	不为空，保证数据完整性
唯一约束, manager_staff_id	保证数据唯一性

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

ON DELETE restrict(约束): 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。

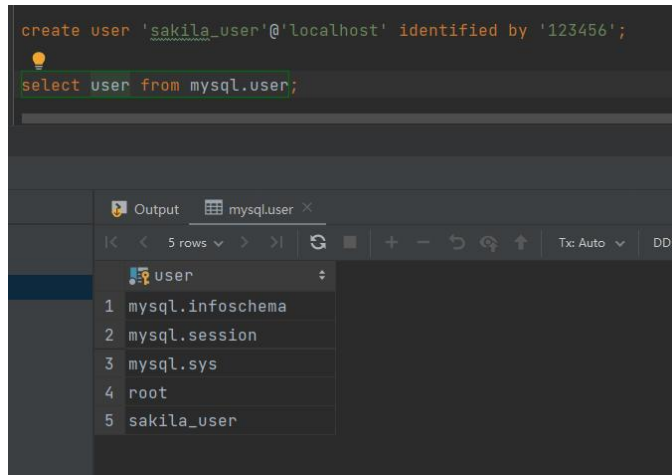
ON UPDATE restrict(约束): 当在父表(即外键的来源表)中更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许更新

二、 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）；
- (1) (2) 放在一起了
- (2) 执行命令查看当前已有用户；

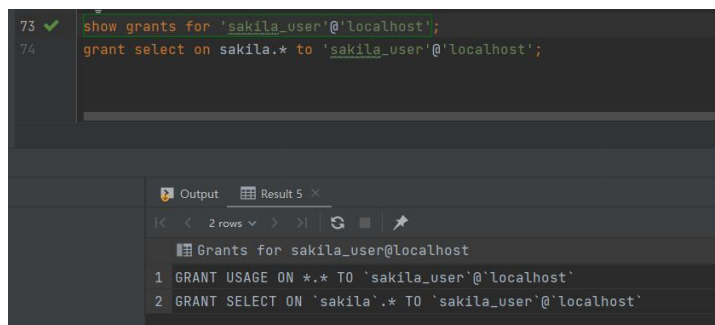
```
create user 'sakila_user'@'localhost' identified by '123456';
select user from mysql.user;
```



user
mysql.infoschema
mysql.session
mysql.sys
root
sakila_user

(3) 执行命令把 sakila 数据库的访问权限赋予 sakila_user 用户；

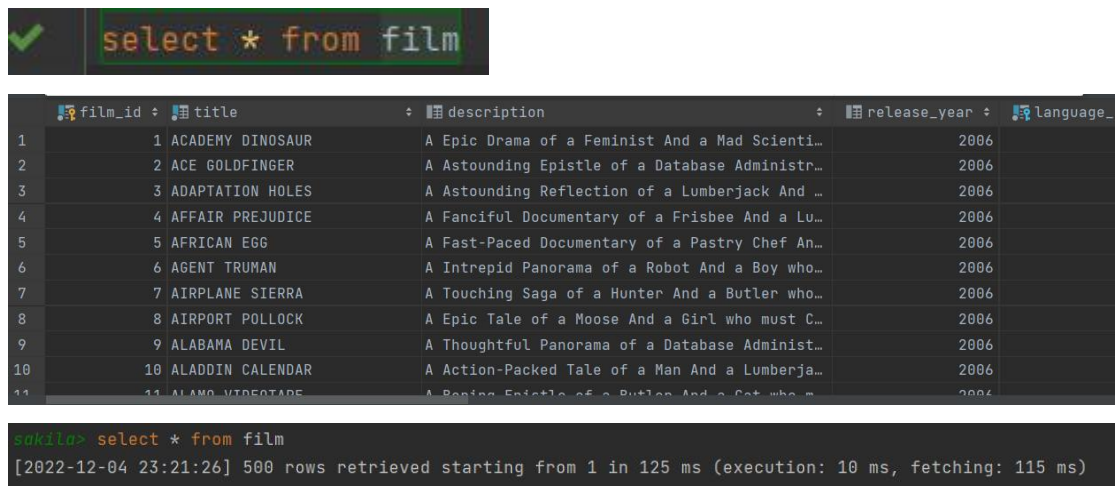
```
show grants for 'sakila_user'@'localhost';
grant select on sakila.* to 'sakila_user'@'localhost';
```



Grants for sakila_user@localhost
GRANT USAGE ON *.* TO 'sakila_user'@'localhost'
GRANT SELECT ON `sakila`.* TO 'sakila_user'@'localhost'

(4) 切换到 sakila_user 用户，执行 select * from film 操作。

```
select * from film
```



film_id	title	description	release_year	language_code
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scienti...	2006	
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administr...	2006	
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And ...	2006	
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lu...	2006	
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...	2006	
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who...	2006	
7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who...	2006	
8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must C...	2006	
9	ALABAMA DEVIL	A Thoughtful Panorama of a Database Administ...	2006	
10	ALADDIN CALENDAR	A Action-Packed Tale of a Man And a Lumberja...	2006	
11	ALAMO VIDEOTAPE	A Boring Epistle of a Butler And a Cat who p...	2006	

```
sakila> select * from film
[2022-12-04 23:21:26] 500 rows retrieved starting from 1 in 125 ms (execution: 10 ms, fetching: 115 ms)
```

三、设计并实现

根据应用场景，为 Sakila 数据库合理地设计并实现：

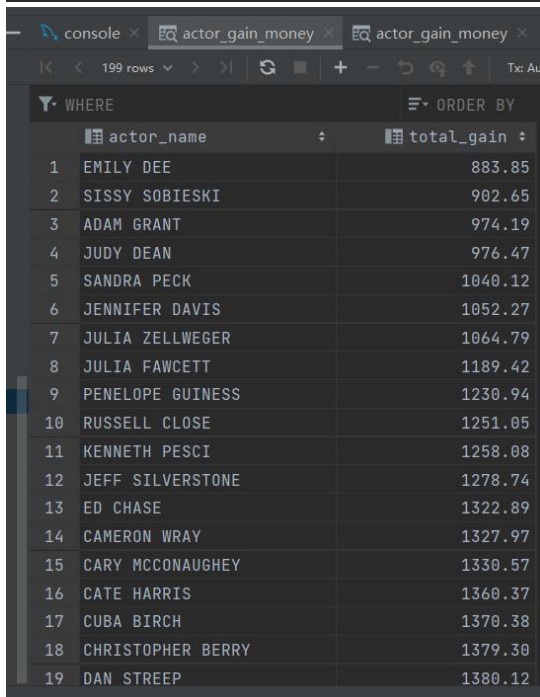
(截图语句和执行结果)

1. 设计 1 个视图，至少关联 2 个表；

(1) 执行新建视图的语句，并截图 SQL 和执行结果：

```
create definer = root@localhost view actor_gain_money as
select concat('sakila`.`actor`.`first_name`, ' ', 'sakila`.`actor`.`last_name`) AS `actor_name`,
       sum('sakila`.`payment`.`amount`) AS `total_gain`
from `sakila`.`actor`
    join `sakila`.`payment`
    join `sakila`.`film_actor`
    join `sakila`.`film`
    join `sakila`.`rental`
    join `sakila`.`inventory`
where (('sakila`.`rental`.`inventory_id` = `sakila`.`inventory`.`inventory_id`) and
      ('sakila`.`inventory`.`film_id` = `sakila`.`film`.`film_id`) and
      ('sakila`.`film`.`film_id` = `sakila`.`film_actor`.`film_id`) and
      ('sakila`.`film_actor`.`actor_id` = `sakila`.`actor`.`actor_id`) and
      ('sakila`.`rental`.`rental_id` = `sakila`.`payment`.`rental_id`))
group by concat('sakila`.`actor`.`first_name`, ' ', 'sakila`.`actor`.`last_name`)
order by `total_gain`;
```

```
sakila> create definer = root@localhost view actor_gain_money as
select concat(actor.first_name, ' ', actor.last_name) as actor_name, sum(amount) as total_gain from acto
where rental.inventory_id=inventory.inventory_id and inventory.film_id=film.film_id and film.film_id=
group by concat(actor.first_name, ' ', actor.last_name)
order by total_gain
[2022-12-05 11:59:44] completed in 21 ms
```



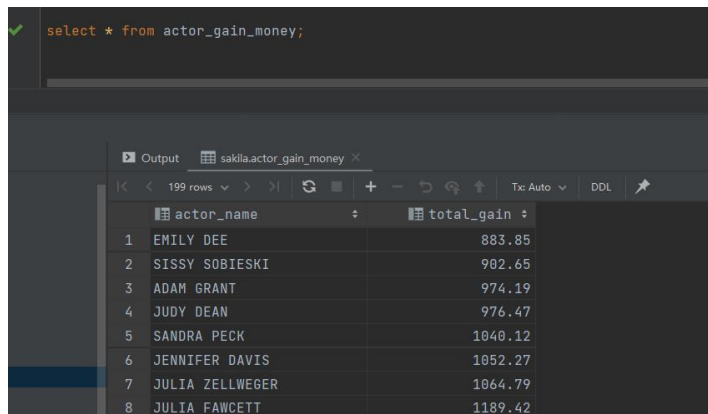
	actor_name	total_gain
1	EMILY DEE	883.85
2	SISSY SOBIESKI	902.65
3	ADAM GRANT	974.19
4	JUDY DEAN	976.47
5	SANDRA PECK	1040.12
6	JENNIFER DAVIS	1052.27
7	JULIA ZELLWEGER	1064.79
8	JULIA FAWCETT	1189.42
9	PENELOPE GUINNESS	1230.94
10	RUSSELL CLOSE	1251.05
11	KENNETH PESCI	1258.08
12	JEFF SILVERSTONE	1278.74
13	ED CHASE	1322.89
14	CAMERON WRAY	1327.97
15	CARY MCCONAUGHEY	1330.57
16	CATE HARRIS	1360.37
17	CUBA BIRCH	1370.38
18	CHRISTOPHER BERRY	1379.30
19	DAN STREEP	1380.12

这个 view 表的含义是每个演员演过的所有电影收获的所有租金

(2) 执行 select * from [视图名]，截图执行结果：

```
select * from actor_gain_money;
```

```
sakila> select * from actor_gain_money
[2022-12-05 12:11:11] 199 rows retrieved starting from 1 in 489 ms (execution: 456 ms, fetching: 33 ms)
```

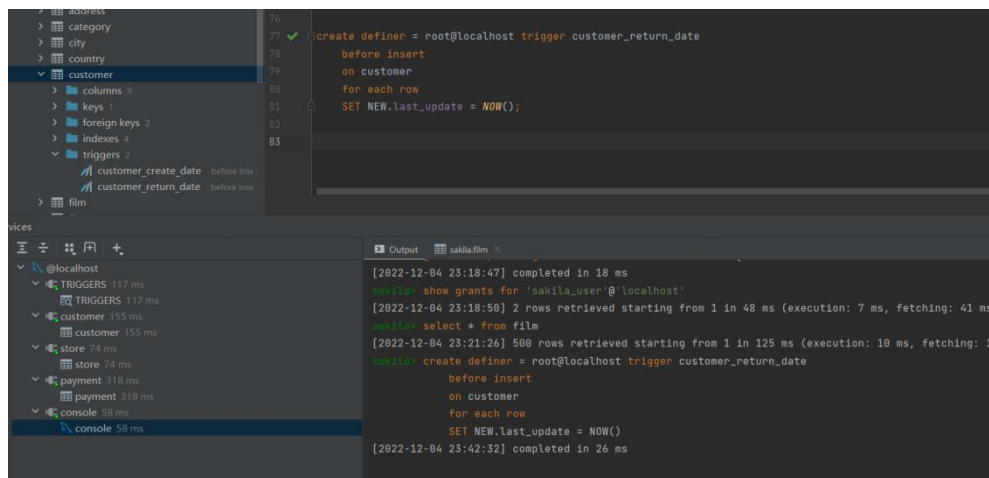


```
select * from actor_gain_money;
```

	actor_name	total_gain
1	EMILY DEE	883.85
2	SISSY SOBIESKI	982.65
3	ADAM GRANT	974.19
4	JUDY DEAN	976.47
5	SANDRA PECK	1040.12
6	JENNIFER DAVIS	1052.27
7	JULIA ZELLWEGER	1064.79
8	JULIA FAWCETT	1189.42

2. 设计 1 个触发器，需要体现触发器生效。

(1) 执行新建触发器的语句，并截图 SQL 和执行结果：



```
create definer = root@localhost trigger customer_return_date
before insert
on customer
for each row
set NEW.last_update = NOW();
```

Output:

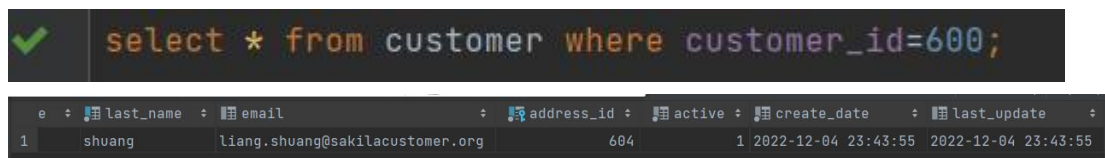
```
[2022-12-04 23:18:47] completed in 18 ms
sakila> show grants for 'sakila_user'@'localhost'
[2022-12-04 23:18:50] 2 rows retrieved starting from 1 in 48 ms (execution: 7 ms, fetching: 41 ms)
sakila> select * from film
[2022-12-04 23:21:26] 500 rows retrieved starting from 1 in 125 ms (execution: 10 ms, fetching: 115 ms)
sakila> create definer = root@localhost trigger customer_return_date
before insert
on customer
for each row
set NEW.last_update = NOW()
[2022-12-04 23:42:32] completed in 26 ms
```

(2) 验证触发器是否生效，截图验证过程：

先在 **customer** 表插入新数据

```
insert into customer values (600,1,'liang','shuang','liang.shuang@sakilacustomer.org',604,1,'2006-02-14 22:04:36','2006-02-15 04:57:20');
```

再查询一下



```
select * from customer where customer_id=600;
```

	customer_id	store_id	last_name	email	address_id	active	create_date	last_update
1	600	1	liang	shuang	liang.shuang@sakilacustomer.org	1	2022-12-04 23:43:55	2022-12-04 23:43:55

四、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？

使用外键，优点是：

保证数据的完整性和一致性

级联操作方便

将数据完整性判断托付给了数据库完成，减少了程序的代码量

缺点是：

外键一般会存在级联功能，级联更新，级联删除导致性能低下

使用外键，外键关联的数据查询要去另一张表，获取额外的锁，容易造成死锁

扩展性问题：触发器 外键 这种依赖于数据库本身的特性 可扩展性较差。 分库分表方便，在水平拆分和分库的情况下，外键是无法生效的。

辩证来看，可以存在，优点缺点都很明显，他能存在肯定是有道理和优势的，但是从现在的角度来看，外键不一定是必须存在的。