

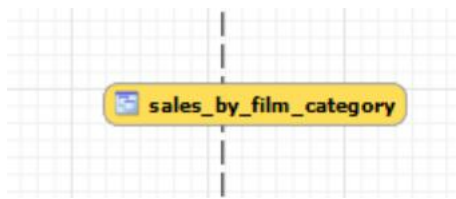
实验二报告

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？

view 在模型图中为黄色部分，如图：



7 个，分别为：

actor_info, customer_list, film_list, nicer_but_slower_film_list, sales_by_film_category, sales_by_store, staff_list

(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, actor, film_actor, category, film_category	列出所有演员信息：编号姓名以及出演的电影
film_list	category, film_category, film, film_actor, actor,	列出所有电影信息：编号，电影名，种类以及参演演员等
sales_by_store	payment, rental, inventory, store, address, city, country, staff	列出商店位置， 经理名， 以及商店销售情况

(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？

第一句：

1 17:59:09 update staff_list set 'zip code' = '518055' where ID = '1'

0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0

0.000 sec

第二句：

2 18:03:32 update film_list set price = 1.99 where FID = '1'

Error Code: 1288. The target table film_list of the UPDATE is not updatable

0.000 sec

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。这样的情况才可以 update。

而第二句 film_list 中含 group by:

```
FROM
((((('sakila`.`category`
LEFT JOIN `sakila`.`film_category` ON (('sakila`.`category`.`category_id` = `sakila`.`film_category`.`category_id`)))
LEFT JOIN `sakila`.`film` ON (('sakila`.`film_category`.`film_id` = `sakila`.`film`.`film_id`)))
JOIN `sakila`.`film_actor` ON (('sakila`.`film`.`film_id` = `sakila`.`film_actor`.`film_id`)))
JOIN `sakila`.`actor` ON (('sakila`.`film_actor`.`actor_id` = `sakila`.`actor`.`actor_id`)))
GROUP BY `sakila`.`film`.`film_id` , `sakila`.`category`.`name`
```

- (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';
```

该视图可更新，执行结果如下：

Result Grid			Filter Rows:	
	TABLE_NAME	IS_UPDATABLE		
▶	customer_list	YES		
	film_list	NO		
	nicer_but_slower_film_list	NO		
	staff_list	YES		
	sales_by_store	NO		
	sales_by_film_category	NO		
	actor_info	NO		

Action Output					Message		Duration / Fetch	
#	Time	Action						
1	21:56:39	SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakila' LIMIT 0, 1000	7 row(s) returned				0.016 sec / 0.000 sec	

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？

该触发器建在 customer 表中，这个触发器实现的功能为：向 customer 表加入新数据时，要将该元组的 create_date 属性值赋为当前时间。

- (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）

语句为：

```
insert into customer(customer_id,store_id,first_name,last_name,email,address_id,active) values (600,1,'Xinhao','Liu','lxh20020510@126.com',1,1);
```

```
insert into customer(customer_id,store_id,first_name,last_name,email,address_id,active) values
(600,1,'Xinhao','Liu','lxh20020510@126.com',1,1);
```

结果为：

```
9 22:20:22 insert into customer(customer_id,store_id,first_name,last_name,email,address_id,active) values (600,1,'Xinhao','Liu',1... 1 row(s) affected
```

customer 表中情况：

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
592	1	TERRANCE	ROUSH	TERRANCE.ROUSH@sakilacustomer.org	598	0	2006-02-14 22:04:37	2006-02-15 04:57:20
593	2	RENE	MCALISTER	RENE.MCALISTER@sakilacustomer.org	599	1	2006-02-14 22:04:37	2006-02-15 04:57:20
594	1	EDUARDO	HIATT	EDUARDO.HIATT@sakilacustomer.org	600	1	2006-02-14 22:04:37	2006-02-15 04:57:20
595	1	TERRENCE	GUNDER...	TERRENCE.GUNDERSON@sakilacustome...	601	1	2006-02-14 22:04:37	2006-02-15 04:57:20
596	1	ENRIQUE	FORSYTHE	ENRIQUE.FORSYTHE@sakilacustomer.org	602	1	2006-02-14 22:04:37	2006-02-15 04:57:20
597	1	FREDDIE	DUGGAN	FREDDIE.DUGGAN@sakilacustomer.org	603	1	2006-02-14 22:04:37	2006-02-15 04:57:20
598	1	WADE	DELVALLE	WADE.DELVALLE@sakilacustomer.org	604	1	2006-02-14 22:04:37	2006-02-15 04:57:20
599	2	AUSTIN	CINTRON	AUSTIN.CINTRON@sakilacustomer.org	605	1	2006-02-14 22:04:37	2006-02-15 04:57:20
600	1	Xinhao	Liu	lxh20020510@126.com	1	1	2022-11-30 22:20:22	2022-11-30 22:20:22
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

可见，该触发器已生效。

- (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

该语句作用是在插入 payment 表前，将 payment_date 用 now()赋值。若将该触发器移到 sakila-schema.sql 里去执行，在 payment_date 只能为建表时间，而不能获取插入元组的时间，无法实现该触发器的目标功能。

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	将 store_id 设置为主键
非空约束	约束 store_id,manager_staff_id,address_id,last_update 几个属性不能为空值
唯一约束	约束 manager_staff_id 属性的值不能重复
外键约束	约束 manager_staff_id 为外键，引用了 staff 表中的 staff_id 属性；约束 address_id 为外键，引用了 address 表中的 address_id 属性

- (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

#	Time	Action	Message
1	10:34:36	select user from mysql.user LIMIT 0, 1000	5 row(s) returned

ON UPDATE CASCADE（级联）: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，如果有也更新外键在子表（即包含外键的表）中的记录。

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）；
如图所示：

<pre>create user 'sakila_user' identified by '123456'</pre>			
12	10:30:24	create user 'sakila_user' identified by '123456'	0 row(s) affected

（2） 执行命令查看当前已有用户；
如图所示：

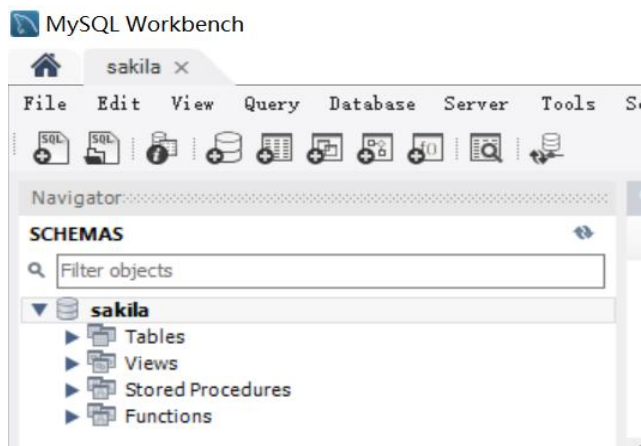
<pre>select user from mysql.user</pre>			
			
1	10:34:36	select user from mysql.user LIMIT 0, 1000	5 row(s) returned

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

<pre>1 • grant all privileges on sakila.* to 'sakila_user';</pre>			
<pre>2</pre>			
<div>Output</div> <div>Action Output</div>			
1	10:41:27	grant all privileges on sakila.* to 'sakila_user'	0 row(s) affected

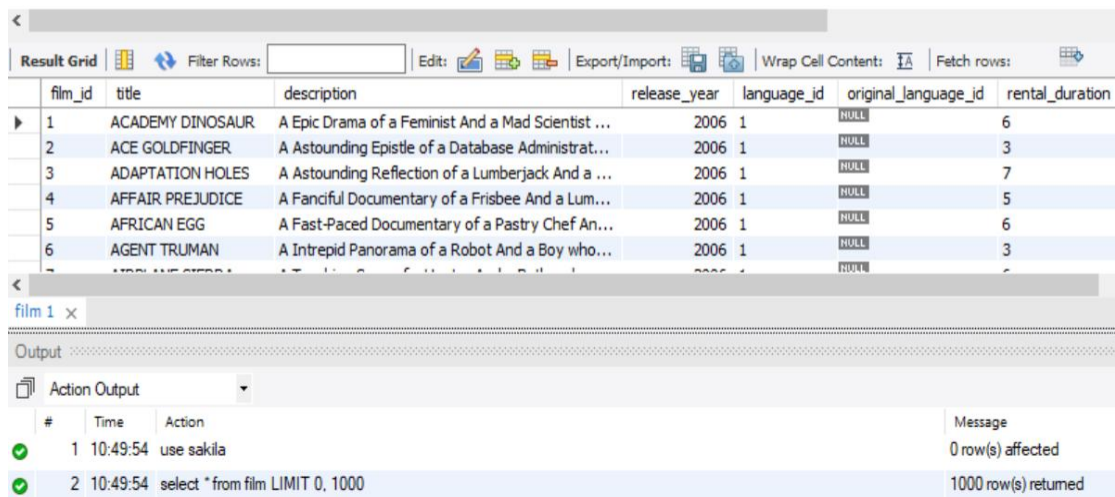
（4） 切换到 sakila_user 用户，执行 `select * from film` 操作。
如图所示，已完成切换到 sakila 用户：

MySQL Connections



并执行如下操作：

- 1 • `use sakila;`
- 2 • `select * from film;`



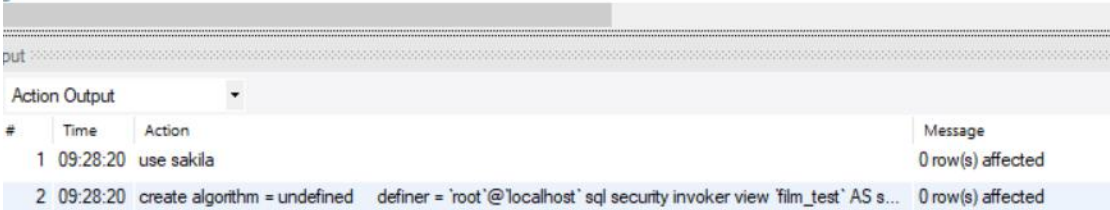
三、设计并实现

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：
- 如图所示：

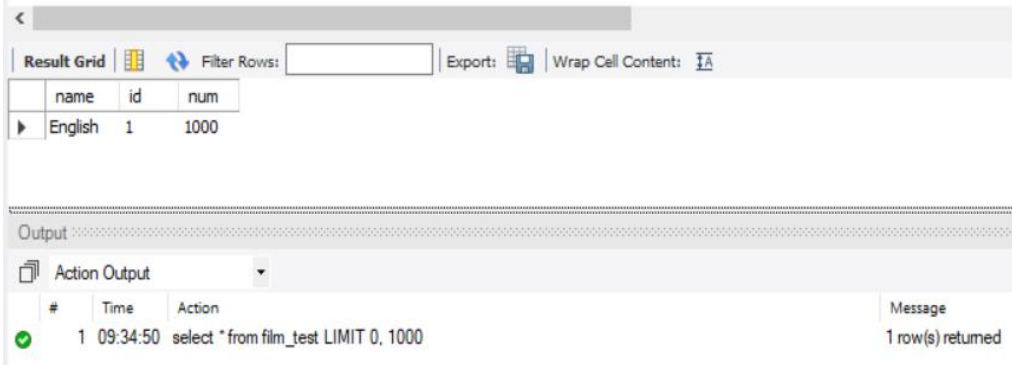
```
1 • use sakila;
2 • create
3     algorithm = undefined
4     definer = `root`@`localhost`
5     sql security invoker
6     view `film_test` AS
7     select
8         `l`.`name` AS `name`,
9         `l`.`language_id` AS `id`,
10        count(`f`.`film_id`) AS `num`
11    from (`language` `l` JOIN `film` `f` ON ((`l`.`language_id` = `f`.`language_id`)))
12    group by `l`.`language_id`
13
```



#	Time	Action	Message
1	09:28:20	use sakila	0 row(s) affected
2	09:28:20	create algorithm = undefined definer = `root`@`localhost` sql security invoker view `film_test` AS ...	0 row(s) affected

- (2) 执行 select * from [视图名]，截图执行结果：

```
1 • select * from film_test
2
```



name	id	num
English	1	1000

#	Time	Action	Message
1	09:34:50	select * from film_test LIMIT 0, 1000	1 row(s) returned

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：
- 如图：新建触发器 actor_update,实现当向 film_actor 表中插入数据时，将相应元组对应 actor 表中的 last_name 属性值改为"Liu"，即实现演员姓氏的更新。

```

1 • use sakila;
2 • CREATE
3   TRIGGER actor_update
4   after INSERT ON film_actor FOR EACH ROW
5   UPDATE actor SET last_name='Liu' WHERE
6   actor.actor_id=new.actor_id;
7
8

```

Output

#	Time	Action	Message
1	17:13:08	use sakila	0 row(s) affected
2	17:13:08	CREATE TRIGGER actor_update after INSERT ON film_actor FOR EACH ROW UPDATE actor SET last_name='Liu' WHERE ...	0 row(s) affected

(2) 验证触发器是否生效，截图验证过程：

原来的 actor 表如图所示：

	actor_id	first_name	last_name	last_update
	34	AUDREY	OLIVIER	2006-02-15 04:34:33
	35	JUDY	DEAN	2006-02-15 04:34:33
	36	BURT	DUKAKIS	2006-02-15 04:34:33
	37	VAL	BOLGER	2006-02-15 04:34:33
	38	TOM	MCKELLEN	2006-02-15 04:34:33
	39	GOLDIE	BRODY	2006-02-15 04:34:33
	40	JOHNNY	CASE	2006-02-15 04:34:33

运行 insert 语句：

```

1 • insert into film_actor(actor_id,film_id) values ('37','293')

```

Output

#	Time	Action	Message
1	17:17:18	insert into film_actor(actor_id,film_id) values ('37','293')	1 row(s) affected

actor 表也完成了相应的更新：

	actor_id	first_name	last_name	last_update
	34	AUDREY	OLIVIER	2006-02-15 04:34:33
	35	JUDY	DEAN	2006-02-15 04:34:33
	36	BURT	DUKAKIS	2006-02-15 04:34:33
	37	VAL	Liu	2022-12-01 17:17:18
	38	TOM	MCKELLEN	2006-02-15 04:34:33
	39	GOLDIE	BRODY	2006-02-15 04:34:33

可见 37 号演员姓氏已完成更改，触发器生效。

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？

原因：

1. 外键与级联适用于单机低并发，不适合分布式以及高并发的集群。
2. 级联更新是强阻塞，存在数据库更新风暴的风险。
3. 外键影响数据库的插入速度。

我认为外键有存在的必要，虽然外键对于阿里这种体量很大的公司会造成极大的数据使用上的风险。但是外键对于我们日常使用还是具有很大的方便性，我们可以扬长避短，外键有助于我们更好的设计数据库的结构，因此还是有存在的必要。