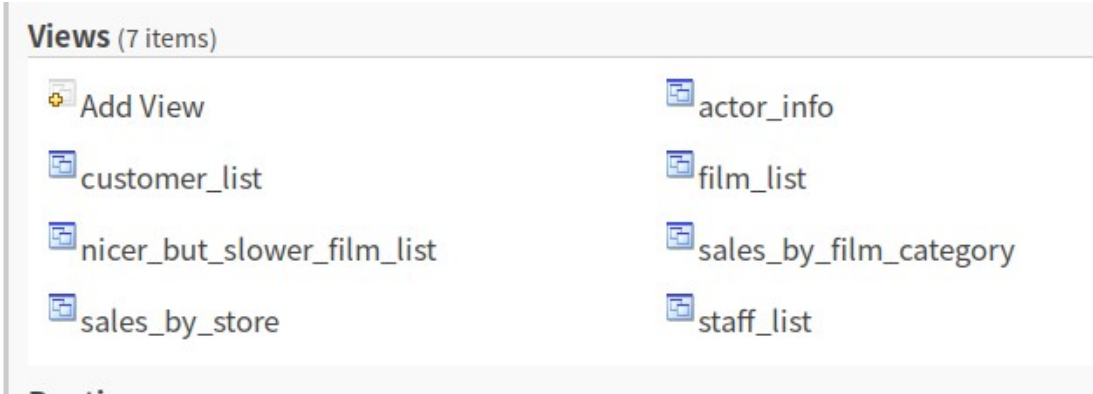


# 实验二报告

## 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?  
一共有 7 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,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。

- (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？

```

Database Changed
MariaDB [sakila]> update staff_list set `zip code` = '518055' where ID = '1';
Query OK, 0 rows affected (0.001 sec)
Rows matched: 1   Changed: 0   Warnings: 0

MariaDB [sakila]>

```

```

MariaDB [sakila]> update film_list set price = 1.99 where FID = '1';
ERROR 1288 (HY000): The target table film_list of the UPDATE is not updatable
MariaDB [sakila]>

```

观察 Select 得到的值，

8 • `select * from staff_list;`

9

10

#	ID	name	address	zip code	phone	city	country	SID
1	1	Mike Hillyer	23 Workhaven Lane	518055	14033335568	Lethbridge	Canada	1
2	2	Jon Stephens	1411 Lillydale Drive		6172235589	Woodridge	Australia	2

9 • `select * from film_list;`

10

#	FID	title	description	category	price	length	rating	actors
1	1	ACADEMY DINOSAUR	A Epic Drama of a Feminist...	Documentary	0.99	86	PG	PENELOPE GUINE
2	2	ACE GOLDFINGER	AAstounding Epistle of a D...	Horror	4.99	48	G	BOB FAWCETT, MI

以及对应的代码：

```

118 CREATE ALGORITHM = UNDEFINED DEFINER = `paul` @`localhost` SQL SECURITY DEFINER VIEW `sakila`.`staff_list` AS
119 select `s`.`staff_id` AS `ID`,
120        concat(`s`.`first_name`, _utf8mb4 ' ', `s`.`last_name`) AS `name`,
121        `a`.`address` AS `address`,
122        `a`.`postal_code` AS `zip code`,
123        `a`.`phone` AS `phone`,
124        `sakila`.`city`.`city` AS `city`,
125        `sakila`.`country`.`country` AS `country`,
126        `s`.`store_id` AS `SID`
127 from (
128     (
129         (
130             `sakila`.`staff` `s`
131             join `sakila`.`address` `a` on((`s`.`address_id` = `a`.`address_id`))
132         )
133         join `sakila`.`city` on((`a`.`city_id` = `sakila`.`city`.`city_id`))
134     )
135     join `sakila`.`country` on(
136         (
137             `sakila`.`city`.`country_id` = `sakila`.`country`.`country_id`
138         )
139     )
140 )

```

```

74 CREATE ALGORITHM = UNDEFINED DEFINER = `paul` @`localhost` SQL SECURITY DEFINER VIEW `sakila`.`film_list` AS
75 select `sakila`.`film`.`film_id` AS `FID`,
76 `sakila`.`film`.`title` AS `title`,
77 `sakila`.`film`.`description` AS `description`,
78 `sakila`.`category`.`name` AS `category`,
79 `sakila`.`film`.`rental_rate` AS `price`,
80 `sakila`.`film`.`length` AS `length`,
81 `sakila`.`film`.`rating` AS `rating`,
82 group_concat(
83   concat(
84     `sakila`.`actor`.`first_name`,
85     _utf8mb4 ' ',
86     `sakila`.`actor`.`last_name`
87   ) separator ','
88 ) AS `actors`
89 from (
90   (
91     (
92       `sakila`.`category`
93       left join `sakila`.`film_category` on(
94         (
95           `sakila`.`category`.`category_id` = `sakila`.`film_category`.`category_id`
96         )
97       )
98     )
99     left join `sakila`.`film` on(
100       (
101         `sakila`.`film_category`.`film_id` = `sakila`.`film`.`film_id`
102       )
103     )
104     join `sakila`.`film_actor` on(
105       (
106         `sakila`.`film`.`film_id` = `sakila`.`film_actor`.`film_id`
107       )
108     )
109     join `sakila`.`actor` on(
110       (
111         `sakila`.`film_actor`.`actor_id` = `sakila`.`actor`.`actor_id`
112       )
113     )
114   )
115   group by `sakila`.`film`.`film_id`,
116   `sakila`.`category`.`name`;
117 CREATE ALGORITHM = UNDEFINED DEFINER = `paul` @`localhost` SQL SECURITY DEFINER VIEW `sakila`.`staff_list` AS

```

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。

(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';

```

```

15 • SELECT table_name, is_updatable FROM information_schema.views
16 WHERE table_schema = 'sakila';

```

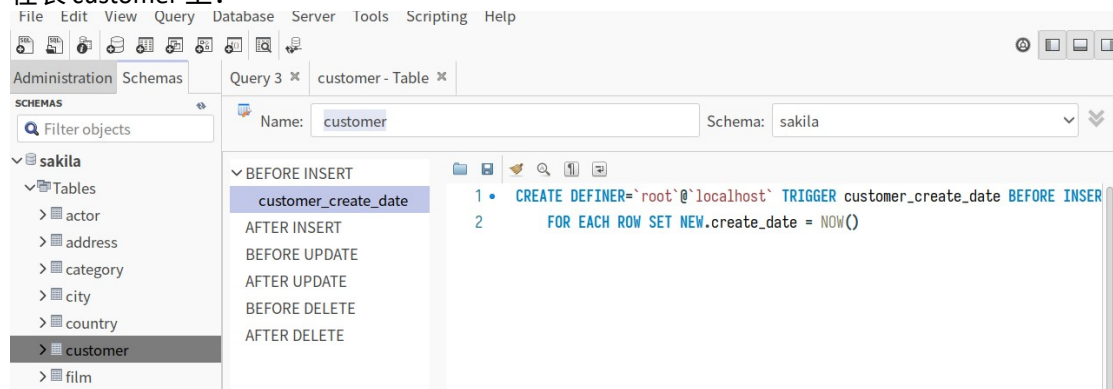
#	table_name	is_updatable
1	film_list	NO
2	sales_by_store	NO
3	nicer_but_slower_film_list	NO
4	staff_list	YES
5	actor_info	NO
6	customer_list	YES
7	sales_by_film_category	NO

views 7 ✕

## 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？

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。

- (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）

```

18 • INSERT INTO `sakila`.`customer` (
19     `store_id`,
20     `first_name`,
21     `last_name`,
22     `email`,
23     `address_id`,
24     `active`
25 )
26 VALUES (
27     1,
28     "Chiro",
29     "Liang",
30     "Chiro2001@163.com",
31     5,
32     1
33 );
34 • select * from customer WHERE first_name = "Chiro";

```

#	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	606	1	Chiro	Liang	Chiro2001@163.com	5	1	2022-12-03 01:39:41	2022-12-03 01:39:41

触发器正确生效了。

- (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，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。

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）

Query 3	customer - Table	SQL File 3	store - Table
Name: store		Schema: sakila	
Column Name	Datatype	PK	NN
store_id	TINYINT(3)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
manager_staff_id	TINYINT(3)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
address_id	SMALLINT(5)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
last_update	TIMESTAMP	<input type="checkbox"/>	<input checked="" type="checkbox"/>
		CURRENT_TIMESTAMP() ON UPDATE CURRENT_TIMESTAMP()	

```
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

(2) 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：

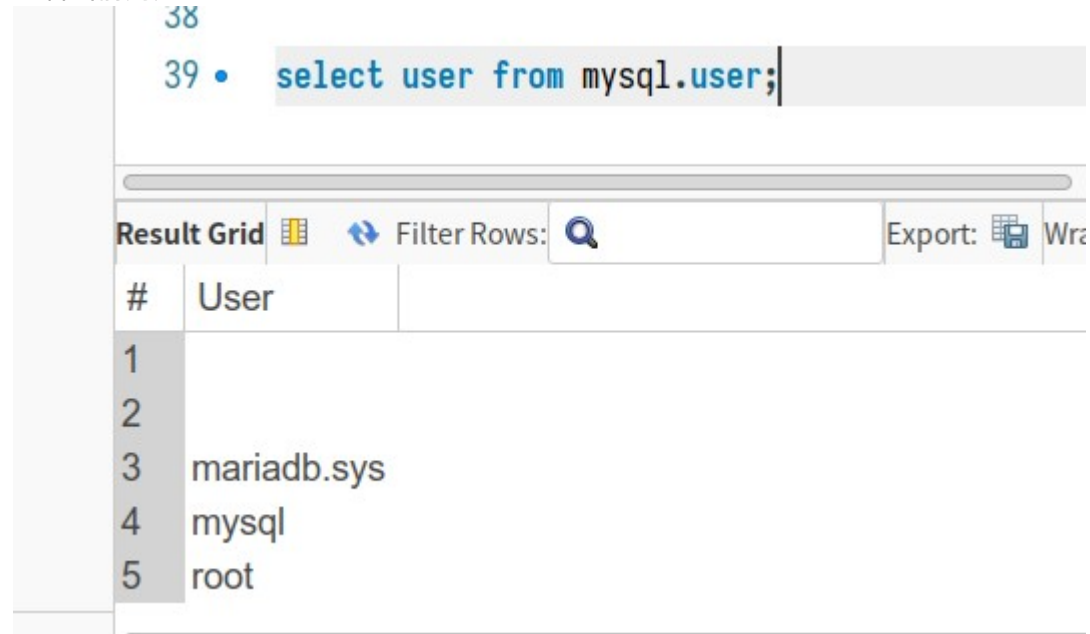
- 1. ON UPDATE：在更新时执行操作
- 2. CASCADE：当在父表中更新对应记录的时候，首先检查该记录是否有对应的外键，如果有则同时更新外键在子表中的记录

## 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）；

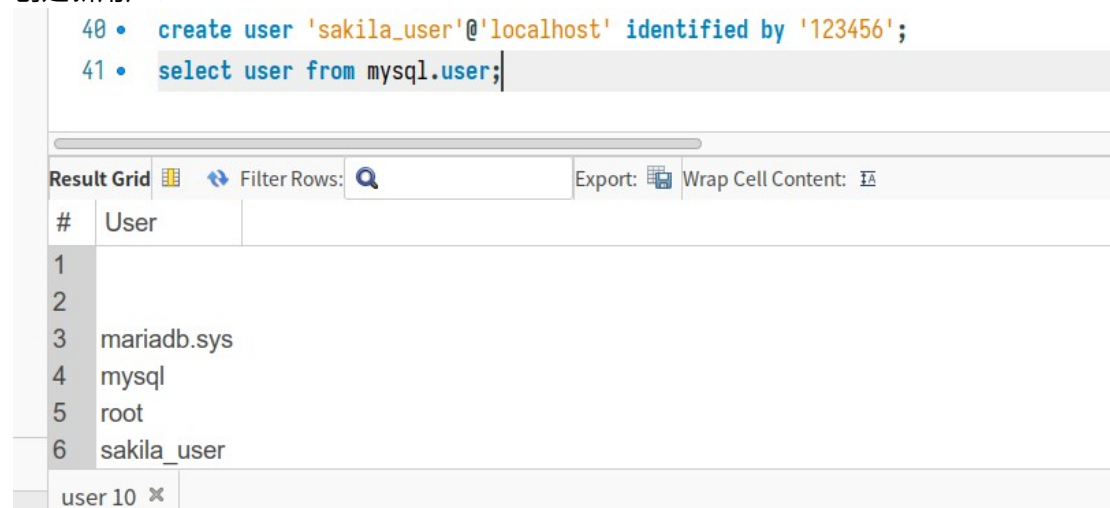
查看当前用户：



The screenshot shows a SQL command being executed in a terminal window. The command is `select user from mysql.user;`. Below the command, a result grid is displayed with the following data:

#	User
1	
2	
3	mariadb.sys
4	mysql
5	root

创建新用户：



The screenshot shows two SQL commands being executed in a terminal window. The first command is `create user 'sakila_user'@'localhost' identified by '123456';`. The second command is `select user from mysql.user;`. Below the commands, a result grid is displayed with the following data:

#	User
1	
2	
3	mariadb.sys
4	mysql
5	root
6	sakila_user

At the bottom of the terminal window, there is a tab labeled "user 10 x".

(2) 执行命令查看当前已有用户；



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

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

#	User
1	
2	
3	mariadb.sys
4	mysql
5	root
6	sakila_user
user 10 ✕	

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

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

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

#	Grants for sakila_user@localhost
1	GRANT USAGE ON *.* TO ...

44 • show grants for 'sakila_user'@'localhost';	
45	
46 • grant all privileges on sakila.* to 'sakila_user'@'localhost';	
Query Completed	

(4) 切换到 sakila\_user 用户，执行 select \* from film 操作。

989	WORKING MICROCOSMOS	2.99	139	28.99	R	Trailers,Commentaries,Behind the Scenes	2006-02-15 05:03:42	2006
990	WORLD LEATHERNECKS	4.99	74	22.99	R	Commentaries,Deleted Scenes	2006-02-15 05:03:42	2006
991	WORST BANGER	0.99	171	13.99	PG-13	Trailers,Behind the Scenes	2006-02-15 05:03:42	2006
992	WRATH MILE	2.99	185	26.99	PG	Deleted Scenes,Behind the Scenes	2006-02-15 05:03:42	2006
993	WRONG BEHAVIOR	0.99	176	17.99	NC-17	Trailers,Commentaries	2006-02-15 05:03:42	2006
994	WYOMING STORM	2.99	178	18.99	PG-13	Trailers,Behind the Scenes	2006-02-15 05:03:42	2006
995	YENTL IDAHO	4.99	108	29.99	PG-13	Deleted Scenes	2006-02-15 05:03:42	2006
996	YOUNG LANGUAGE	0.99	183	9.99	G	Trailers,Behind the Scenes	2006-02-15 05:03:42	2006
997	YOUTH KICK	4.99	86	11.99	R	Trailers,Commentaries,Deleted Scenes	2006-02-15 05:03:42	2006
998	ZHIVAGO CORE	0.99	179	14.99	NC-17	Trailers,Behind the Scenes	2006-02-15 05:03:42	2006
999	ZOOLANDER FICTION	0.99	165	18.99	NC-17	Deleted Scenes	2006-02-15 05:03:42	2006
1000	ZORRO ARK	2.99	181	28.99	R	Trailers,Deleted Scenes	2006-02-15 05:03:42	2006
1		4.99	58	18.99	NC-17	Trailers,Commentaries,Behind the Scenes	2006-02-15 05:03:42	2006

1000 rows in set (0.001 sec)

MariaDB [sakila]> 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：

```
174
195 |-- view: address_in_city
196 • CREATE ALGORITHM = UNDEFINED DEFINER = `root` @`localhost` SQL SECURITY DEFINER VIEW `sakila`.`sales_in_country` AS
197 select country.country as `country`, COUNT(customer.customer_id) as `customers`, SUM(payment.amount) as `sales`,
198        SUM(payment.amount) / COUNT(customer.customer_id) as `ave`
199 from country
200      join city on country.country_id=city.country_id
201      join address on city.city_id=address.city_id
202      join customer on customer.address_id=address.address_id
203      join payment on customer.customer_id=payment.customer_id
204 group by city.country_id
205 order by SUM(payment.amount) desc;
```

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。

(2) 执行 select \* from [视图名]，截图执行结果：

```
206 • select * from sales_in_country;
```

Result Grid					Filter Rows:	Export:	Wrap Cell Content:
#	country	customers	sales	ave			
1	India	1573	6630.27	4.215048			
2	China	1427	5802.73	4.066384			
3	United States	968	4110.32	4.246198			
4	Japan	826	3471.74	4.203075			
5	Mexico	796	3307.04	4.154573			
6	Brazil	748	3200.52	4.278770			
7	Russian Federation	713	3045.87	4.271907			
8	Philippines	568	2381.32	4.192465			
9	Turkey	388	1662.12	4.283814			
10	Nigeria	352	1511.48	4.293977			
11	Indonesia	307	1510.33	4.115041			

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：

```
208 • use sakila;
209
210 DELIMITER ;;
211 • CREATE TRIGGER `ins_customer` BEFORE INSERT ON `customer` FOR EACH ROW BEGIN
212     SET new.active = 1;
213 END;;
214
```

```
MariaDB [sakila]> CREATE TRIGGER `ins_customer` BEFORE INSERT ON `customer` FOR EACH ROW BEGIN SET new.active = 1; END
;;
Query OK, 0 rows affected (0.010 sec)
```

本触发器用于强制使插入行的 active 字段设置为 1。

(2) 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：

```
Database changed
MariaDB [sakila]> INSERT INTO customer (
-> store_id, first_name, last_name, email, address_id, active
-> ) VALUES (
-> (SELECT store_id FROM store ORDER BY store_id DESC LIMIT 1),
-> "Chiro2",
-> "Liang2",
-> "Chiro2001@163.com",
-> (SELECT address_id FROM address ORDER BY address_id DESC LIMIT 1),
-> 0
-> );
Query OK, 1 row affected (0.005 sec)
```

检索这条数据：

```
MariaDB [sakila]> SELECT * FROM customer WHERE first_name="Chiro2";
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| customer_id | store_id | first_name | last_name | email | address_id | active | create_date | last_update |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 609 | 2 | Chiro2 | Liang2 | Chiro2001@163.com | 605 | 1 | 2022-12-06 14:47:28 | 2022-12-06 14:47:28 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)
```

可以看到 active = 1，触发器已经生效。

## 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？

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。