实验二报告

一、 观察并回答问题

1. 观察 sakila.mwb 并回答问题

（1） 图中共有几个 View？

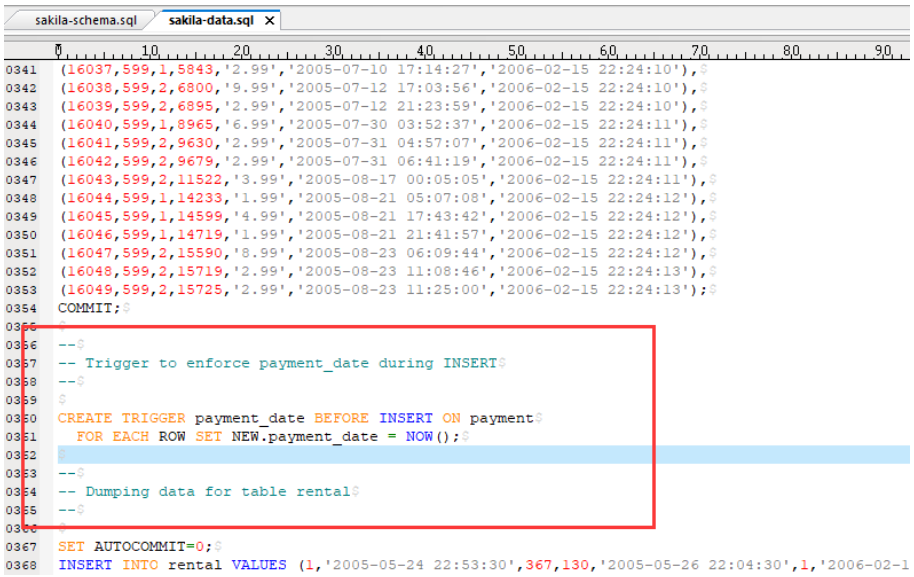
图中共 7 个 View。

（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	Actor,film_actor,film_category,category	按演员划分，展示每个演员所演的电影信息
film_list	film,film_actor,film_category,category	按电影划分，展示每个电影的描述、类别、价格、时长、演员、影片分级。
sales_by_store	Payment,rental,inventory,store,address,city,country,staff	展示每个店铺的店铺信息，经理，以及总销售额

2. 观察 sakila-schema.sql 和 sakila-data.sql 并回答问题

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的语句：



请同学们思考，这个触发器是否可以移到 sakila-schema.sql 里去执行？为什么？

不能，因为需要先将数据导入数据库后再设置 trigger，否则在运行 sakila_data.sql 进行插入数据时，要插入到 payment 中的数据的 payment_date 属性均会被设置为 NOW()。

3. 观察数据库的触发器 customer_create_date 并回答问题

（1） customer_create_date 触发器建在哪个表上？

该触发器创建在 customer 表上

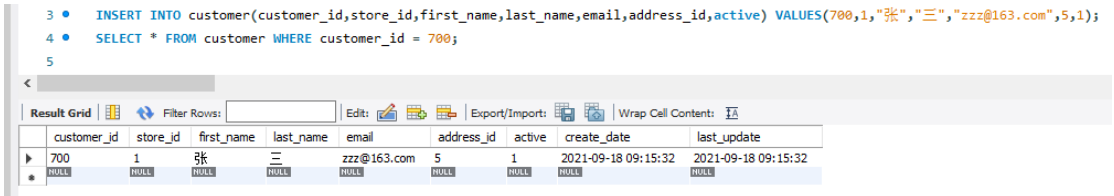
（2） 这个触发器实现什么功能？

在向 customer 插入一条数据时，将创建日期设置为当前时间。

（3） 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(customer_id,store_id,first_name,last_name,email,address_id, active) VALUES(700,1,"张","三","zzz@163.com",5,1);

SELECT * FROM customer WHERE customer_id = 700;



可见 create_date 被设置为插入该数据时的时间。

二、 设计并实现

根据应用场景，为 Sakila 数据库合理地设计并实现：

（注意：请将创建语句、调用结果截图记录到实验报告里）

1. 设计 1 个视图，至少关联 3 个表；

创建视图 customer_info 展示顾客的 customer_id 号以及其姓名，展示其租借的总花费和租借的所有影片名称，SQL 代码及结果截图如下：

```
CREATE VIEW customer_info AS
SELECT customer.customer_id CID,
CONCAT(customer.first_name,' ',customer.last_name) CNAME,
SUM(payment.amount) TOTAL_SPEND,
GROUP_CONCAT(CONCAT(film.title) SEPARATOR ',') film_info
FROM customer,payment,rental,inventory,film
WHERE rental.rental_id = payment.rental_id AND rental.inventory_id = inventory.inventory_id
AND film.film_id = inventory.film_id AND customer.customer_id = rental.customer_id
GROUP BY customer.customer_id;
```

进行查询：SELECT * FROM customer_info;

CID	CNAME	TOTAL_SPEND	film_info
1	MARY SMITH	118.68	TALENTED HOMICIDE,UNFORGIVEN ZOOLANDE...
2	PATRICIA JOHNSON	128.73	FANTASY TROOPERS,FORWARD TEMPLE,HOLI...
3	LINDA WILLIAMS	135.74	ANACONDA CONFESSIONS,BIRDS PERDITION,...
4	BARBARA JONES	81.78	TOMORROW HUSTLER,VERTIGO NORTHWEST,...
5	ELIZABETH BROWN	144.62	FIGHT JAWBREAKER,FREAKY POCUS,GABLES ...
6	JENNIFER DAVIS	93.72	AMADEUS HOLY,BLUES INSTINCT,CLUB GRAFFI...
7	MARIA MILLER	151.67	STRANGERS GRAFFITI,STREETCAR INTENTION...
8	SUSAN WILSON	92.76	FORREST SONS,HAROLD FRENCH,HIGH ENCIN...
9	MARGARET MOORE	89.77	BLOOD ARGONAUTS,BORN SPINAL,COLOR PHI...
10	DOROTHY TAYLOR	99.75	VIETNAM SMOOCHY,WEST LION,WHISPERER G...
11	LISA ANDERSON	106.76	FANTASIA PARK,FARGO GANDHI,GORGEOUS B...
12	NANCY THOMAS	103.72	ARMY FLINTSTONES,BEVERLY OUTLAW,BRIDE ...
13	KAREN JACKSON	131.73	SUNRISE LEAGUE,TROUBLE DATE,MISSION ZO...

customer_info 26 ×

2. 设计 1 个触发器，需要在报告里体现触发器生效。

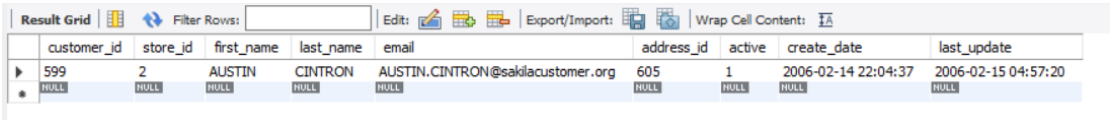
设计该触发器实现在 customer 中删除一个顾客信息时，将 rental 和 payment 表中对应的 customer_id 有关的元组删除。代码如下：

```
delimiter $$
CREATE TRIGGER customer_del BEFORE DELETE ON customer
    FOR EACH ROW BEGIN
        DELETE FROM rental WHERE rental.customer_id = old.customer_id;
        DELETE FROM payment WHERE payment.customer_id = old.customer_id;
END$$
```

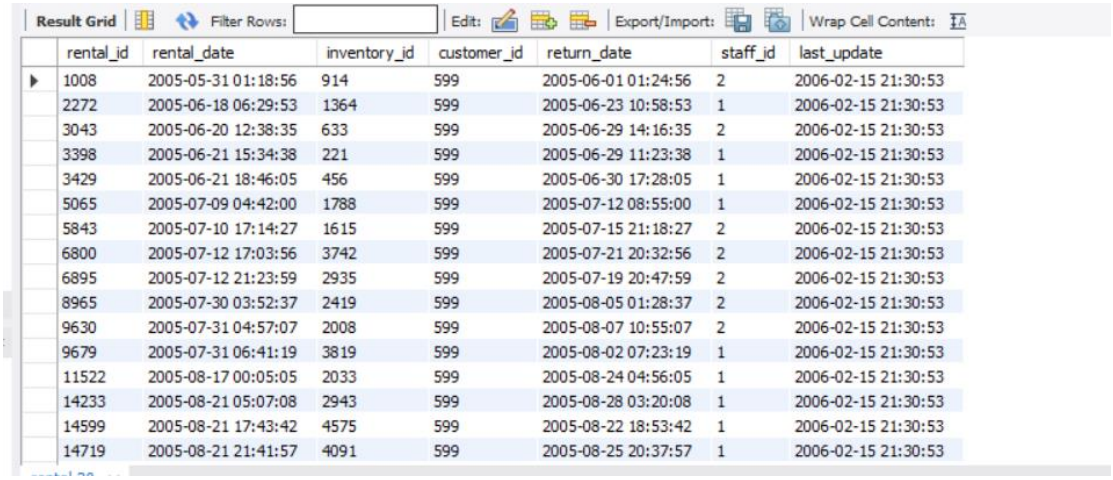
删除之前进行查询：

```
SELECT * FROM customer WHERE customer_id = 599;
SELECT * FROM rental WHERE rental.customer_id = 599;
SELECT * FROM payment WHERE payment.customer_id = 599;
```

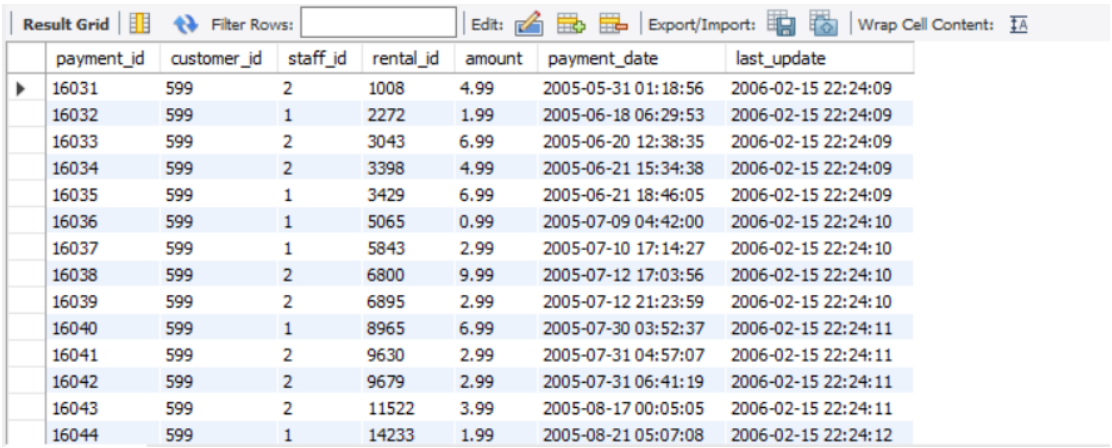
结果如图：



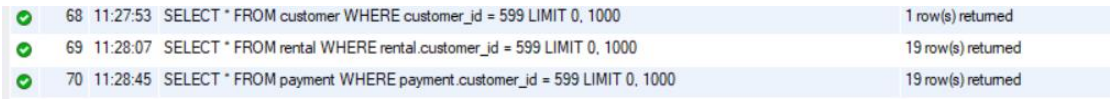
customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
599	2	AUSTIN	CINTRON	AUSTIN.CINTRON@sakilacustomer.org	605	1	2006-02-14 22:04:37	2006-02-15 04:57:20



rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
1008	2005-05-31 01:18:56	914	599	2005-06-01 01:24:56	2	2006-02-15 21:30:53
2272	2005-06-18 06:29:53	1364	599	2005-06-23 10:58:53	1	2006-02-15 21:30:53
3043	2005-06-20 12:38:35	633	599	2005-06-29 14:16:35	2	2006-02-15 21:30:53
3398	2005-06-21 15:34:38	221	599	2005-06-29 11:23:38	1	2006-02-15 21:30:53
3429	2005-06-21 18:46:05	456	599	2005-06-30 17:28:05	1	2006-02-15 21:30:53
5065	2005-07-09 04:42:00	1788	599	2005-07-12 08:55:00	1	2006-02-15 21:30:53
5843	2005-07-10 17:14:27	1615	599	2005-07-15 21:18:27	2	2006-02-15 21:30:53
6800	2005-07-12 17:03:56	3742	599	2005-07-21 20:32:56	2	2006-02-15 21:30:53
6895	2005-07-12 21:23:59	2935	599	2005-07-19 20:47:59	2	2006-02-15 21:30:53
8965	2005-07-30 03:52:37	2419	599	2005-08-05 01:28:37	2	2006-02-15 21:30:53
9630	2005-07-31 04:57:07	2008	599	2005-08-07 10:55:07	2	2006-02-15 21:30:53
9679	2005-07-31 06:41:19	3819	599	2005-08-02 07:23:19	1	2006-02-15 21:30:53
11522	2005-08-17 00:05:05	2033	599	2005-08-24 04:56:05	1	2006-02-15 21:30:53
14233	2005-08-21 05:07:08	2943	599	2005-08-28 03:20:08	1	2006-02-15 21:30:53
14599	2005-08-21 17:43:42	4575	599	2005-08-22 18:53:42	1	2006-02-15 21:30:53
14719	2005-08-21 21:41:57	4091	599	2005-08-25 20:37:57	1	2006-02-15 21:30:53



payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
16031	599	2	1008	4.99	2005-05-31 01:18:56	2006-02-15 22:24:09
16032	599	1	2272	1.99	2005-06-18 06:29:53	2006-02-15 22:24:09
16033	599	2	3043	6.99	2005-06-20 12:38:35	2006-02-15 22:24:09
16034	599	2	3398	4.99	2005-06-21 15:34:38	2006-02-15 22:24:09
16035	599	1	3429	6.99	2005-06-21 18:46:05	2006-02-15 22:24:09
16036	599	1	5065	0.99	2005-07-09 04:42:00	2006-02-15 22:24:10
16037	599	1	5843	2.99	2005-07-10 17:14:27	2006-02-15 22:24:10
16038	599	2	6800	9.99	2005-07-12 17:03:56	2006-02-15 22:24:10
16039	599	2	6895	2.99	2005-07-12 21:23:59	2006-02-15 22:24:10
16040	599	1	8965	6.99	2005-07-30 03:52:37	2006-02-15 22:24:11
16041	599	2	9630	2.99	2005-07-31 04:57:07	2006-02-15 22:24:11
16042	599	2	9679	2.99	2005-07-31 06:41:19	2006-02-15 22:24:11
16043	599	2	11522	3.99	2005-08-17 00:05:05	2006-02-15 22:24:11
16044	599	1	14233	1.99	2005-08-21 05:07:08	2006-02-15 22:24:12



68	11:27:53	SELECT * FROM customer WHERE customer_id = 599 LIMIT 0, 1000	1 row(s) returned
69	11:28:07	SELECT * FROM rental WHERE rental.customer_id = 599 LIMIT 0, 1000	19 row(s) returned
70	11:28:45	SELECT * FROM payment WHERE payment.customer_id = 599 LIMIT 0, 1000	19 row(s) returned

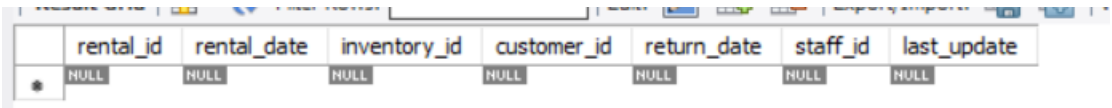
删除语句：

DELETE FROM customer WHERE customer_id = 599;

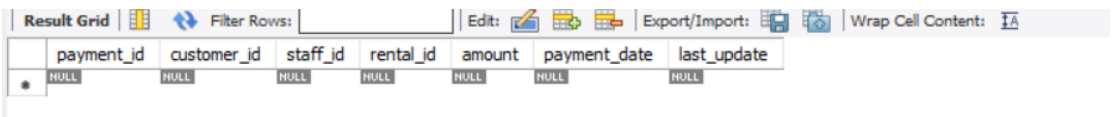
71	11:29:46	DELETE FROM customer WHERE customer_id = 599	1 row(s) affected	0.031 sec

再次查询，结果均为空，触发器触发成功：

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update
NULL	NULL	NULL	NULL	NULL	NULL	NULL



payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
NULL	NULL	NULL	NULL	NULL	NULL	NULL

72	11:30:09	SELECT * FROM customer WHERE customer_id = 599 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
73	11:30:18	SELECT * FROM rental WHERE rental.customer_id = 599 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
74	11:30:29	SELECT * FROM payment WHERE payment.customer_id = 599 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

三、 思考题

（这部分不是必做题，供有兴趣的同学思考）

如果可以给这个数据库做修改，你想修改哪个地方？指出并说明原因。

增加每部影片租借时长所需要的价格表，当插入一条新的租借信息时，根据该条租借信息(return_date-rental_date)*价格自动向 payment 表里添加数据。