

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?
有 7 个
- (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 film_category category	可以快速查询到一个演员的基本信息（ID、姓名）和所有参演过的电影名称及其种类。
film_list	film film_actor actor film_category category	可以快速查询到一部电影的基本信息和其所对应的类别名称，以及所有参演的演员姓名。
sales_by_film_category	category film_category film inventory rental payment	可以快速查询到每个类别名称的电影所获得的总租金之和。

(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？

The screenshot shows a SQL IDE window titled 'query1' with a tab for 'staff_list'. The SQL editor contains the following queries:

```
1 • Use sakila;
2 • Update staff_list
3     Set `zip code` = '518055'
4     Where ID = '1';
5 • Select *
6     From staff_list;
```

Below the editor, the 'Result Grid' displays the data from the staff_list view:

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

The 'Output' pane shows the execution results:

#	Time	Action	Message
1	18:39:20	Use sakila	0 row(s) affected
2	18:39:20	Update staff_list Set `zip code` = '518055' Where ID = '1'	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
3	18:39:20	Select * From staff_list LIMIT 0, 50000	2 row(s) returned

The screenshot shows a SQL IDE window titled 'query1' with tabs for 'staff_list' and 'film_list'. The SQL editor contains the following queries:

```
1 • Use sakila;
2 • Update film_list
3     Set price = 1.99
4     Where FID = '1';
5 • Select *
6     From film_list;
```

The 'Output' pane shows the execution results:

#	Time	Action	Message
1	18:43:13	Use sakila	0 row(s) affected
2	18:43:13	Update film_list Set price = 1.99 Where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is not updatable

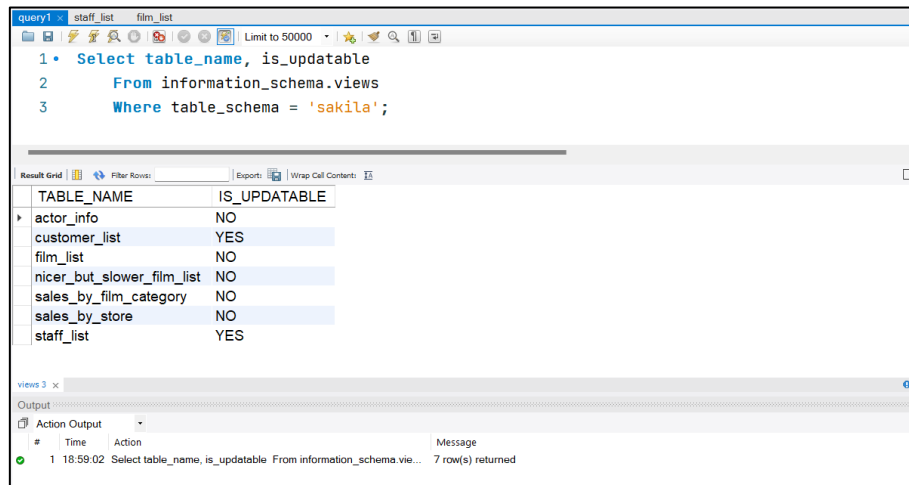
如上图，第一个更新语句运行成功，第二个运行失败，查看视图创建语句可知，其使用了 Group By 语句，因此无法更新。

如果视图是从单个基本表使用选择、投影操作导出的，并且包含了基本表的主键，则可以使用 Update 语句更新。

如果视图定义中使用了多个基本表或使用了 Join、Group By、Union 等语句，或查询中使用了 Distinct、Unique 等限定词，或目标列使用了聚集函数或算数表达式，或来自单个基本表但未包含主键，则不可使用 Update 语句更新。

- (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

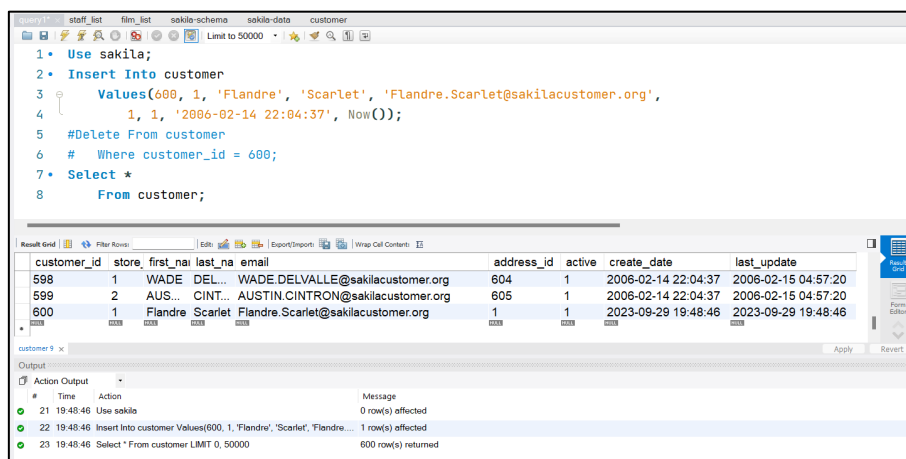
2. 关于触发器

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

Trigger 定义如下：

```
--
-- Trigger to enforce create dates on INSERT
--
CREATE TRIGGER customer_create_date BEFORE INSERT ON customer
FOR EACH ROW SET NEW.create_date = NOW();
```

该触发器建在 customer 表上，功能为在向 customer 表中插入记录前，将其 create_date 设为当前时间。



customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
598	1	WADE	DEL...	WADE.DELVALLE@sakilacustomer.org	604	1	2006-02-14 22:04:37	2006-02-15 04:57:20
599	2	AUS...	CINT...	AUSTIN.CINTRON@sakilacustomer.org	605	1	2006-02-14 22:04:37	2006-02-15 04:57:20
600	1	Flandre	Scarlet	Flandre.Scarlet@sakilacustomer.org	1	1	2023-09-29 19:48:46	2023-09-29 19:48:46

运行结果如上图，在 Insert 语句中，将 create_time 设为 '2006-02-14 22:04:37'，但注意到查询结果中，create_date 属性值为 '2023-9-29 19:48:46'，由此可见此触发器有效。

- (2) 触发器 `upd_film` 建在哪个表上？这个触发器实现什么功能？在这个表上修改一条数据的 `description` 字段，验证一下触发器是否生效。（截图语句和执行结果）
Trigger 定义如下：

```
220 CREATE TRIGGER `upd_film` AFTER UPDATE ON `film` FOR EACH ROW BEGIN
221 IF (old.title != new.title) OR (old.description != new.description) OR (old.film_id != new.film_id)
222 THEN
223     UPDATE film_text
224     SET title=new.title,
225         description=new.description,
226         film_id=new.film_id
227     WHERE film_id=old.film_id;
228 END IF;
229 END;;
```

该触发器建在 `film` 表上，功能为在向更新 `film` 表中的记录后，检测 `title`、`description`、`film_id` 这 3 条属性是否发生变化，如果有变化，则在 `film_text` 视图中同步更新这些变化。

The screenshot shows a MySQL query window with the following SQL code:

```
1. Use sakila;
2. Update film
3.   Set description = 'Test'
4.   Where film_id = 1;
5. #Delete From customer
6. # Where customer_id = 600;
7. Select *
8.   From film_text;
```

The results pane shows the output of the query:

film_id	title	description
1	ACADEMY DINOSAUR	Test
2	ACE GOLDFINGER	A Astounding Epistle of a ...
3	ADAPTATION HOLES	A Astounding Reflection of...
4	AFFAIR PREJUDICE	A Fanciful Documentary of...
5	AFRICAN EGG	A Fast Paced Documentary

The output pane shows the execution details:

#	Time	Action	Message
28	20:08:53	Use sakila	0 row(s) affected
29	20:08:53	Update film Set description = 'Test' Where film_id = 1	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
30	20:08:53	Select * From film_text LIMIT 0, 50000	1000 row(s) returned

运行结果如上图，可见，修改 `film_id=1` 的记录的 `description` 后，`film_text` 视图中的 `description` 也相应地更新，因此该触发器有效。

- (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？

The screenshot shows a MySQL query window with the following SQL code:

```
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
```

不可以移到 sakila-schema.sql 中执行，因为如果在 sakila-schema.sql 执行过程中就创建了这个触发器，那么在执行 sakila-data.sql 写入数据时，其会受到该触发器约束，于是 payment 表中的 payment_date 都会被改写为 sakila-data.sql 这个文件实际运行的时间，而不是提前写好的 payment_date。

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	唯一记录一列数据属性 store_id
非空约束	确保一列数值不为空 last_update
唯一约束	保证一列数值数据唯一 address_id

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

On Delete Restrict: 如果试图操作删除某一行数据，而这一行的键被其他表外键引用，中止该操作。

On Update Cascade: 如果试图操作更新某一行中的键值，而这一行的键值被其他表的外键所引用，则更新组成外键的所有值为该键最新的值。

4. 关于存储过程

- (1) 这个存储过程 rewards_report 实现了什么功能？输出参数 count_rewardees 是什么？

这个存储过程有两个输入 min_monthly_purchases, min_dollar_amout_purchased，一个输出 count_rewardees，其功能为：统计月购买数量大于 min_monthly_purchases，且月消费金额大于 min_dollar_amout_purchased 的顾客数量。

输出参数 count_rewardees 即为上述的顾客数量统计结果。

- (2) 图中第 483 行的 NOT DETERMINISTIC 和第 485 行的 SQL SECURITY DEFINER 分别是什么含义？

“NOT DETERMINISTIC”：是一个用来描述存储过程的属性，表示存储过程的执行结果在多次调用时可能不确定。换句话说，存储过程的结果可能因为外部因素的改变而发生变化。这个属性通常用于具有随机性或依赖外部因素的存储过程。

“SQL SECURITY DEFINER”：这也是一个用来描述存储过程的属性，指定了存储过程的执行权限。具体来说，当存储过程被定义为 “SQL SECURITY DEFINER” 时，它将使用定义时的执行权限进行执行，而不是调用它的用户的执行权限。这样做可以确保存储过程以定义者的权限执行，而不会受到调用者权限的限制。

5. 关于函数

- (1) 这个函数 get_customer_balance 实现了什么功能？返回值是什么？

这个函数用于计算客户的租赁费用(v_rentfees)、逾期费用(v_overfees)和付款总额(v_payments),从而确定客户在特定日期之前的账户余额(租赁费用+逾期费用-付款总额)。**返回值**为指定的用户在指定的日期之前的账户余额 (5 位整数精度,保留两位小数)。

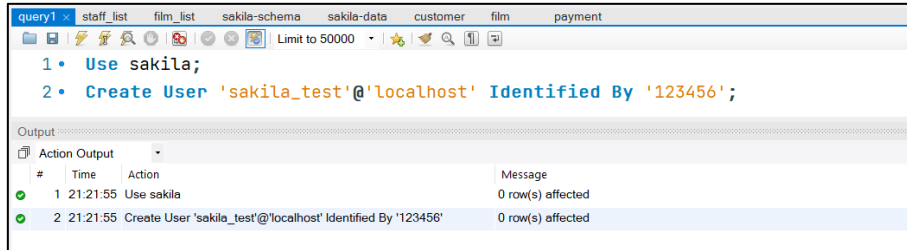
(2) 这个函数体中用到了 3 个函数,是哪几个函数?这 3 个函数的作用分别是?

函数	作用
SUM()	计算一组数值或表达式的和
TO_DAYS()	将一个日期对象转换为整型,用于数值计算
IFNULL(expr1, expr2)	判断 expr1 是否为 null,如果不为 null 则返回 expr1 的值,如果为 null 则返回 expr2 的值

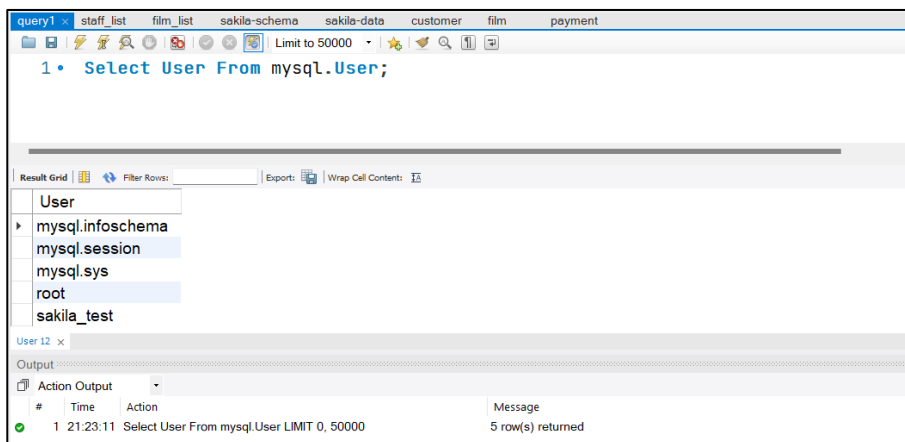
二、创建新用户并分配权限

(截图语句和执行结果)

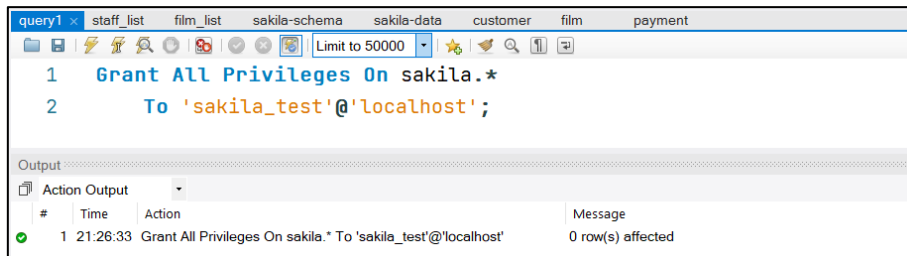
- (1) 执行命令新建 sakila_test 用户（密码 123456）；



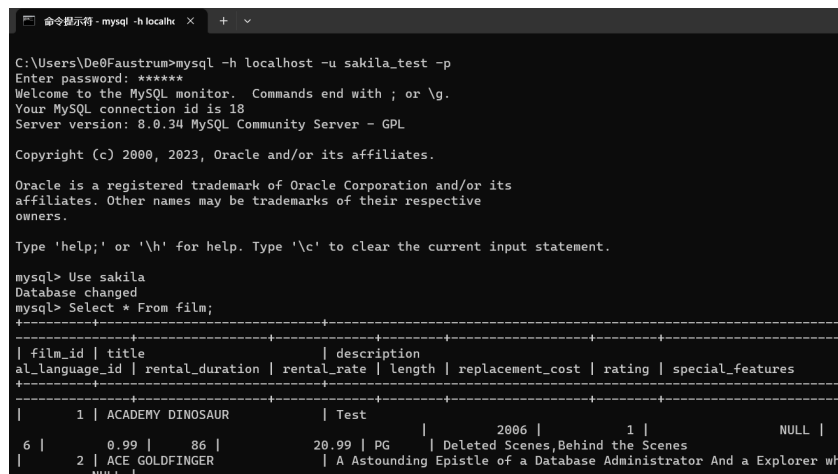
- (2) 执行命令查看当前已有用户；



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



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



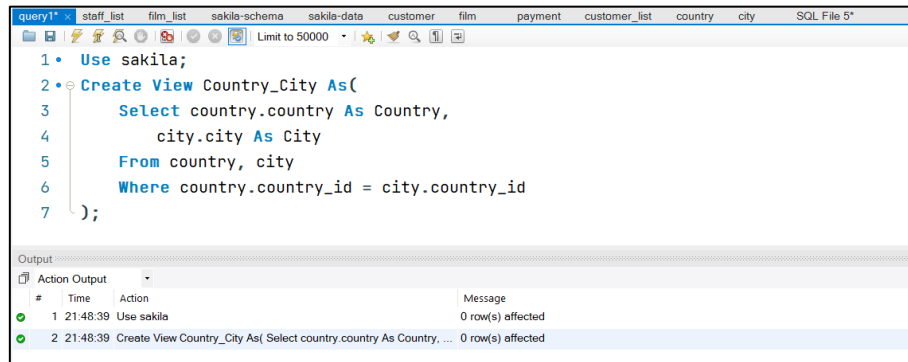
三、设计并实现

根据应用场景，为 Sakila 数据库合理地设计并实现：

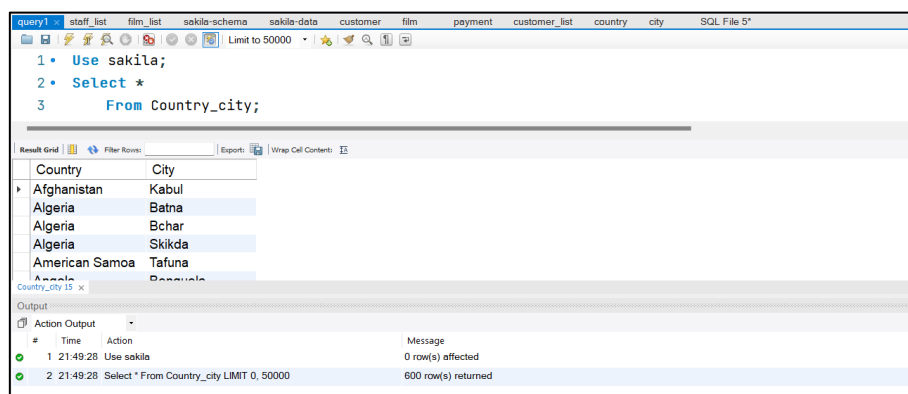
（截图语句和执行结果）

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：

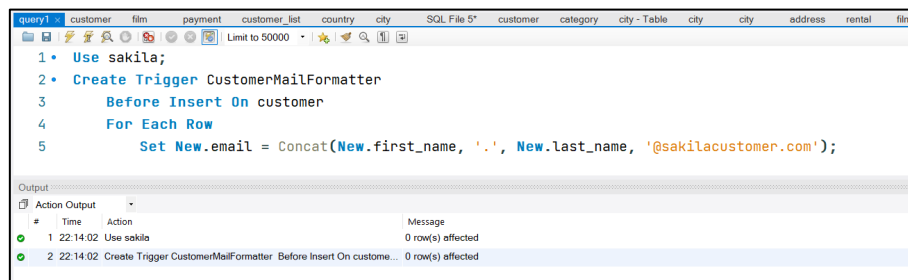


（2） 执行 select * from [视图名]，截图执行结果：

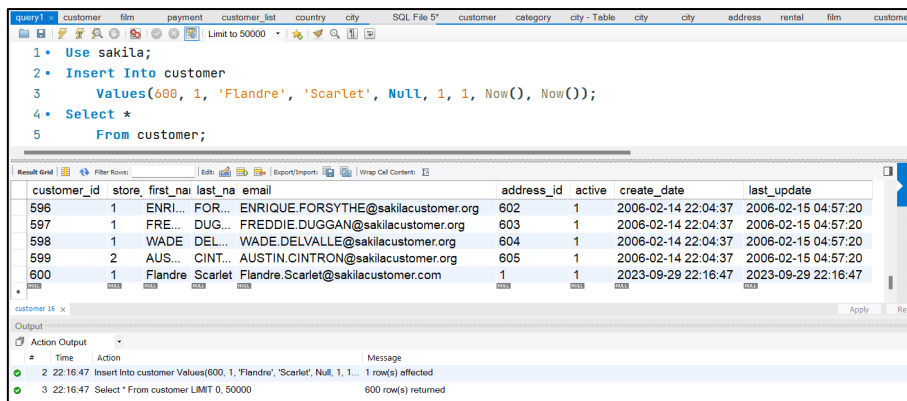


2. 设计 1 个触发器，需要体现触发器生效。

（1） 执行新建触发器的语句，并截图 SQL 和执行结果：

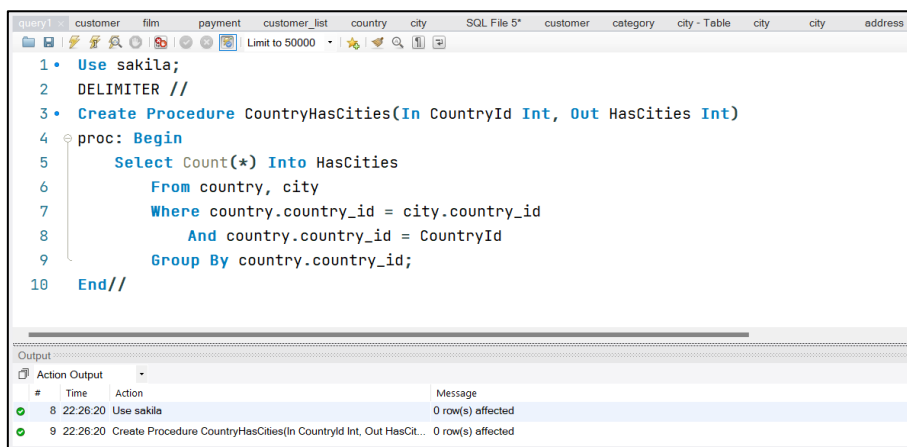


（2） 验证触发器是否生效，截图验证过程：



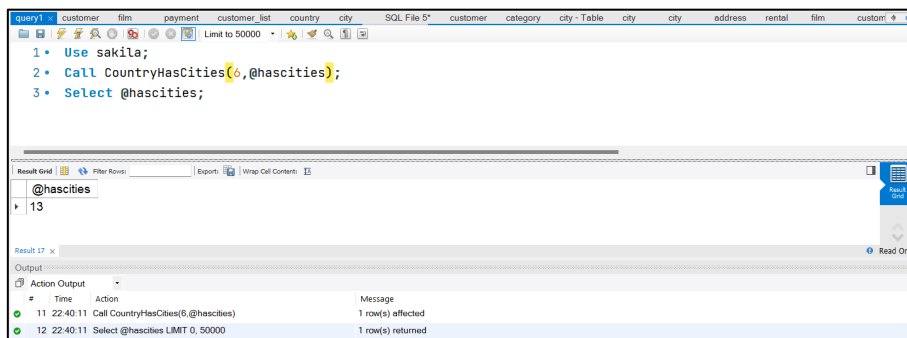
3. 设计 1 个存储过程，需要调用该存储过程。

(1) 执行新建存储过程的语句，并截图 SQL 和执行结果：



输入 country_id，返回其拥有的城市数量

(2) 调用该存储过程，截图调用结果：



country_id 为 6 的国家拥有 13 个城市，经验证结果正确。

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？

阿里数据库开发规范中禁止使用外键与级联的原因主要有以下几个方面：

性能问题:使用外键与级联会增加数据库的复杂性和开销。外键约束会引入额外的查询和锁操作,影响数据库的性能和响应速度。级联操作可能引发大量的级联更新或删除操作,导致性能下降、死锁等问题。

分布式部署问题:在分布式数据库系统中,外键约束的维护涉及到跨节点的协调和通信,增加了系统的复杂性和风险,不利于系统的可扩展性和高可用性。

数据一致性问题:外键约束会在数据库层面保证数据的一致性,但在实际应用中,处理数据的一致性通常更适合在应用层解决。应用层可以通过使用事务、业务逻辑代码等控制数据的操作和一致性,减少了对数据库层面的依赖。

因此,阿里数据库开发规范鼓励在应用层处理外键关系,例如通过应用层代码实现业务逻辑的验证和处理。这样可以更好地控制数据的处理和维护,并提高系统的性能和可扩展性。

需要注意的是,虽然阿里数据库开发规范中建议不使用外键与级联,但在某些特定情况下,如业务需求明确、数据库结构简单、性能要求不高的场景下,使用外键约束也是可以考虑的。在实际应用开发中,根据具体的业务需求、系统性能要求和团队约定来选择是否使用外键与级联,以满足项目的实际需要。