

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_category category flim	查询所有演员的 id、姓名以及他们所演的电影信息，电影信息包括按类别分类的电影名
film_list	Flim Category film_category actor film_actor	查询所有电影的 id、电影名、类型、分级、租金、描述、长度、演员表。
sales_by_film_category	Category Payment Rental inventory film film_category category	查询各类别电影的总销金额

(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？

21 11:13:25 update staff_list set `zip code` = '518055' where ID = '1'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
22 11:13:46 update film_list set price = 1.99 where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is not updatable

如图，第一句成功进行了更新，第二局不能进行更新。经分析，staff_list 视图是对表通过简单地选择若干属性得到的，进行更新时可以更新到对应的表。film_list 视图对几个表用了 group-by 操作，因此进行更新时无法确定要更新到哪个表。

在视图不包含复杂的计算、聚合操作时，可以进行 update 操作，否则不能。

- (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';
```

The screenshot shows a database management tool interface. At the top, there's a 'Result Grid' tab with a table showing the results of the query. Below it, there's an 'Output' tab showing a log of actions.

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

#	Time	Action	Message
✓ 15	17:05:37	select *, timestampdiff(day, rental_date, return_date) from rental LIMIT 0, 1000	1000 row(s) returned
✓ 16	17:06:32	select first_name, last_name, TIMESTAMPDIFF(second, rental_date, return_date) as duration from r...	6 row(s) returned
✓ 17	17:09:22	delete from customer where customer_id = 600	0 row(s) affected
✗ 18	17:15:52	insert into actor values (1,'P22','GUINESZ','2006-02-15 04:34:33'),	Error Code: 1064. You have an error in your SQL syntax; check
✗ 19	17:15:58	insert into actor values (1,'P22','GUINESZ','2006-02-15 04:34:33')	Error Code: 1062. Duplicate entry '1' for key 'actor.PRIMARY'
✓ 20	10:35:59	SELECT * FROM sakila.staff_list LIMIT 0, 1000	2 row(s) returned
✓ 21	11:13:25	update staff_list set 'zip code' = '518055' where ID = '1'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✗ 22	11:13:46	update film_list set price = 1.99 where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is no
✓ 23	11:23:59	SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakil...	7 row(s) returned

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）

Limit to 1000 rows

```

1 -- insert into customer( store_id, first_name, last_name, email, address_id)
2 -- values (1, 'Kobe', 'Bryant', null, 1)
3
4 • select * from customer

```

Result Grid

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
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
601	1	Kobe	Bryant	NULL	1	1	2024-09-26 11:33:10	2024-09-26 11:33:10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

customer 1

Output

Action Output

#	Time	Action	Message
18	17:15:52	insert into actor values (1,'P22','GUINESZ','2006-02-15 04:34:33'),	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '2006-02-15 04:34:33' at line 1
19	17:15:58	insert into actor values (1,'P22','GUINESZ','2006-02-15 04:34:33')	Error Code: 1062. Duplicate entry '1' for key 'actor.PRIMARY'
20	10:35:59	SELECT * FROM sakila.staff LIMIT 0, 1000	2 row(s) returned
21	11:13:25	update staff_list set 'zip code' = '518055' where ID = '1'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
22	11:13:46	update film_list set price = 1.99 where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is not updatable
23	11:23:59	SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakila'	7 row(s) returned
24	11:29:41	SELECT * FROM sakila.customer LIMIT 0, 1000	588 row(s) returned
25	11:33:10	insert into customer(store_id, first_name, last_name, email, address_id) values (1, 'Kobe', 'Bryant', null, 1)	1 row(s) affected
26	11:33:33	select * from customer LIMIT 0, 1000	589 row(s) returned

```

1 -- insert into customer( store_id, first_name, last_name, email, address_id, create_date)
2 -- values (1, 'Kobe', 'LaoDa', null, 1, '2006-02-15 05:09:17')
3
4 • select * from customer

```

建在 customer 表上。在 customer 表新建一条数据时，自动把 create-date 设置为当前时间。如上图所示，新增了一条信息，create-date 可以设置或者不设置，最终 create-date 都会自动用当前填入。

- (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）

```
2
3  -- id=1 的 description 原本为'A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies'
4
5  -- up date film
6  -- set description = 'A Bpic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies'
7  -- where film_id = 1;
8
9  • select * from film;
```

film_id	title	description	release_year	language_id	original_language_id	rental_duration	r
1	ACADEMY DINOSAUR	A Bpic Drama of a Feminist And a Mad Scientist who must Battle a T...	2006	1	NULL	6	0
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrator And a Explorer wh...	2006	1	NULL	3	4
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a Car who must Sink a...	2006	1	NULL	7	2
4	AFFAIR PREJUDICE	A Fandful Documentary of a Frisbee And a Lumberjack who must Ch...	2006	1	NULL	5	2

film 2 x

Output

Action Output

#	Time	Action	Message
22	11:13:46	update film_list set price = 1.99 where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is not updatable
23	11:23:59	SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakil...	7 row(s) returned
24	11:29:41	SELECT * FROM sakila.customer LIMIT 0, 1000	588 row(s) returned
25	11:33:10	insert into customer(store_id, first_name, last_name, email, address_id) values (1, 'Kobe', 'Bryant', nul...	1 row(s) affected
26	11:33:33	select * from customer LIMIT 0, 1000	589 row(s) returned
27	11:40:45	SELECT * FROM sakila.film LIMIT 0, 1000	1000 row(s) returned
28	11:40:52	SELECT * FROM sakila.film LIMIT 0, 1000	1000 row(s) returned
29	11:42:57	update film set description = 'A Bpic Drama of a Feminist And a Mad Scientist who must Battle a Tea...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
30	11:43:22	select * from film LIMIT 0, 1000	1000 row(s) returned

建在 `film` 表上。一旦检测到 `film` 的 `title`、`description`、`film_id` 三个字段中任何一个发生了变化，就把 `film_text` 表中对应的字段更新成新值。保证 `film_text` 与 `film` 的信息是一致的。

上图所示，先将 `film` 表中，`film_id` 为 1 的项的 `description` 字段，从 'A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies' 改成了 'A Bpic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies'。

```
10
11 • select * from film_text
```

film_id	title	description
1	ACADEMY DINOSAUR	A Bpic Drama of a Feminist And a Mad Scientist ...
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrat...
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a ...
4	AFFAIR PREJUDICE	A Fandful Documentary of a Frisbee And a Lum...
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...

film_text 3 x

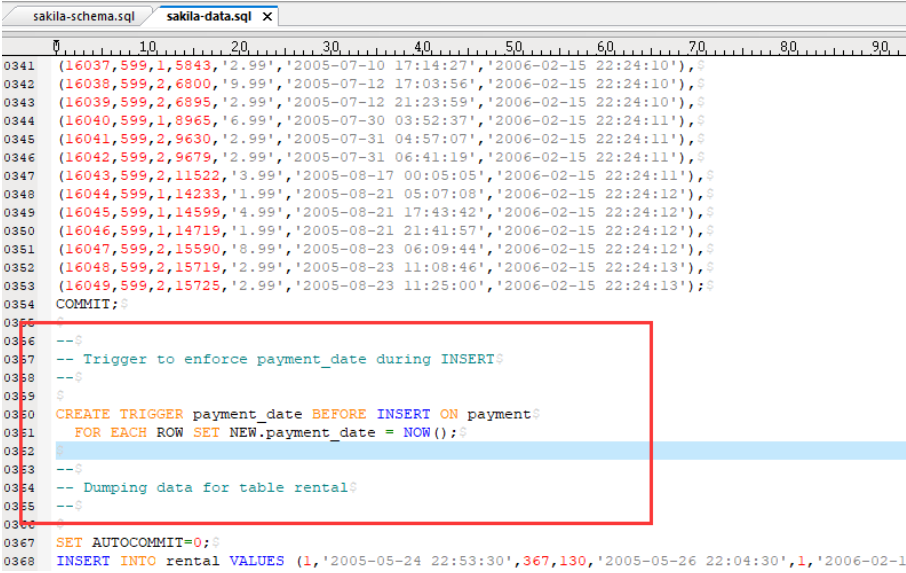
Output

Action Output

#	Time	Action	Message
23	11:23:59	SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakil...	7 row(s) returned
24	11:29:41	SELECT * FROM sakila.customer LIMIT 0, 1000	588 row(s) returned
25	11:33:10	insert into customer(store_id, first_name, last_name, email, address_id) values (1, 'Kobe', 'Bryant', nul...	1 row(s) affected
26	11:33:33	select * from customer LIMIT 0, 1000	589 row(s) returned
27	11:40:45	SELECT * FROM sakila.film LIMIT 0, 1000	1000 row(s) returned
28	11:40:52	SELECT * FROM sakila.film LIMIT 0, 1000	1000 row(s) returned
29	11:42:57	update film set description = 'A Bpic Drama of a Feminist And a Mad Scientist who must Battle a Tea...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
30	11:43:22	select * from film LIMIT 0, 1000	1000 row(s) returned
31	11:46:30	select * from film_text LIMIT 0, 1000	1000 row(s) returned

如上图所示，更新后再到 `film_text` 表进行查询，`film_id` 为 1 的项的 `description` 已经同步地进行了修改。

(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。`payment_date` 触发器必须放在 `payment` 表数据构建完成后创建。因为 `payment` 表需要填入 `payment_date` 为 2005 年或 2006 年的数据,如果 `payment_date` 触发器已经创建好,那么再向 `payment` 表新增的数据 `payment_date` 字段都会自动获取为当前时间。

这和 2 (1) 题中 `customer_create_date` 触发器是一样的原因,在 `customer_create_date` 触发器创建完成之后,新增到 `customer` 的数据即使设置了 `create_date`,最终也会更新成系统当前时间。

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 primary key	约束 <code>store_id</code> , 确保 <code>store_id</code> 是唯一的, 不会有重复值, 并且不允许为 <code>NULL</code> , 确保了每个商店都有唯一标识符。
唯一键约束 UNIQUE KEY	约束 <code>manager_staff_id</code> , 确保该字段的值在该表中是唯一值
非空约束 NOT NULL	约束的属性不能为 <code>null</code> 值

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

由于 343 行的属性引用自 `staff` (`staff_id`), `ON DELETE RESTRICT` 用于限制 `staff` 表中的删除操作, 限制 `staff` 表中数据的删除, 防止 `staff` 表删除了数据, 而 `store` 表又要引用, 破坏数据库完整性。 `ON UPDATE CASCADE` 使 `staff` 表有更新时, `store` 表同步更新。

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？
读入客户每月的最低购买次数和客户每月的最低消费金额，返回过去一个月内符合这两个条件的客户数量，并输出符合条件的客户。

(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`: 指示存储过程的输出可能不是唯一的，意味着同样的输入可以得到不同的输出。

`SQL SECURITY DEFINER`: 定义存储过程的执行权限是基于创建者的权限。

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？

读入客户 `id` 和一个日期，计算这个客户在该日期还需要支付的金额。返回值是客户还需支付的金额，是十进制数字。

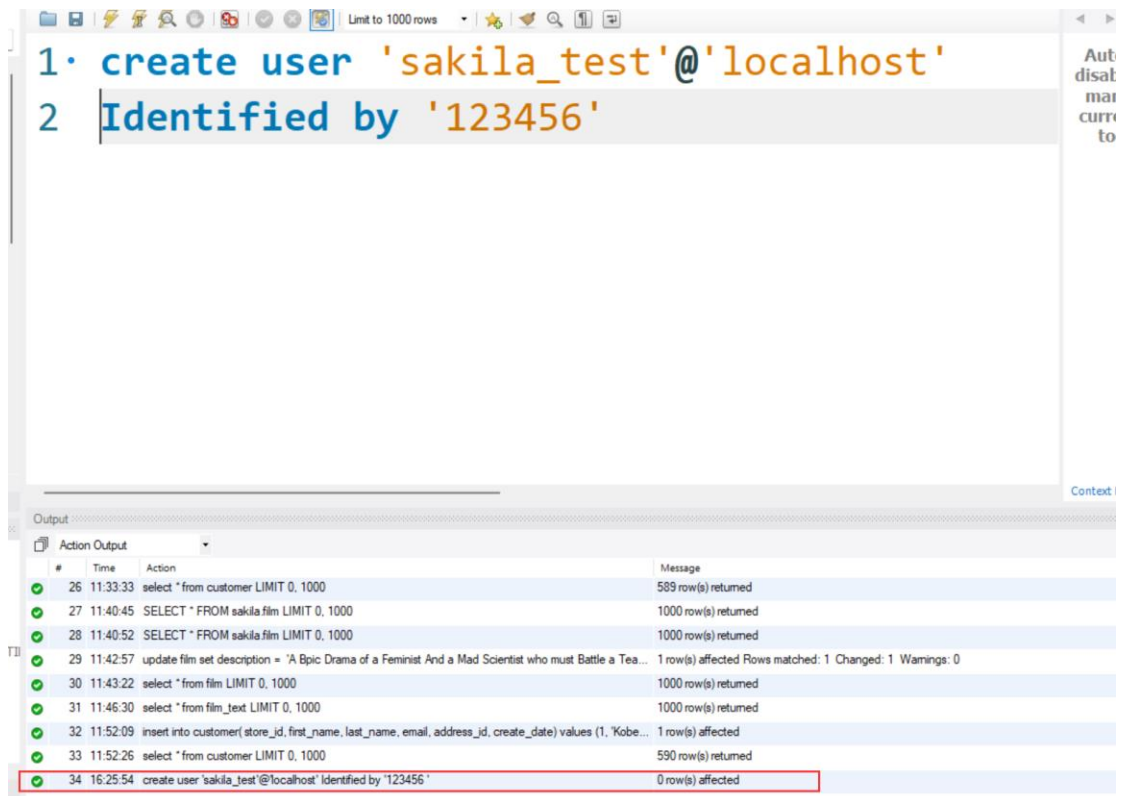
(2) 这个函数体中用到了 3 个函数，是哪几个函数？这 3 个函数的作用分别是？

函数	作用
<code>IFNULL()</code>	检查指定字段是否为 <code>null</code> ，若不为 <code>null</code> 则返回它的值，若为 <code>null</code> 则返回指定的默认值
<code>SUM()</code>	对数据进行求和
<code>TO_DAYS()</code>	输入一个日期，将其转换为天数

二、 创建新用户并分配权限

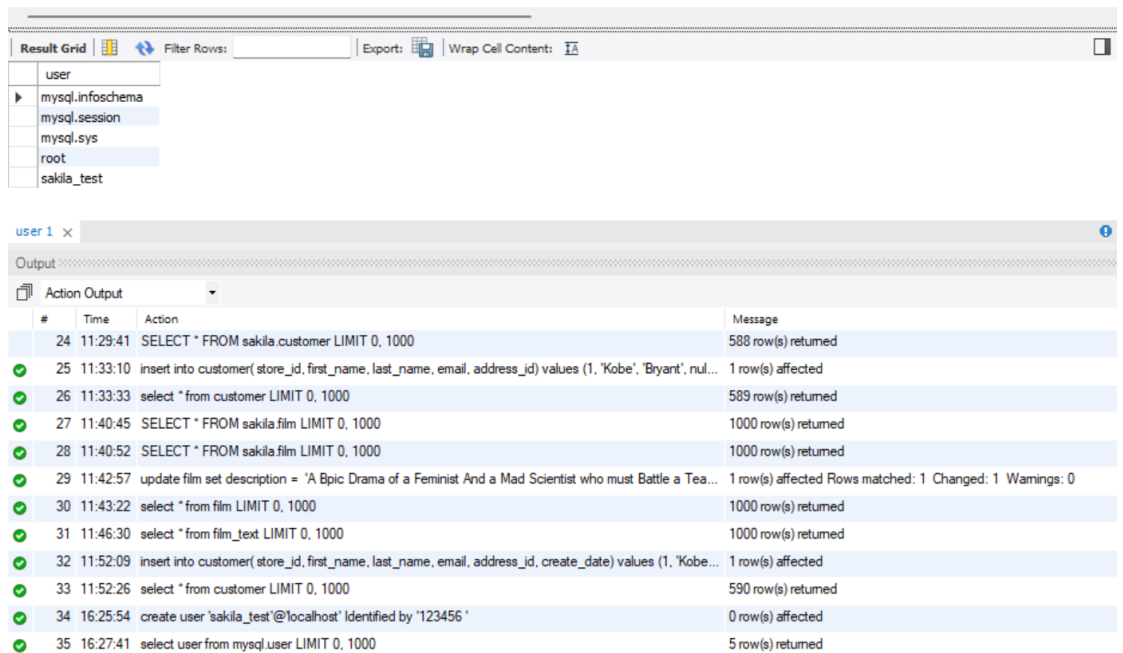
(截图语句和执行结果)

(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；

4. select user from mysql.user



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

```
6. grant all privileges on sakila.*
7. to 'sakila_test'@'localhost'
```

Output			
Action Output			
#	Time	Action	Message
34	16:25:54	create user 'sakila_test'@'localhost' identified by '123456'	0 row(s) affected
35	16:27:41	select user from mysql.user LIMIT 0, 1000	5 row(s) returned
36	16:29:37	grant all privileges on sakila.* to 'sakila_test'@'localhost'	0 row(s) affected

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

Query 1									
1 • select * from film									
2									
Result Grid									
film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	
1	ACADEMY DINOSAUR	A Bpic Drama of a Feminist And a Mad Scientist ...	2006	1	NULL	6	0.99	86	
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrat...	2006	1	NULL	3	4.99	48	
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a ...	2006	1	NULL	7	2.99	50	
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lum...	2006	1	NULL	5	2.99	117	
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...	2006	1	NULL	6	2.99	130	
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who...	2006	1	NULL	3	2.99	169	
7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who ...	2006	1	NULL	6	4.99	62	
8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must Con...	2006	1	NULL	6	4.99	54	
9	ALABAMA DEVIL	A Thoughtful Panorama of a Database Administ...	2006	1	NULL	3	2.99	114	
10	ALADDIN CALENDAR	A Action-Packed Tale of a Man And a Lumberjac...	2006	1	NULL	6	4.99	63	
11	ALAMO VIDEOTAPE	A Boring Epistle of a Butler And a Cat who must ...	2006	1	NULL	6	0.99	126	
12	ALASKA PHANTOM	A Fanciful Saga of a Hunter And a Pastry Chef ...	2006	1	NULL	6	0.99	136	
13	ALI FOREVER	A Action-Packed Drama of a Dentist And a Croc...	2006	1	NULL	4	4.99	150	
14	ALICE FANTASIA	A Emotional Drama of a A Shark And a Databas...	2006	1	NULL	6	0.99	94	
15	ALIEN CENTER	A Brilliant Drama of a Cat And a Mad Scientist w...	2006	1	NULL	5	2.99	46	
16	ALLEY EVOLUTION	A Fast-Paced Drama of a Robot And a Compose...	2006	1	NULL	6	2.99	180	
17	ALONE TRIP	A Fast-Paced Character Study of a Composer A...	2006	1	NULL	3	0.99	82	
18	ALTER VICTORY	A Thoughtful Drama of a Composer And a Femi...	2006	1	NULL	6	0.99	57	
19	AMADEUS HOLY	A Emotional Display of a Pioneer And a Technica...	2006	1	NULL	6	0.99	113	
20	AMELIE HELLFIGHTERS	A Boring Drama of a Woman And a Squirrel who...	2006	1	NULL	4	4.99	79	
21	AMERICAN CIRCUS	A Insightful Drama of a Girl And a Astronaut wh...	2006	1	NULL	3	4.99	129	
22	AMISTAD MIDSUMMER	A Emotional Character Study of a Dentist And a...	2006	1	NULL	6	2.99	85	
23	ANACONDA CONFES...	A Lacklustre Display of a Dentist And a Dentist...	2006	1	NULL	3	0.99	92	
Output									
Action Output									
#	Time	Action	Message						
1	22:45:17	use sakila	0 row(s) affected						
2	22:45:29	select * from film LIMIT 0, 1000	1000 row(s) returned						

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：

```
1  -- 创建一个view
2  • create view hot_films
3  as
4  select film.film_id, title , count(distinct customer_id) as rent_times from rental, inventory, film
5  where rental.inventory_id = inventory.inventory_id
6  and inventory.film_id = film.film_id
7  group by film.film_id
8  having rent_times >= 20
9  order by rent_times desc
10
```

Output			
Action Output			
#	Time	Action	Message
10	23:05:43	select * as rent_times, film.film_id, title from rental, inventory, film where rental.inventory_id = inventor...	Error Code: 1064. You have an error in your SQL syntax; check the manual that c
11	23:05:48	select *, film.film_id, title from rental, inventory, film where rental.inventory_id = inventory.inventory_id...	15709 row(s) returned
12	23:06:35	select count(distinct customer_id), film.film_id, title from rental, inventory, film where rental.inventory_...	Error Code: 1054. Unknown column 'rent_times' in 'having clause'
13	23:06:48	select count(distinct customer_id) as rent_times, film.film_id, title from rental, inventory, film where re...	305 row(s) returned
14	23:07:30	create view hot_films as select count(distinct customer_id) as rent_times, film.film_id, title from rental, i...	0 row(s) affected
15	23:08:38	SELECT * FROM sakila.hot_films	305 row(s) returned
16	23:08:56	create view hot_films as select film.film_id, title , count(distinct customer_id) as rent_times from rental, ...	Error Code: 1050. Table 'hot_films' already exists
17	23:09:04	DROP VIEW 'sakila'.hot_films	0 row(s) affected
18	23:09:05	create view hot_films as select film.film_id, title , count(distinct customer_id) as rent_times from rental, ...	0 row(s) affected
19	23:09:09	SELECT * FROM sakila.hot_films	305 row(s) returned

如图，新建了一个 hot_film 视图，关联了 rental, inventory, film 表，查询热门的电影（用被不同人租赁的次数作为热门的指标）。

（2） 执行 select * from [视图名]，截图执行结果：

The screenshot shows a database management tool interface. At the top, a SQL query is entered: `SELECT * FROM sakila.hot_films;`. Below the query editor, the 'Result Grid' displays the results of the query. The results are organized into a table with three columns: `film_id`, `title`, and `rent_times`. The table contains 15 rows of data, including titles like 'ROBBERS JOON', 'ROCKETEER MOTHER', 'TIMBERLAND SKY', 'HOBBIT ALIEN', 'IDOLS SNATCHERS', 'SHOCK CABIN', 'CAT CONEHEADS', 'DOGMA FAMILY', 'ENEMY ODDS', 'ENGLISH BULWORTH', 'FAMILY SWEET', 'GRAFFITI LOVE', 'HARRY IDAHO', and 'MARRIED GO'. Below the 'Result Grid', the 'Output' section shows the execution log. The log includes the following entries:

#	Time	Action	Message
13	23:06:48	select count(distinct customer_id) as rent_times, film.film_id, title from rental, inventory, film where re...	305 row(s) returned
14	23:07:30	create view hot_films as select count(distinct customer_id) as rent_times, film.film_id, title from rental, i...	0 row(s) affected
15	23:08:38	SELECT * FROM sakila.hot_films	305 row(s) returned
16	23:08:56	create view hot_films as select film.film_id, title, count(distinct customer_id) as rent_times from rental, ...	Error Code: 1050. Table 'hot_films' already exists
17	23:09:04	DROP VIEW 'sakila'.hot_films	0 row(s) affected
18	23:09:05	create view hot_films as select film.film_id, title, count(distinct customer_id) as rent_times from rental, ...	0 row(s) affected
19	23:09:09	SELECT * FROM sakila.hot_films	305 row(s) returned

如图，运行命令查询了 `hot_film` 视图。

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：

The screenshot shows a SQL IDE with a script editor and an output window. The script editor contains the following SQL code:

```
1  -- 触发器: actor 如果检测到一个项全名是伟大的名字, 就给他换成RIP Laoda, 因为伟大去世了, 我们忌讳他的名字
2  DELIMITER ;;
3  CREATE TRIGGER `sensitive_name_filter` BEFORE INSERT ON `actor` FOR EACH ROW BEGIN
4      IF concat(new.first_name, ' ', new.last_name) in ('kobe bryant') -- 其他敏感名字也可以放进去
5      THEN
6
7          set new.first_name = 'RIP',
8              new.last_name = 'SeeYouAG';
9
10     END IF;
11 END;;
12
13
```

The output window shows the execution results:

#	Time	Action	Message
44	23:38:26	Apply changes to actor	Changes applied
45	23:38:34	CREATE TRIGGER 'sensitive_name_filter' BEFORE INSERT...	0 row(s) affected
46	23:38:41	INSERT INTO actor(first_name, last_name) values('kobe', bry...	Error Code: 4048. Invalid target for assignment in INSERT or UPDATE statement 'NEW first_name'.
47	23:40:42	Apply changes to actor	Changes applied
48	23:41:06	CREATE TRIGGER 'sensitive_name_filter' BEFORE INSERT...	0 row(s) affected

在 actor 表上新建了一个敏感名字过滤的触发器, 在 actor 表的插入操作之前触发, 如果检测到插入 actor 表的新数据是敏感名字集合中的, 就将名字强行修改。

(2) 验证触发器是否生效, 截图验证过程:

The screenshot displays a database IDE interface. At the top, a SQL script is shown with four lines: an INSERT statement for 'Kobe Bryant', followed by a SELECT statement. Below the script, the 'Result Grid' shows a table with columns: actor_id, first_name, last_name, and last_update. The table contains several rows, with the last row (actor_id 204) showing 'RIP' as the first name and 'SeeYouAG' as the last name. Below the result grid, the 'Output' pane shows an 'Action Output' log. The log entries include timestamps, actions, and messages. The final two entries are highlighted with a red box: entry 49 shows the INSERT statement affecting 1 row, and entry 50 shows the SELECT statement returning 201 rows.

```
1  INSERT INTO actor(first_name, last_name)
2  values('kobe', 'bryant');
3
4 • select * from actor;
```

actor_id	first_name	last_name	last_update
195	JAYNE	SILVERSTONE	2006-02-15 04:34:33
196	BELA	WALKEN	2006-02-15 04:34:33
197	REESE	WEST	2006-02-15 04:34:33
198	MARY	KEITEL	2006-02-15 04:34:33
199	JULIA	FAWCETT	2006-02-15 04:34:33
200	THORA	TEMPLE	2006-02-15 04:34:33
204	RIP	SeeYouAG	2024-09-26 23:41:09

#	Time	Action	Message
44	23:38:26	Apply changes to actor	Changes applied
45	23:38:34	CREATE TRIGGER 'sensitive_name_filter' BEFORE INSERT...	0 row(s) affected
46	23:38:41	INSERT INTO actor(first_name, last_name) values('kobe', 'bry...'	Error Code: 4048. Invalid target for assignment in INSERT or UPDATE statement 'NEW first_name'
47	23:40:42	Apply changes to actor	Changes applied
48	23:41:06	CREATE TRIGGER 'sensitive_name_filter' BEFORE INSERT...	0 row(s) affected
49	23:41:09	INSERT INTO actor(first_name, last_name) values('kobe', 'bry...'	1 row(s) affected
50	23:41:09	select * from actor	201 row(s) returned

在创建了触发器后，再插入一条全名为“Kobe Bryant”的记录，并查询，发现该记录自动设置成了“RIP SeeYouAG”。

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：

```
1  -- 存储程序 返回某个国家所能查到的城市
2  DELIMITER ;;
3  • create procedure cities_of_country(IN p_country VARCHAR(50), OUT cities VARCHAR(50))
4    READS SQL DATA
5  BEGIN
6    select city from country, city
7    where country.country_id = city.country_id
8    and country like p_country;
9
10   select count(*) from country, city
11   where country.country_id = city.country_id
12   and country like p_country
13   into cities;
14 END;;
```

Output

#	Time	Action	Message
54	23:55:56	create procedure cities_of_country(IN p_country VARCHAR(...	0 row(s) affected
55	23:55:59	select country, city from country, city where country.country_j...	589 row(s) returned
56	23:56:49	call cities_of_country('algeria', @cities)	Error Code: 1172. Result consisted of more than one row
57	00:00:35	DROP PROCEDURE 'sakila'.cities_of_country'	0 row(s) affected
58	00:00:36	create procedure cities_of_country(IN p_country VARCHAR(...	0 row(s) affected

创建存储过程，读入一个国家，返回这个国家能够查询到的城市数量，并输出这些城市名称。

(2) 调用该存储过程，截图调用结果：

```
1
2 • call cities_of_country('algeria', @cities);
3 -- select @cities
```

Result Grid

city
Batna
Béchar
Skikda

Result 6 x

Output

#	Time	Action	Message
55	23:55:59	select country, city from country, city where country.country_j...	589 row(s) returned
56	23:56:49	call cities_of_country('algeria', @cities)	Error Code: 1172. Result consisted of more than one row
57	00:00:35	DROP PROCEDURE 'sakila'.cities_of_country'	0 row(s) affected
58	00:00:36	create procedure cities_of_country(IN p_country VARCHAR(...	0 row(s) affected
59	00:00:44	call cities_of_country('algeria', @cities)	3 row(s) returned
60	00:01:02	select @cities	1 row(s) returned
61	00:02:02	call cities_of_country('algeria', @cities)	3 row(s) returned

图 1

```

2  -- call cities_of_country('algeria', @cities);
3 • select @cities

```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

#	Time	Action	Message
56	23:56:49	call cities_of_country('algeria', @cities)	Error Code: 1172. Result consisted of more than one row
57	00:00:35	DROP PROCEDURE 'sakila'.cities_of_country'	0 row(s) affected
58	00:00:36	create procedure cities_of_country(IN p_country VARCHAR(...	0 row(s) affected
59	00:00:44	call cities_of_country('algeria', @cities)	3 row(s) returned
60	00:01:02	select @cities	1 row(s) returned
61	00:02:02	call cities_of_country('algeria', @cities)	3 row(s) returned
62	00:02:51	select @cities	1 row(s) returned

图 2

```

1 • select country, city from country, city
2   where country.country_id = city.country_id
3
4
5  -- call cities_of_country('algeria', @cities);
6  -- select @cities

```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

country	city
Afghanistan	Kabul
Algeria	Batna
Algeria	Béchar
Algeria	Skikda
American Samoa	Tafuna
Angola	Benguela
Angola	Namibe
Anguilla	South Hill
Argentina	Almirante Brown

Result 8 x

Output

Action Output

#	Time	Action	Message
57	00:00:35	DROP PROCEDURE 'sakila'.cities_of_country'	0 row(s) affected
58	00:00:36	create procedure cities_of_country(IN p_country VARCHAR(...	0 row(s) affected
59	00:00:44	call cities_of_country('algeria', @cities)	3 row(s) returned
60	00:01:02	select @cities	1 row(s) returned
61	00:02:02	call cities_of_country('algeria', @cities)	3 row(s) returned
62	00:02:51	select @cities	1 row(s) returned
63	00:04:48	select country, city from country, city where country.country_i...	589 row(s) returned

图 3

图 1 调用了存储过程, 输出了 'Algeria' 能查询到的三座城市并把城市数量存储到 @cities, 图 2 验证了 @cities 的值确实为 3。图 3 为直接使用查询语句查询的结果, 验证了调用存储过程的查询结果正确。

四、附加题

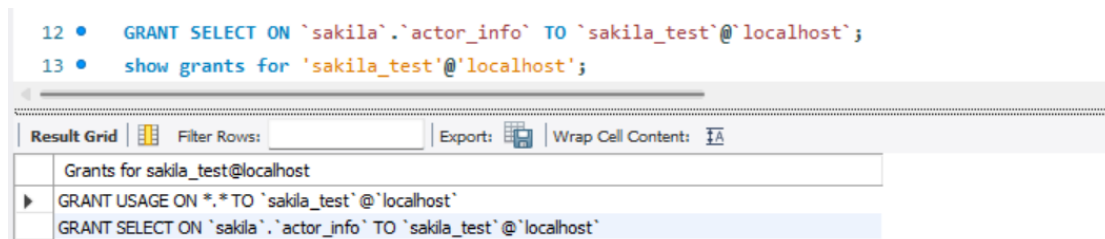
1. 为什么在 AFTER 触发器中只能对 NEW 取值, 不能对 NEW 进行赋值?

After 触发器是在数据操作之后执行的, 此时数据已经完成了更新, NEW 是最终确定的状态, 不能再进行更改。假如在 AFTER 触发器中对 NEW 赋值, 触发器和其坚实的操作会同时修改同一个表, 从而报错如下图。对于想要在数据操作后更改 NEW 的值的场景, 需要使用 BEFORE 触发器。

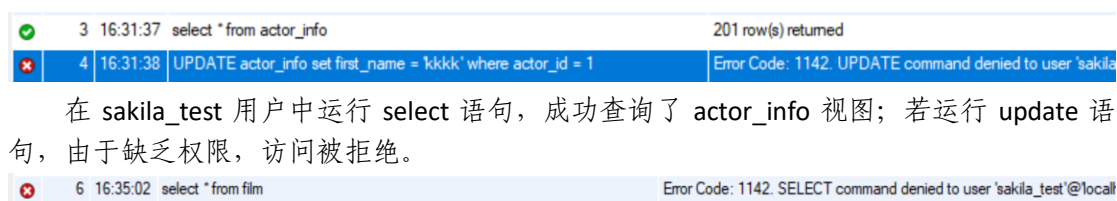
... Error Code: 1442. Can't update table 'actor' in stored function/trigger because it is already used by statement which invoked this stored function/trigger.

2. 请结合具体场景举例说明如何利用视图实现权限控制。

在实际的场景中, 我们希望不同的用户可以查询的数据库内容是不同的, 可以通过创建多个视图, 分别赋予不同的用户不同视图的权限, 每个用户只能查询自己有权限的视图。例如, 在 sakila 数据库中, 想要让 test 用户能够访问有关演员及他们出演的电影的信息, 但是不想让用户查询数据库的其他内容。就可以创建一个 actor_info 的视图, 仅赋予 test 用户该视图的权限。



通过执行上图命令, 可以看出, sakila_test 用户仅具有 sakila.actor_info 这个视图的查询权限。



在 sakila_test 用户中运行 select 语句, 成功查询了 actor_info 视图; 若运行 update 语句, 由于缺乏权限, 访问被拒绝。

如图, 在 sakila_test 用户中运行 select 语句查询基本表 film, 由于该用户没有权限, 访问被拒绝。

对于多个用户有不同查询权限的情况, 对每个用户分别创建可以访问的视图, 并且赋予每个用户对应视图的权限, 即可实现权限的控制。