

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?

7 个

(2) 分析以下 3 个视图，回答以下问题：

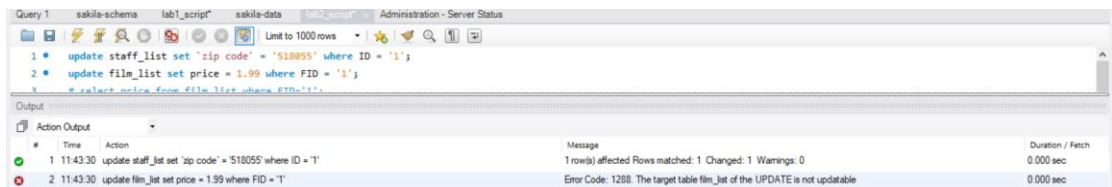
视图名	关联表	作用
actor_info	film,film_actor,actor	指示演员与电影间的关系
film_list	film,category,film_category,film_actor	指示电影的类别，租借率，以及电影演员
sales_by_store	payment,rental,inventory,store, address, city, country, staff,	指示一家商店在不同地区的销量以及负责人员

(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？



staff_list 这个表的视图修改的字段只有 staff 这个表有，而 film_list 这个视图的 FID 在多个表中都有，不能如此更新

(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';
```

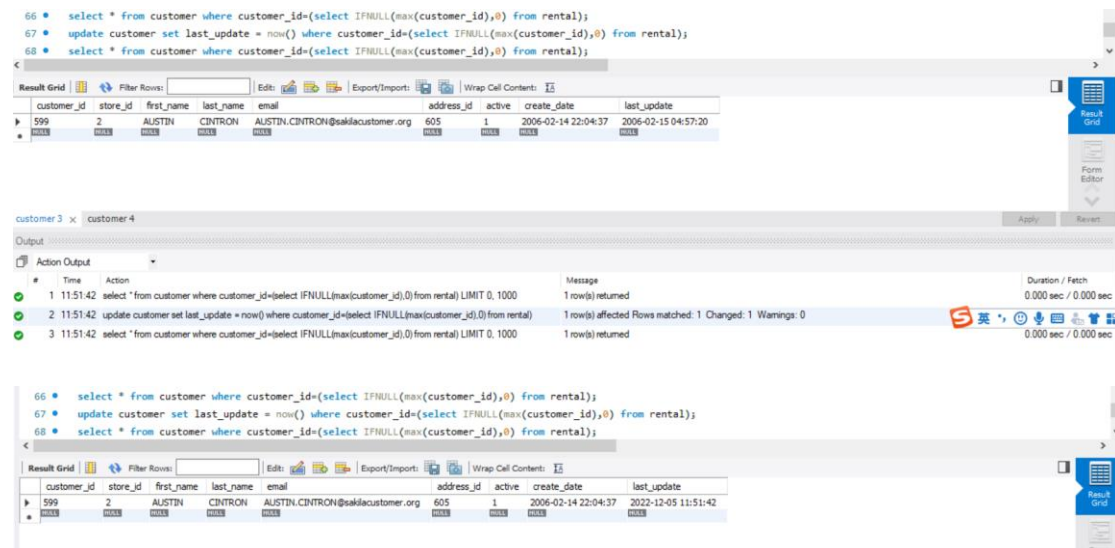


2. 关于触发器

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

建在 customer 表上，用于默认设置 create_date 这个属性

- (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）



- (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?

不可以,如果移入 schema 中,data 插入的数据中的 create_date 会强制变成插入时的时间

```

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

```

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
非空	
唯一性	保证商店 id 的唯一性
默认值	last_update 必须要有一个默认值

- (2) 图中第 343 行的 ON DELETE RESTRICT 和 ON UPDATE CASCADE 是什么意思？
只在 delete 或 update 时生效的约束

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）；

```

6 DROP USER IF EXISTS 'sakila_user'@'localhost';
7 create user 'sakila_user'@'localhost' identified by '123456';
8 grant all privileges on sakila.* to 'sakila_user'@'localhost';
9 show grants for 'sakila_user'@'localhost';
10 # 建立一个库存和商店的视图 inventory_info
11

```

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

Grants for sakila_user@localhost

GRANT USAGE ON *.* TO 'sakila_user'@'localhost'

GRANT ALL PRIVILEGES ON 'sakila'.* TO 'sakila_user'@'localhost'

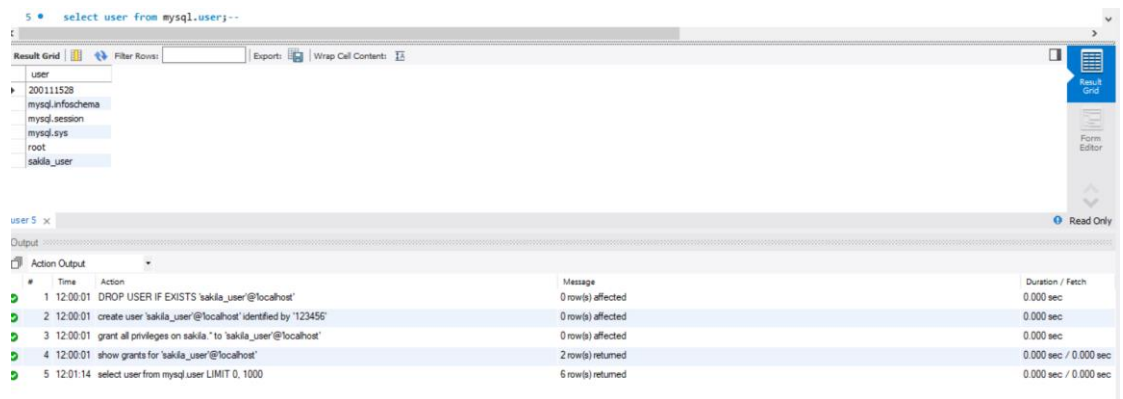
Result 4 x

Output

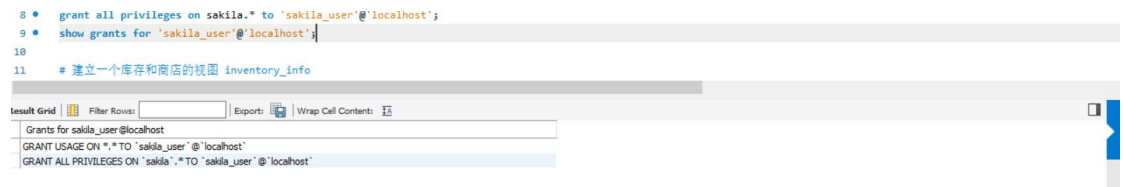
Action Output

#	Time	Action	Message	Duration / Fetch
1	12:00:01	DROP USER IF EXISTS 'sakila_user'@'localhost'	0 row(s) affected	0.000 sec
2	12:00:01	create user 'sakila_user'@'localhost' identified by '123456'	0 row(s) affected	0.000 sec
3	12:00:01	grant all privileges on sakila.* to 'sakila_user'@'localhost'	0 row(s) affected	0.000 sec
4	12:00:01	show grants for 'sakila_user'@'localhost'	2 row(s) returned	0.000 sec / 0.000 sec

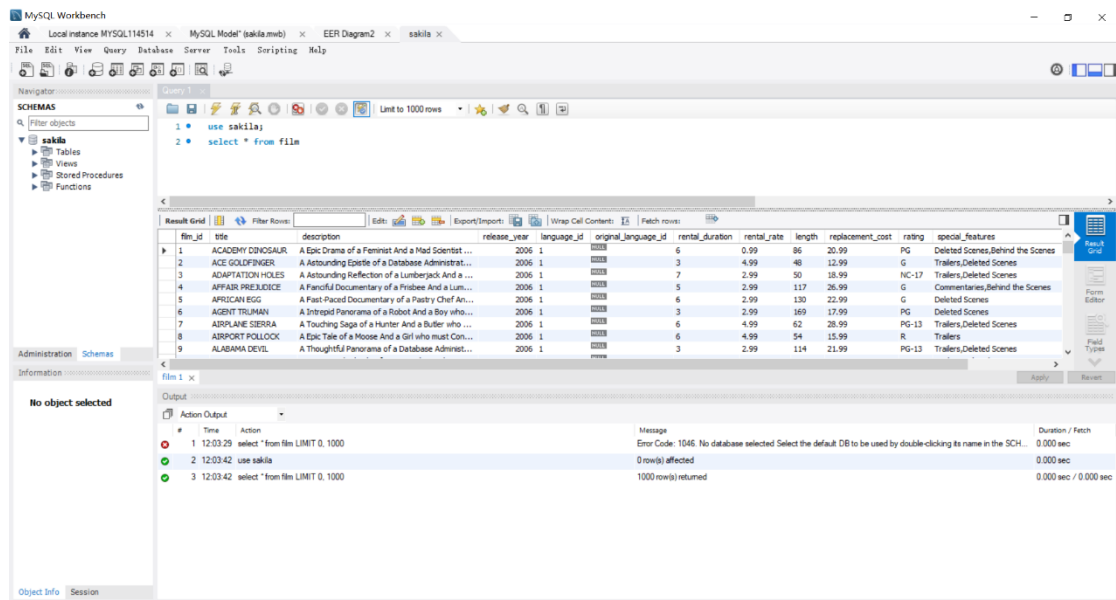
(2) 执行命令查看当前已有用户；



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



(4) 切换到 sakila_user 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)

语句:

```
DROP VIEW IF EXISTS inventory_info;# -- 支持多次运行 --
```

```
CREATE VIEW inventory_info
```

```
AS
```

```
SELECT st.store_id AS SID, inv.inventory_id AS INVID, f.title as FilmTitle, inv.last_update as Lastupdate
```

```
FROM store AS st JOIN inventory AS inv ON st.store_id = inv.store_id JOIN film AS f ON inv.film_id = f.film_id;
```

```
select * from inventory_info;
```

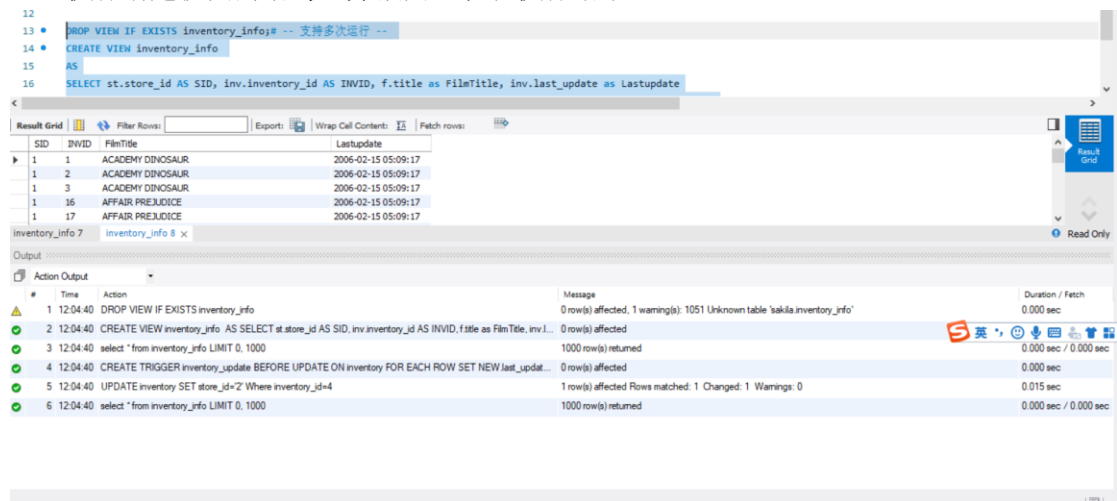
```
CREATE TRIGGER inventory_update BEFORE UPDATE ON inventory
FOR EACH ROW SET NEW.last_update = NOW();
```

```
UPDATE inventory SET store_id='2' Where inventory_id=4;
```

```
select * from inventory_info;
```

1. 设计 1 个视图，至少关联 2 个表；

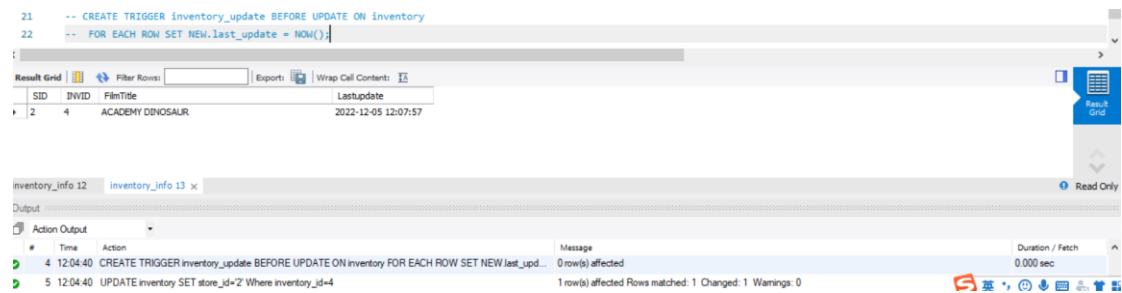
(1) 执行新建视图的语句，并截图 SQL 和执行结果：



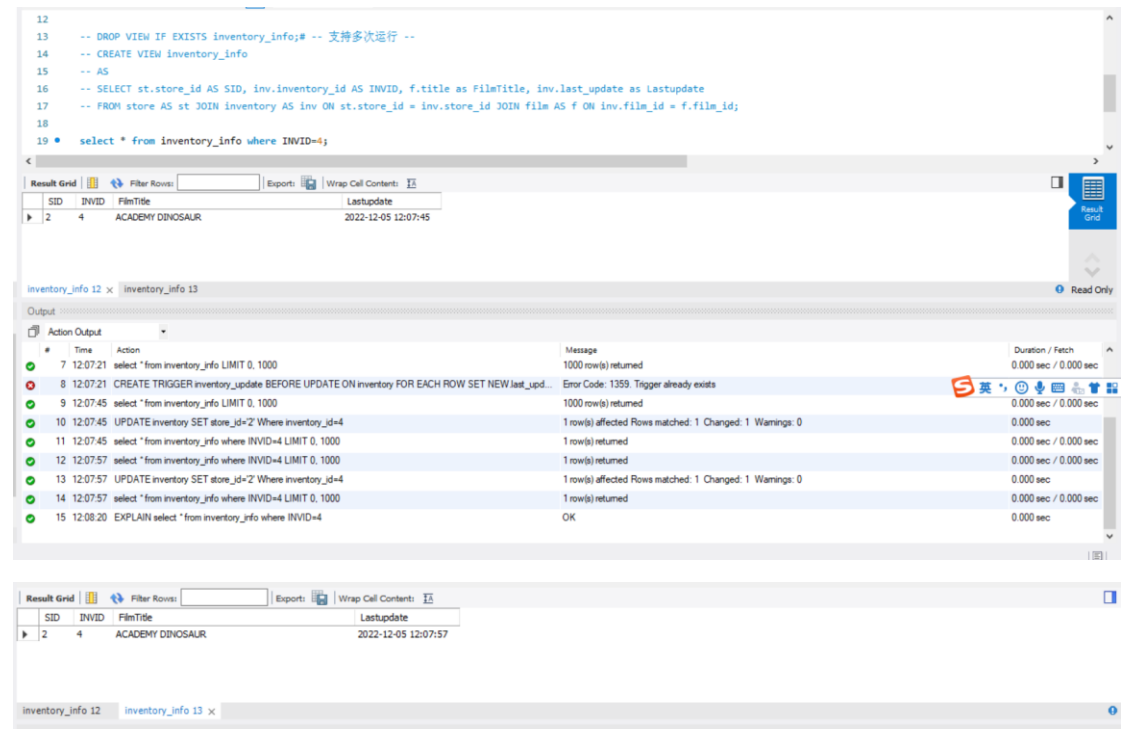
(2) 执行 select * from [视图名], 截图执行结果：

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：



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？

引入外键会引起维护以及添加的难度