

## 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

(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 、 film_actor、category、 film_category、	展示演员的 id、姓、名、演过影片的影片信息(包含电影类型和主题组成的字符串) (影片类型 1:影片主题 1,影片主题 2;)
film_list	film、category、actor、 film_category 、 film_actor	展示电影 id、主题、描述、类型、价格、时长、包含的演员
sales_by_store	payment 、 rental 、 inventory 、 store 、 address 、 city 、 country、 staff	展示商店的位置(城市,国家)、商店管理者姓名、总折扣销售额

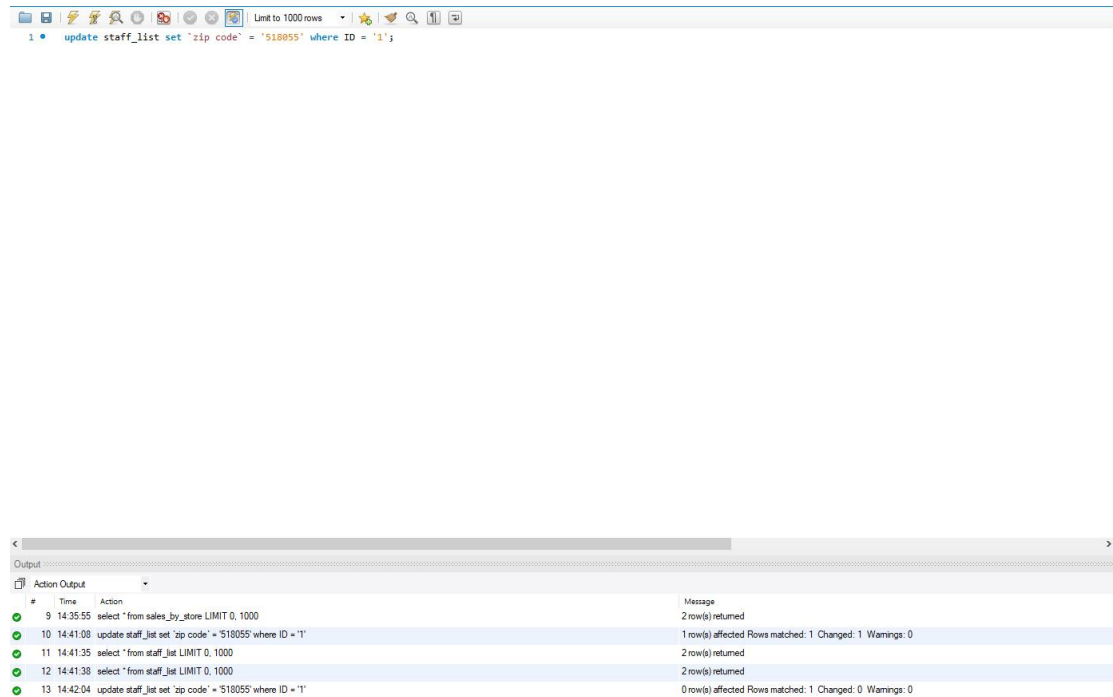
(3) 分别执行以下 2 句 SQL 语句：

```
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？

```
update staff_list set `zip code` = '518055' where ID = '1';
```



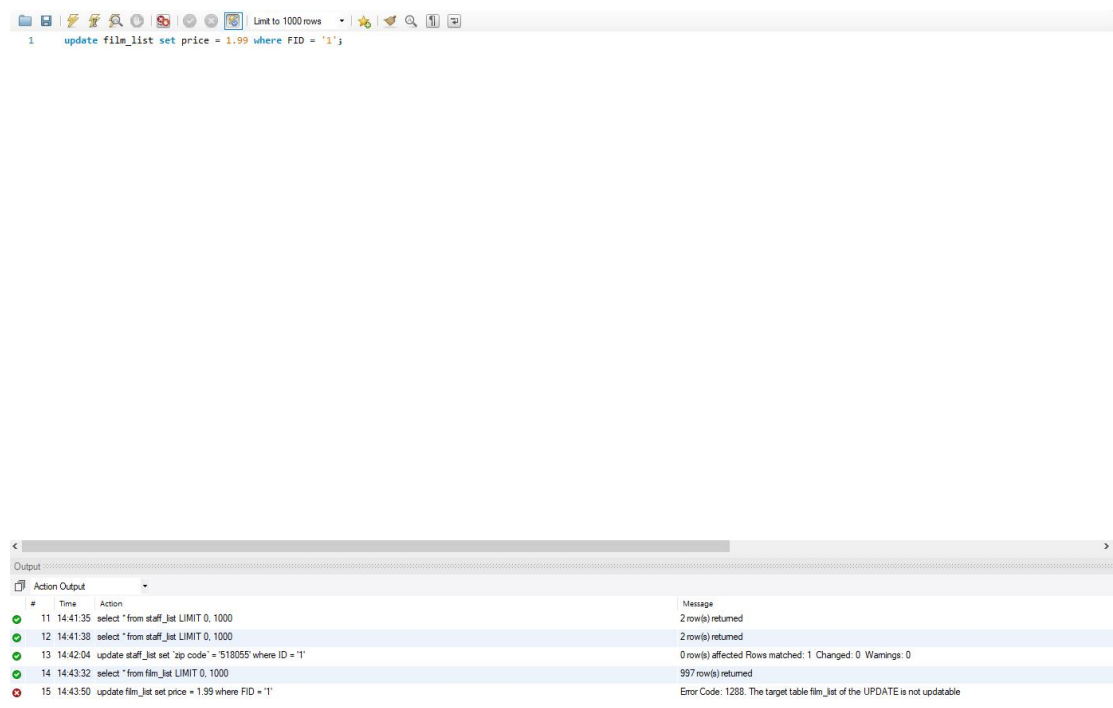
The screenshot shows a database client window with a toolbar at the top. Below the toolbar, a SQL query is entered: `update staff_list set `zip code` = '518055' where ID = '1';`. The main area of the window displays the 'Output' tab, which contains a table with columns: #, Time, Action, and Message. The table lists several actions, including a successful update of the 'zip code' for staff ID '1'.

#	Time	Action	Message
9	14:35:55	select * from sales_by_store LIMIT 0, 1000	2 row(s) returned
10	14:41:08	update staff_list set `zip code` = '518055' where ID = '1'	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
11	14:41:35	select * from staff_list LIMIT 0, 1000	2 row(s) returned
12	14:41:38	select * from staff_list LIMIT 0, 1000	2 row(s) returned
13	14:42:04	update staff_list set `zip code` = '518055' where ID = '1'	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0

分析：

第一条 update 语句修改成功，查看 staff\_list 视图，发现修改的 zip code 来源于 address 的 pental\_code 而 pental\_code 是只存在于 address 基表的数据，而 address 的主键 address\_id 作为 staff 的外键与 staff 相连，所以修改只影响了 address 基本表，而且该视图的构建没有聚合函数、派生列、分组函数，修改成功。

```
update film_list set price = 1.99 where FID = '1';
```



The screenshot shows a database client window with a toolbar at the top. Below the toolbar, a SQL query is entered: `update film_list set price = 1.99 where FID = '1';`. The main area of the window displays the 'Output' tab, which contains a table with columns: #, Time, Action, and Message. The table lists several actions, including a failed update of the 'price' for film ID '1'.

#	Time	Action	Message
11	14:41:35	select * from staff_list LIMIT 0, 1000	2 row(s) returned
12	14:41:38	select * from staff_list LIMIT 0, 1000	2 row(s) returned
13	14:42:04	update staff_list set `zip code` = '518055' where ID = '1'	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
14	14:43:32	select * from film_list LIMIT 0, 1000	997 row(s) returned
15	14:43:50	update film_list set price = 1.99 where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is not updatable

分析：

第二条 `update` 语句修改失败，查看 `film_list` 视图，发现修改的 `price` 来源于 `film` 的 `rental_rate`，该视图的构建运用了分组函数，因此不能 `update`，修改失败。

视图 `update` 修改条件：

1、`select` 语句在选择列表中不能包含聚合函数，也不包含 `TOP`, `GROUP BY`, `UNION` (除非视图是分区视图) 或 `DISTINCT` 子句。聚合函数可以用在 `FROM` 子句的子查询中，只要不修改函数返回的值。

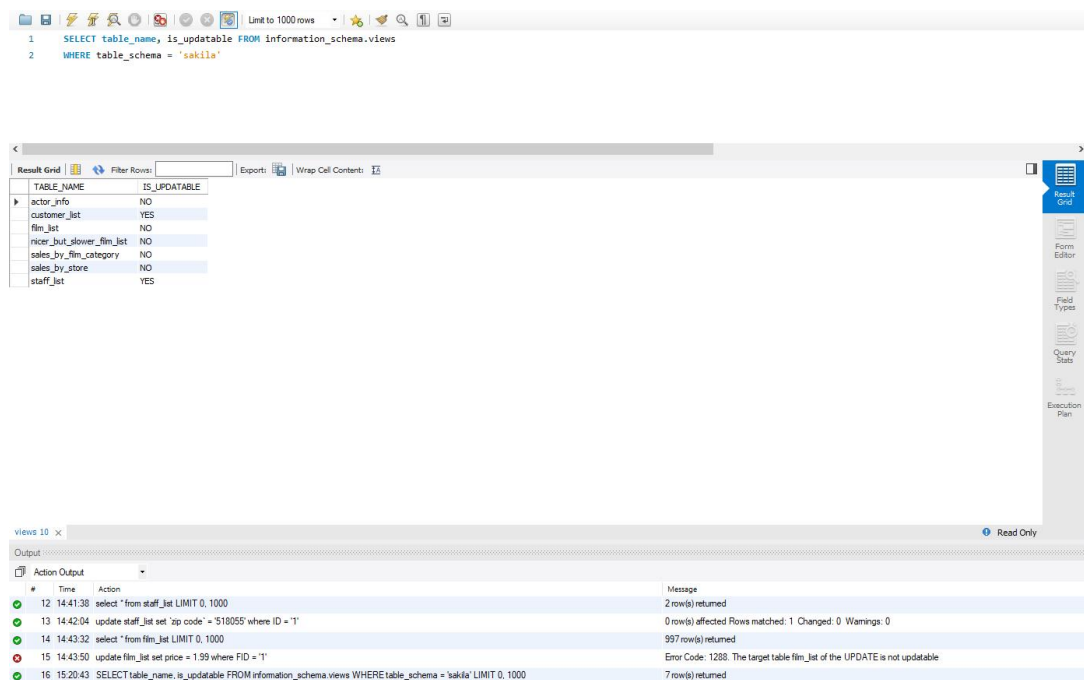
2、`select` 语句的选择列表中不能包含派生列。派生列是由任何非简单列表达式(使用函数、加法或减法运算符等)所构成的结果集列。

3、`select` 语句中的 `FROM` 子句至少引用一个表。`select` 语句不能只包含非表格格式的表达式(即不是从表派生出的表达式)。

4、`INSERT`, `UPDATE` 和 `DELETE` 语句在引用可更新视图之前，也必须如上述条件指定的那样满足某些限制条件。只有当视图可更新，并且所编写的 `UPDATE` 或 `INSERT` 语句只修改视图的 `FROM` 子句引用的一个基表中的数据时，`UPDATE` 和 `INSERT` 语句才能引用视图。

(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 displays a database management tool interface. At the top, the SQL query is entered: `SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakila';`. Below the query editor, the 'Result Grid' shows the following data:

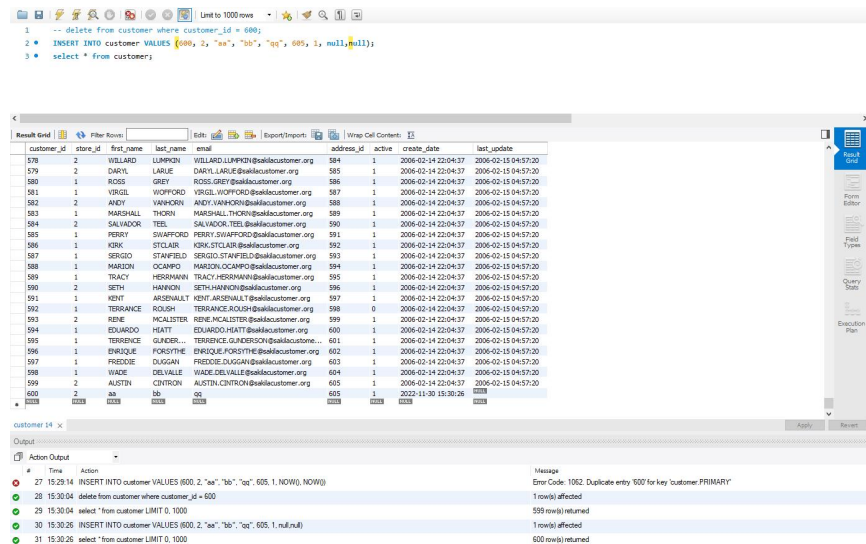
TABLE_NAME	IS_UPDATABLE
actor_info	NO
customer_list	YES
film_list	NO
ricer_bui_slower_film_list	NO
sales_by_film_category	NO
sales_by_store	NO
staff_list	YES

Below the result grid, the 'Output' window shows the execution log with the following messages:

#	Time	Action	Message
12	14:41:38	select "from staff_list LIMIT 0, 1000	2 row(s) returned
13	14:42:04	update staff_list set 'zip code' = 'S18055' where ID = '1'	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0
14	14:43:32	select "from film_list LIMIT 0, 1000	997 row(s) returned
15	14:43:50	update film_list set price = 1.99 where FID = '1'	Error Code: 1288. The target table film_list of the UPDATE is not updatable
16	15:20:43	SELECT table_name, is_updatable FROM information_schema.views WHERE table_schema = 'sakila' LIMIT 0, 1000	7 row(s) returned

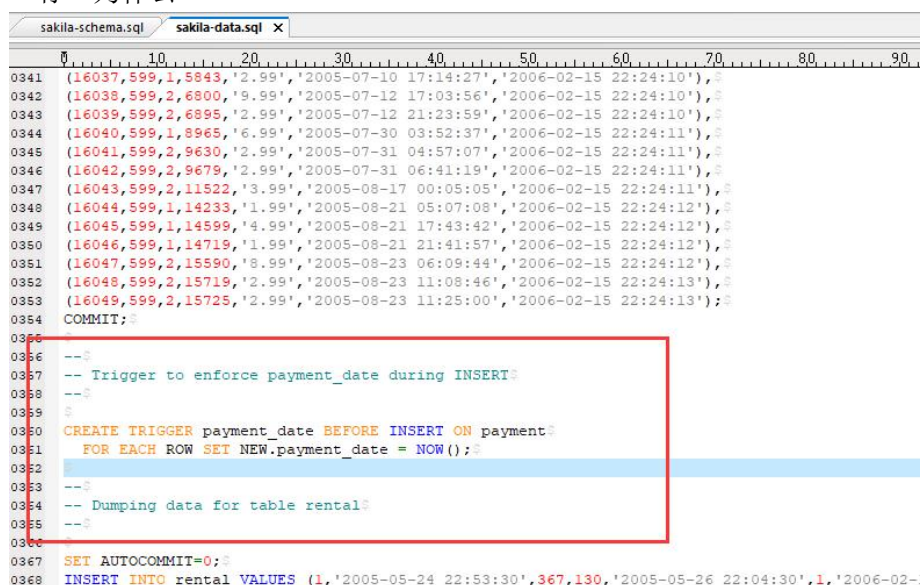
## 2. 关于触发器

- (1) 触发器 `customer_create_date` 建在哪个表上？这个触发器实现什么功能？  
`customer` 表上，在 `customer` 表的插入操作中，每次插入操作都自动将插入信息元组中的 `create_date` 赋值为当前时间 `NOW()`
- (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）



发现 `customer_id` 为 600 的数据中(新插入的数据)`create_date` 为当前时间

- (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？



不能，因为数据库先执行 `sakila-schema.sql` 里的语句是用于创建数据库的结构，再执行 `sakila-data.sql` 里的语句将初始数据插入数据库中，`payment_date` 触发器用于将之后每次插入 `payment` 的数据的 `payment_date` 赋值为当前时间，若是移动到 `sakila-schema.sql` 里执行，则 `sakila-data.sql` 里的 `payment` 的初始数据插入操作将把已经存在于 `payment` 的数据的 `payment_date` 都改为当前时间，错误。

### 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 PK = PRIMARY KEY	主键约束每个表只有一个，表示该属性是主键，用以确定数据的唯一性和标识该行数据
唯 一 约 束 UQ = UNIQUE	唯一约束，表示该属性的值是唯一的，用以确定数据的唯一性
非 空 约 束 NN = NOT NULL	非空约束，表示该属性的值不能为空

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

ON DELETE RESTRICT: 当在父表(即外键的来源表)中删除对应记录时，首先检查该记录是否有对应外键，如果有则不允许删除。

ON UPDATE CASCADE: 当在父表(即外键的来源表)中更新对应记录时，首先检查该记录是否有对应外键，如果有则不允许更新。

## 二、 创建新用户并分配权限

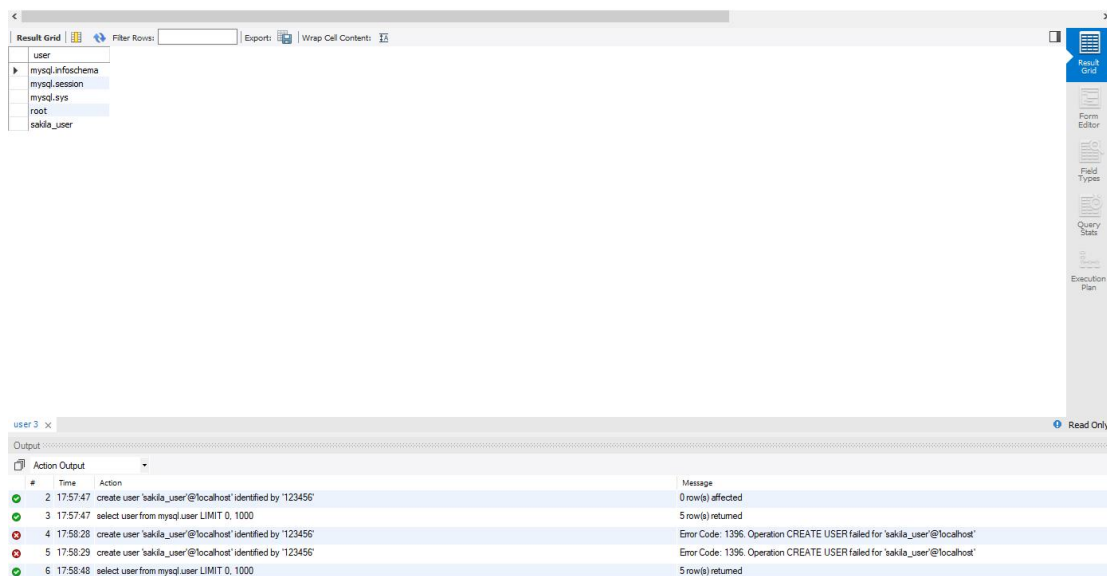
(截图语句和执行结果)

(1) 执行命令新建 sakila\_user 用户（密码 123456）；  
create user 'sakila\_user'@'localhost' identified by '123456';

2 17:57:47 create user 'sakila\_user'@'localhost' identified by '123456' 0 row(s) affected

(2) 执行命令查看当前已有用户；

```
4 -- create user 'sakila_user'@'localhost' identified by '123456';
5 • select user from mysql.user;
```



(3) 执行命令把 sakila 数据库的访问权限赋予 sakila\_user 用户；  
grant all privileges on sakila.\* to 'sakila\_user'@'localhost';

8 18:01:25 grant all privileges on sakila.\* to 'sakila\_user'@'localhost' 0 row(s) affected

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

```
1 • use sakila;
2 • select * from film;
```

The screenshot shows a database client interface. At the top, there's a 'Result Grid' displaying a table of film data. The table has columns: film\_id, title, description, release\_year, language\_id, original\_language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating, special\_features, and last\_update. Below the table, there's an 'Action Output' pane showing a log of executed SQL statements and their results.

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_update
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a Mad Scientist ...	2006	1	en_US	6	0.99	86	20.99	PG	Deleted Scenes, Behind the Scenes	2006-02-11
2	ACE GOLDFINGER	A Astounding Epistle of a Database Administrat...	2006	1	en_US	3	4.99	48	12.99	G	Trailers, Deleted Scenes	2006-02-11
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack And a ...	2006	1	en_US	7	2.99	50	18.99	NC-17	Trailers, Deleted Scenes	2006-02-11
4	AFFAIR PREJUDICE	A Fervid Documentary of a Pribee And a Lum...	2006	1	en_US	5	2.99	117	26.99	G	Commentaries, Behind the Scenes	2006-02-11
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef And...	2006	1	en_US	6	2.99	130	22.99	G	Deleted Scenes	2006-02-11
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who ...	2006	1	en_US	3	2.99	169	17.99	PG	Deleted Scenes	2006-02-11
7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a Butler who ...	2006	1	en_US	6	4.99	62	28.99	PG-13	Trailers, Deleted Scenes	2006-02-11
8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl who must Con...	2006	1	en_US	6	4.99	54	15.99	R	Trailers	2006-02-11
9	ALABAMA DEVL	A Thoughtful Panorama of a Database Administ...	2006	1	en_US	3	2.99	114	21.99	PG-13	Trailers, Deleted Scenes	2006-02-11

The Action Output pane shows the following log:

#	Time	Action	Message
1	18:03:47	select * from film LIMIT 0, 1000	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHI
2	18:03:53	select * from film LIMIT 0, 1000	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHI
3	18:03:57	select * from film LIMIT 0, 1000	Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHI
4	18:04:21	use sakila	0 row(s) affected
5	18:04:21	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：

use sakila;

CREATE VIEW category\_list

AS

SELECT distinct category.category\_id as id, category.name as name ,

(SELECT GROUP\_CONCAT(film.title ORDER BY film.title SEPARATOR ', ')

FROM film

INNER JOIN sakila.film\_category fc1

ON film.film\_id = fc1.film\_id

WHERE fc1.category\_id = category.category\_id

) as films

FROM category, film\_category fc2 WHERE category.category\_id = fc2.category\_id;

```
1 • use sakila;
2 • CREATE VIEW category_list
3 AS
4 SELECT distinct category.category_id as id, category.name as name ,
5 (SELECT GROUP_CONCAT(film.title ORDER BY film.title SEPARATOR ', ')
6 FROM film
7 INNER JOIN sakila.film_category fc1
8 ON film.film_id = fc1.film_id
9 WHERE fc1.category_id = category.category_id
10 ) as films
11 FROM category, film_category fc2 WHERE category.category_id = fc2.category_id;
```

The screenshot shows the 'Action Output' pane of the database client. It displays the execution log for the SQL statements provided in the previous block.

#	Time	Action	Message
20	18:37:07	use sakila	0 row(s) affected
21	18:37:07	CREATE VIEW category_list AS SELECT distinct category.category_id as id, category.name as name , (SELECT GROUP_CONCAT(film.title ORDER BY film.title SEPARATOR ', ')	0 row(s) affected

(2) 执行 `select * from [视图名]`，截图执行结果：

```
1 use sakila;
2 /*
3 CREATE VIEW category_list
4 AS
5 SELECT distinct category.category_id as id, category.name as name ,
6 (SELECT GROUP_CONCAT(film.title ORDER BY film.title SEPARATOR ', ')
7 FROM film
8 INNER JOIN sakila.film_category fc1
9 ON fc1.film_id = fc1.film_id
10 WHERE fc1.category_id = category.category_id
11 ) as films
12 FROM category, film_category fc2 WHERE category.category_id = fc2.category_id;
13 */
14 select * from category_list;
```

id	name	films
1	Action	AMADEUS HOLY, AMERICAN CIRCUS, ANTIHILL...
2	Animation	ALTER VICTORY, ANACONDA CONFESSIONS, A...
3	Children	BAGDASH UNDEFEATED, BEAR GRACELAND, B...
4	Classics	ALICE FANTASIA, ARIZONA BANG, BEAST HUN...
5	Comedy	AIRPLANE SIERRA, ANTHEM LUKE, BRINGING H...
6	Documentary	ACADEMY DINOSAUR, ADAPTATION HOLES, A...
7	Drama	APOLLO TEEN, BEAUTY GREASE, BEETHOVEN E...
8	Family	AFRICAN EGG, APACHE DIVINE, ATLANTIS CA...
9	Foreign	AGENT TRULMAN, ALAMO VIDEOTAP, ALIEN C...
10	Games	AUTUMN CROW, BULWORTH COMMANDMENTS...
11	Horror	ACE GOLDFINGER, AFFAIR PREJUDICE, AIRPO...
12	Music	ALASKA PHANTOM, ALONE TRIP, AMELIE HELL...
13	New	AMISTAD MIDSUMMER, ANGELS LIFE, APOCAL...
14	Sci-Fi	ANNIE IDENTITY, ARMAGEDDON LOST, ATTAC...
15	Sports	ALADDIN CALENDAR, ANONYMOUS HUMAN, A...
16	Travel	ARSENIC INDEPENDENCE, BASIC EASY, BIRD I...

category\_list9 x

Output

#	Time	Action	Message
22	18:39:43	use sakila	0 row(s) affected
23	18:39:43	select * from category_list LIMIT 0, 1000	16 row(s) returned

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：

```
CREATE TRIGGER customer_update_change_last_update BEFORE UPDATE ON customer
FOR EACH ROW SET new.last_update = NOW();
```

该触发器用以在每次修改 `customer` 表中数据时自动更新修改时间

```
7 use sakila;
8 CREATE TRIGGER customer_update_change_last_update BEFORE UPDATE ON customer
9 FOR EACH ROW SET new.last_update = NOW();
```

Output

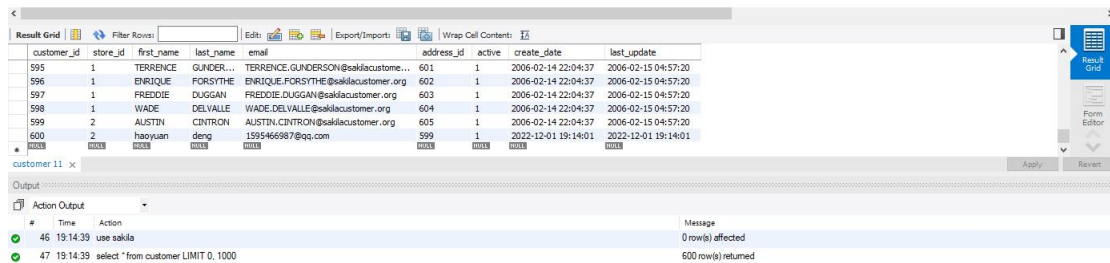
#	Time	Action	Message
16	19:11:08	use sakila	0 row(s) affected
17	19:11:08	CREATE TRIGGER customer_update_change_last_update BEFORE UPDATE ON customer FOR EACH ROW SET new.last_update = NOW()	0 row(s) affected

(2) 验证触发器是否生效，截图验证过程：



先插入一条自己的数据(customer\_id = 600)

```
16 • use sakila;  
17 • select * from customer;
```



The screenshot shows a database client interface. The top part displays a table with columns: customer\_id, store\_id, first\_name, last\_name, email, address\_id, active, create\_date, and last\_update. The table contains several rows, with the last row (customer\_id 600) highlighted. Below the table, the 'Output' window shows the results of SQL execution. It includes a message: '0 row(s) affected' and '600 row(s) returned'.

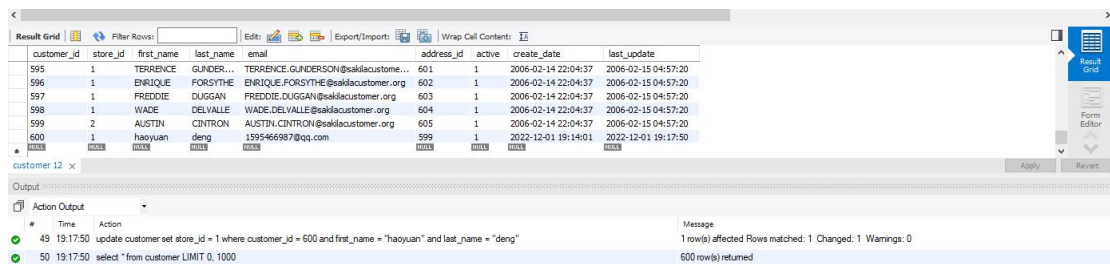
customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
595	1	TERRENCE	GUNDER...	TERRENCE.GUNDERSON@sakilacustome...	601	1	2006-02-14 22:04:37	2006-02-15 04:57:20
596	1	ENRIQUE	FORSYTHE	ENRIQUE.FORSYTHE@sakilacustomer.org	602	1	2006-02-14 22:04:37	2006-02-15 04:57:20
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
600	2	haoyuan	deng	1595466987@qq.com	599	1	2022-12-01 19:14:01	2022-12-01 19:14:01

Output:

#	Time	Action	Message
46	19:14:39	use sakila	0 row(s) affected
47	19:14:39	select * from customer LIMIT 0, 1000	600 row(s) returned

进行自己数据的更新之后，查看 last\_update 时间是否发生变化

```
16 • use sakila;  
17 • update customer set store_id = 1 where customer_id = 600 and first_name = "haoyuan" and last_name = "deng";  
18 • select * from customer;
```



The screenshot shows the same database client interface as before. The table now shows the updated data for customer\_id 600. The 'last\_update' column now shows '2022-12-01 19:17:50'. The 'Output' window shows the results of the update operation, indicating that 1 row was affected and 600 rows were returned.

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
595	1	TERRENCE	GUNDER...	TERRENCE.GUNDERSON@sakilacustome...	601	1	2006-02-14 22:04:37	2006-02-15 04:57:20
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
600	1	haoyuan	deng	1595466987@qq.com	599	1	2022-12-01 19:14:01	2022-12-01 19:17:50

Output:

#	Time	Action	Message
49	19:17:50	update customer set store_id = 1 where customer_id = 600 and first_name = "haoyuan" and last_name = "deng"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
50	19:17:50	select * from customer LIMIT 0, 1000	600 row(s) returned

发现 update\_time 产生了改变，变成了当时时间，触发器生效。

## 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？

原因：太多外键和级联的使用会大大增加表与表之间的耦合度和复杂度，让错误修改表的某个属性可能造成整个数据库的错误。

有必要，虽然外键的存在很容易让表之间的耦合性大大提升，但可以保持数据一致性，完整性，主要目的是控制存储在外键表中的数据。