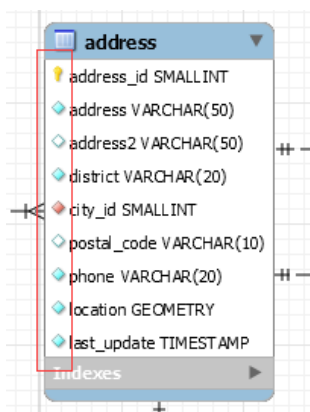


# 实验一报告

## 一、回答问题

请一边熟悉 sakila 数据库，一边回答以下问题：

1. sakila.mwb 模型中，表结构里每个字段前面的小标记分别表示什么意思？



标记	意义
⚡	该字段为该表的主键。
◆	该字段不允许为空值，且不是主键和外键。
◇	该字段允许为空值。
◆	该字段为该表的外键，且不允许空值。

2. 图中哪部分体现影片-演员关系？换句话说,如果要找出演某个影片的演员名字，访问哪几张表可以获得信息？

访问 `film`、`actor` 和 `film_actor` 即可获得所需信息。总体分为三个步骤：

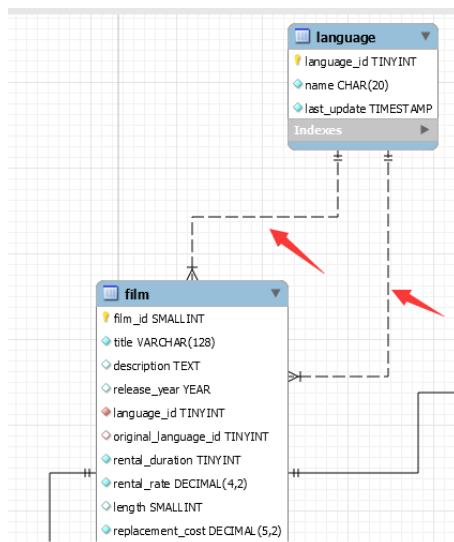
- ① 在 `film` 表中通过影片名 `title` 获得影片编号 `film_id`.
- ② 在 `film_actor` 表中通过影片编号 `film_id` 获得出演该影片的所有演员编号 `actor_id`.
- ③ 在 `actor` 表中通过演员编号 `actor_id` 获得演员名字 `first_name`、`last_name`.

3. 如果已知某个顾客姓名，要找到他租借的所有影片名，需要访问哪几张表？

访问 `customer`、`rental`、`inventory` 和 `film` 即可获得所需信息。总体分为四个步骤：

- ① 在 `customer` 表中通过顾客姓名 `first_name` 和 `last_name` 获得顾客编号 `customer_id`。
- ② 在 `rental` 表中通过顾客编号 `customer_id` 获得该顾客租借的所有 DVD 编号 `inventory_id`。
- ③ 在 `inventory` 表中通过 DVD 编号 `inventory_id` 获得影片编号 `film_id`。
- ④ 在 `film` 表中通过影片编号 `film_id` 获得影片名 `title`。

4. 为什么 `film` 和 `language` 表间会有 2 条线？



表示 `film` 表中的两个外键 `language_id` 与 `original_language_id` 均对应 `language` 表中的主键。

## 二、实验截图

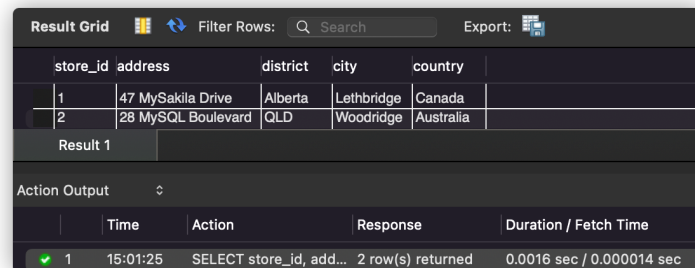
(注意截图清晰，截图时需要体现SQL语句、执行结果、Output窗口)

- 1、 请列出所有商店的详细地址，显示商店 id，商店地址，所在区域，所在城市，所在国家；

SQL 语句：

```
SELECT store_id, address, district, city, country
FROM store, address, country, city
WHERE store.address_id = address.address_id
      AND address.city_id = city.city_id
      AND city.country_id = country.country_id;
```

结果截图：



store_id	address	district	city	country
1	47 MySakila Drive	Alberta	Lethbridge	Canada
2	28 MySQL Boulevard	QLD	Woodridge	Australia

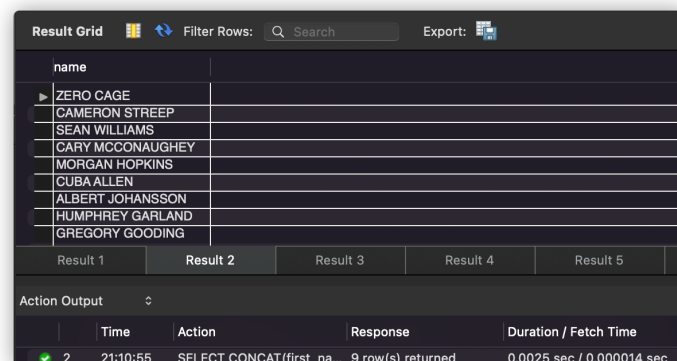
	Time	Action	Response	Duration / Fetch Time
1	15:01:25	SELECT store_id, add...	2 row(s) returned	0.0016 sec / 0.000014 sec

- 2、 哪些演员出演过影片《WEST LION》？请列出他的姓名；

SQL 语句：

```
SELECT CONCAT(first_name, ' ', last_name) as name
FROM actor, film, film_actor
WHERE film.title = 'WEST LION'
      AND actor.actor_id = film_actor.actor_id
      AND film.film_id = film_actor.film_id;
```

结果截图：



name
ZERO CAGE
CAMERON STREEP
SEAN WILLIAMS
CARY MCCONAUHUEY
MORGAN HOPKINS
CUBA ALLEN
ALBERT JOHANSSON
HUMPHREY GARLAND
GREGORY GOODING

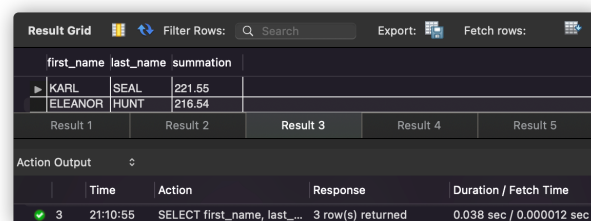
	Time	Action	Response	Duration / Fetch Time
2	21:10:55	SELECT CONCAT(first_na...	9 row(s) returned	0.0025 sec / 0.000014 sec

- 3、 找出租 DVD 花费最高的前 3 名，请列出他们的 first\_name, last\_name 和每个人花费的金额；

SQL 语句：

```
SELECT first_name, last_name, SUM(amount) as summation
FROM customer, payment
WHERE payment.customer_id = customer.customer_id
GROUP BY first_name, last_name
ORDER BY summation DESC
LIMIT 3;
```

结果截图：



first_name	last_name	summation
KARL	SEAL	221.55
ELEANOR	HUNT	216.54

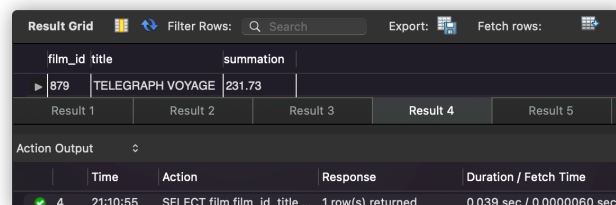
The screenshot shows a 'Result Grid' window with a search bar and 'Export'/'Fetch rows' buttons. Below the table, there are tabs for 'Result 1' through 'Result 5'. An 'Action Output' section at the bottom shows a successful query execution at 21:10:55, returning 3 rows in 0.038 seconds.

- 4、 哪个影片获得了总体最高的租金？请列出影片 id、影片名、总租金；

SQL 语句：

```
SELECT film.film_id, title, SUM(amount) as summation
FROM rental, payment, inventory, film
WHERE film.film_id = inventory.film_id
      AND payment.rental_id = rental.rental_id
      AND rental.inventory_id = inventory.inventory_id
GROUP BY title, film.film_id
ORDER BY summation DESC
LIMIT 1;
```

结果截图：



film_id	title	summation
879	TELEGRAPH VOYAGE	231.73

The screenshot shows a 'Result Grid' window with a search bar and 'Export'/'Fetch rows' buttons. Below the table, there are tabs for 'Result 1' through 'Result 5'. An 'Action Output' section at the bottom shows a successful query execution at 21:10:55, returning 1 row in 0.039 seconds.

- 5、 哪个演员出演的电影超过 40 部？ 请列出演员 id、演员名、出演的电影数；

SQL 语句：

```
SELECT actor.actor_id, CONCAT(first_name, ' ', last_name) as name,
COUNT(film_actor.film_id) as summation
```

```

FROM actor, film_actor
WHERE actor.actor_id = film_actor.actor_id
GROUP BY actor.actor_id, name
HAVING summation > 40;

```

结果截图：

The screenshot shows a database interface with a 'Result Grid' and an 'Action Output' section. The 'Result Grid' displays the results of a query, showing actor IDs and names with their summation values. The 'Action Output' section shows the execution details of the query.

actor_id	name	summation
102	WALTER TORN	41
107	GINA DEGENERES	42

Time	Action	Response	Duration / Fetch Time
21:10:55	SELECT actor.actor_id, C...	2 row(s) returned	0.0033 sec / 0.0000050 sec

6、 请找出没有租借过电影《WEST LION》的顾客姓名；

SQL 语句：

```

SELECT DISTINCT CONCAT(first_name, ' ', last_name) as name
FROM rental, customer
WHERE rental.customer_id = customer.customer_id
      AND rental.customer_id NOT IN (
        SELECT rental.customer_id
        FROM rental, inventory, film
        WHERE film.title = 'WEST LION'
              AND film.film_id = inventory.film_id
              AND inventory.inventory_id = rental.inventory_id
      );

```

结果截图：

The screenshot shows a database interface with a 'Result Grid' and an 'Action Output' section. The 'Result Grid' displays the results of a query, showing customer names. The 'Action Output' section shows the execution details of the query.

name
MARY SMITH
PATRICIA JOHNSON
LINDA WILLIAMS
BARBARA JONES
ELIZABETH BROWN
JENNIFER DAVIS
MARIA MILLER
SUSAN WILSON
MARGARET MOORE
LISA ANDERSON
NANCY THOMAS
KAREN JACKSON
BETTY WHITE
HELEN HARRIS
SANDRA MARTIN
DOMINA THOMPSON
CAROL GARCIA
RUTH MARTINEZ
MICHELLE CLARK
LAURA RODRIGUEZ
SARAH LEWIS
KIMBERLY LEE
DEBORAH WALKER
JESSICA HALL
SHIRLEY ALLEN
CYNTHIA YOUNG
ANGELA HERNAN...
MELISSA KING
BRENDA WRIGHT
AMY LOPEZ
ANNA HILL
REBECCA SCOTT
VIRGINIA GREEN

Time	Action	Response	Duration / Fetch Time
21:10:55	SELECT DISTINCT CONC...	583 row(s) returned	0.0077 sec / 0.000047 sec

- 7、 查询演过《FIRE WOLVES》和《JAWBREAKER BROOKLYN》这两部电影的演员，列出其姓名；

SQL 语句：

```
SELECT CONCAT(first_name, ' ', last_name) as name
FROM actor
WHERE actor_id = (
    SELECT FA1.actor_id
    FROM film_actor FA1, film_actor FA2
    WHERE FA1.actor_id = FA2.actor_id
        AND FA1.film_id = (
            SELECT film_id FROM film WHERE title = 'FIRE WOLVES'
        )
        AND FA2.film_id = (
            SELECT film_id FROM film WHERE title = 'JAWBREAKER BROOKLYN'
        )
);
```

结果截图：

Result Grid				
Filter Rows: Search				
Export:				
name				
CHRISTIAN AKROYD				
Result 1	Result 2	Result 3	Result 4	Result 5
Action Output				
	Time	Action	Response	Duration / Fetch Time
7	21:10:55	SELECT CONCAT(first_na...	1 row(s) returned	0.00057 sec / 0.0000041 sec

- 8、 统计每种类型的影片数，显示类型编号、类型名称、该类型影片数；

SQL 语句：

```
SELECT category.category_id, category.name, COUNT(category.category_id)
as amount
FROM film_category, category
WHERE category.category_id = film_category.category_id
GROUP BY category.category_id, category.name;
```

结果截图：

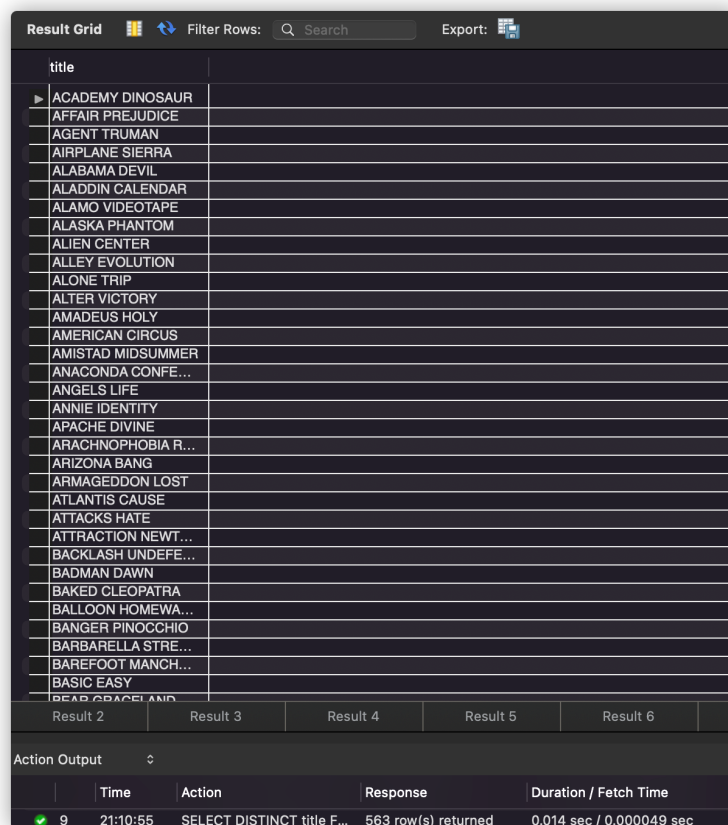
Result Grid				
Filter Rows: Search				
Export:				
category_id name amount				
1	Action	64		
2	Animation	66		
3	Children	60		
4	Classics	57		
5	Comedy	58		
6	Documentary	68		
7	Drama	62		
8	Family	69		
9	Foreign	73		
10	Games	61		
11	Horror	56		
12	Music	51		
13	New	63		
14	Sci-Fi	61		
15	Sports	74		
16	Travel	57		
Result 1	Result 2	Result 3	Result 4	Result 5
Action Output				
	Time	Action	Response	Duration / Fetch Time
8	21:10:55	SELECT category.categor...	16 row(s) returned	0.00077 sec / 0.0000062 sec

9、 有哪些影片是 2 个商店都有库存的？

SQL 语句：

```
SELECT DISTINCT title
FROM film, inventory I1, inventory I2
WHERE I1.film_id = I2.film_id
      AND I1.store_id <> I2.store_id
      AND I1.film_id = film.film_id;
```

结果截图：



title	
ACADEMY DINOSAUR	
AFFAIR PREJUDICE	
AGENT TRUMAN	
AIRPLANE SIERRA	
ALABAMA DEVIL	
ALADDIN CALENDAR	
ALAMO VIDEOTAPE	
ALASKA PHANTOM	
ALIEN CENTER	
ALLEY EVOLUTION	
ALONE TRIP	
ALTER VICTORY	
AMADEUS HOLY	
AMERICAN CIRCUS	
AMISTAD MIDSUMMER	
ANACONDA CONFE...	
ANGELS LIFE	
ANNIE IDENTITY	
APACHE DIVINE	
ARACHNOPHOBIA R...	
ARIZONA BANG	
ARMAGEDDON LOST	
ATLANTIS CAUSE	
ATTACKS HATE	
ATTRACTION NEWT...	
BACKLASH UNDEFE...	
BADMAN DAWN	
BAKED CLEOPATRA	
BALLOON HOMEWA...	
BANGER PINOCCHIO	
BARBARELLA STRE...	
BAREFOOT MANCH...	
BASIC EASY	
BEAR GRACELAND	

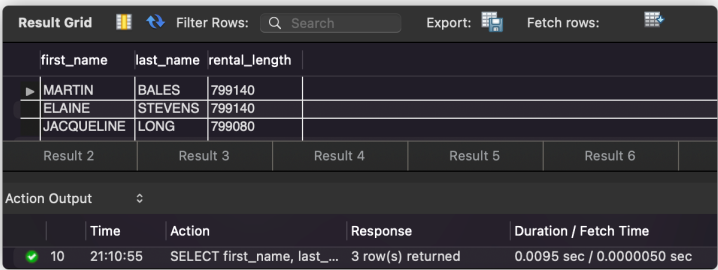
Time	Action	Response	Duration / Fetch Time
21:10:55	SELECT DISTINCT title F...	563 row(s) returned	0.014 sec / 0.000049 sec

10、 查询单次租借影片时间最长的 3 位客户，列出其 first\_name、last\_name 和当次租借时长；

SQL 语句：

```
SELECT first_name, last_name, TIMESTAMPDIFF(second , rental_date,
return_date) as rental_length
FROM customer, rental
WHERE customer.customer_id = rental.customer_id
ORDER BY rental_length DESC
LIMIT 3;
```

结果截图：



The screenshot shows a database client interface. At the top, there's a 'Result Grid' with columns: first\_name, last\_name, rental\_length. It contains three rows of data: MARTIN BALES 799140, ELAINE STEVENS 799140, and JACQUELINE LONG 799080. Below the grid is an 'Action Output' section with columns: Time, Action, Response, and Duration / Fetch Time. It shows a successful execution of a query at 21:10:55, returning 3 rows in 0.0095 seconds.

first_name	last_name	rental_length
MARTIN	BALES	799140
ELAINE	STEVENS	799140
JACQUELINE	LONG	799080

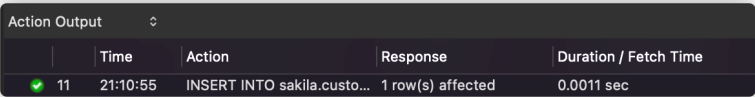
Time	Action	Response	Duration / Fetch Time
21:10:55	SELECT first_name, last_...	3 row(s) returned	0.0095 sec / 0.0000050 sec

11、 在 customer 表中新增一条数据，注意 customer 表与其他表的关系；

SQL 语句：

```
INSERT INTO sakila.customer (store_id, first_name, last_name, email, address_id, create_date)
VALUES (1, 'PINSKY', 'ROBIN', 'xu568059888@gmail.com', 6, NOW());
```

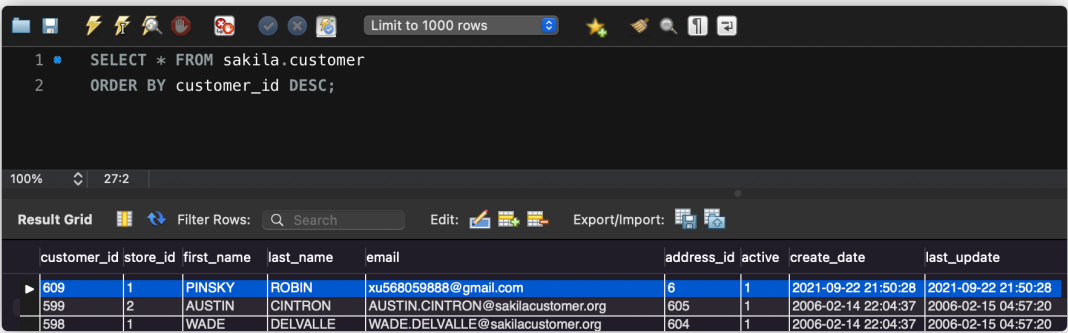
结果截图：



The screenshot shows the 'Action Output' section of a database client. It displays the execution of an INSERT statement at 21:10:55, which affected 1 row in 0.0011 seconds.

Time	Action	Response	Duration / Fetch Time
21:10:55	INSERT INTO sakila.custo...	1 row(s) affected	0.0011 sec

customer 表新增数据：



The screenshot shows a database client interface with a SQL query editor and a 'Result Grid'. The query is: SELECT \* FROM sakila.customer ORDER BY customer\_id DESC;. The result grid shows three rows of customer data, ordered by customer\_id descending. The first row is highlighted in blue.

```
1 SELECT * FROM sakila.customer
2 ORDER BY customer_id DESC;
```

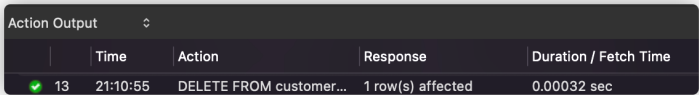
customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
609	1	PINSKY	ROBIN	xu568059888@gmail.com	6	1	2021-09-22 21:50:28	2021-09-22 21:50:28
599	2	AUSTIN	CINTRON	AUSTIN.CINTRON@sakilacustomer.org	605	1	2006-02-14 22:04:37	2006-02-15 04:57:20
598	1	WADE	DELVALLE	WADE.DELVALLE@sakilacustomer.org	604	1	2006-02-14 22:04:37	2006-02-15 04:57:20

12、 修改刚才在 customer 表中新增的那条数据；

SQL 语句：

```
UPDATE customer
SET active = 0
WHERE customer_id = 609;
```

结果截图：

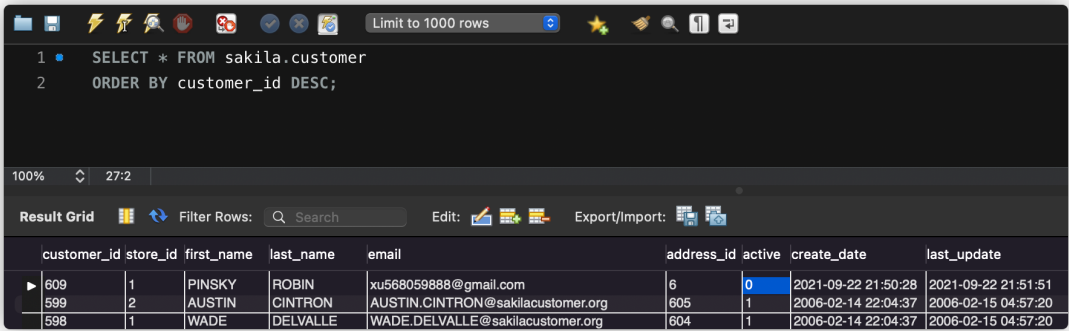


The screenshot shows the 'Action Output' section of a database client. It displays the execution of an UPDATE statement at 21:10:55, which affected 1 row in 0.00032 seconds.

Time	Action	Response	Duration / Fetch Time
21:10:55	DELETE FROM customer...	1 row(s) affected	0.00032 sec



customer 表修改数据:



The screenshot shows a database client interface with a SQL query editor at the top containing two lines of code: `1 SELECT * FROM sakila.customer` and `2 ORDER BY customer_id DESC;`. Below the editor, a 'Result Grid' displays the query results. The grid has columns for customer\_id, store\_id, first\_name, last\_name, email, address\_id, active, create\_date, and last\_update. The results are ordered by customer\_id in descending order, showing records for customer\_id 609, 599, and 598.

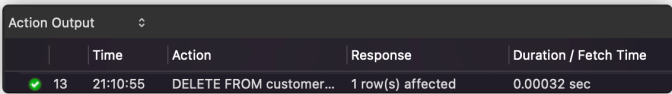
customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
609	1	PINSKY	ROBIN	xu568059888@gmail.com	6	0	2021-09-22 21:50:28	2021-09-22 21:51:51
599	2	AUSTIN	CINTRON	AUSTIN.CINTRON@sakilacustomer.org	605	1	2006-02-14 22:04:37	2006-02-15 04:57:20
598	1	WADE	DELVALLE	WADE.DELVALLE@sakilacustomer.org	604	1	2006-02-14 22:04:37	2006-02-15 04:57:20

13、 删除第 11 步新增的那条数据。

SQL 语句:

```
UPDATE customer
SET active = 0
WHERE customer_id = 609;
```

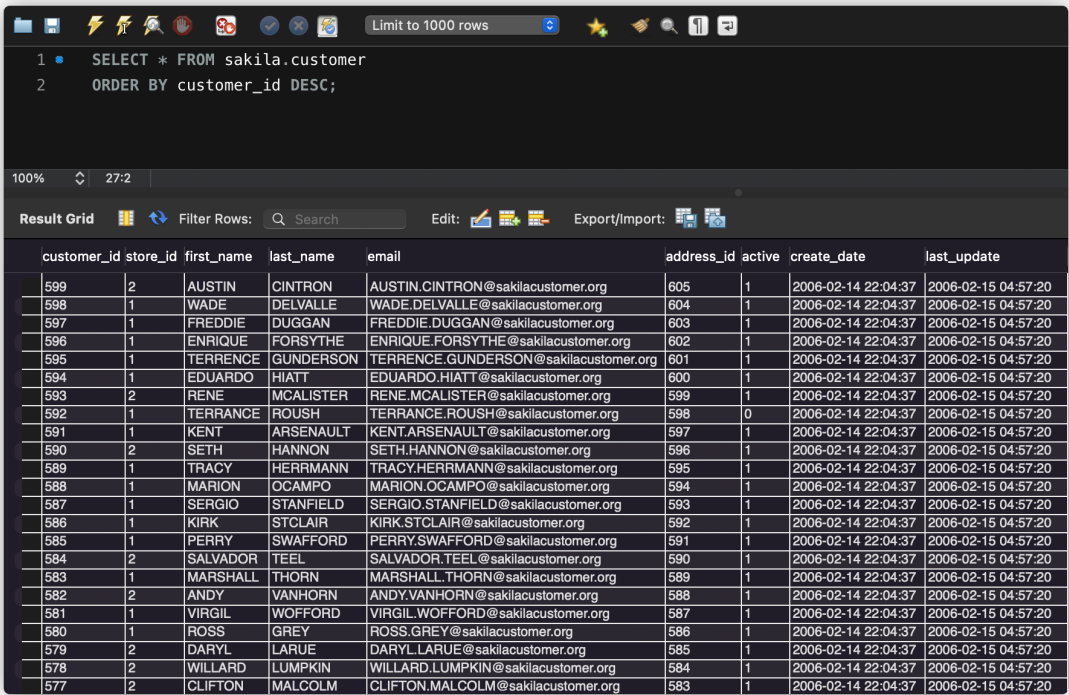
结果截图:



The screenshot shows an 'Action Output' window with a table containing one row of data. The row indicates that a DELETE operation was successful, affecting 1 row(s) with a duration of 0.00032 seconds.

	Time	Action	Response	Duration / Fetch Time
13	21:10:55	DELETE FROM customer...	1 row(s) affected	0.00032 sec

customer 表删除数据:



The screenshot shows a database client interface with a SQL query editor at the top containing two lines of code: `1 SELECT * FROM sakila.customer` and `2 ORDER BY customer_id DESC;`. Below the editor, a 'Result Grid' displays the query results. The grid has columns for customer\_id, store\_id, first\_name, last\_name, email, address\_id, active, create\_date, and last\_update. The results are ordered by customer\_id in descending order, showing records for customer\_id 599, 598, 597, 596, 595, 594, 593, 592, 591, 590, 589, 588, 587, 586, 585, 584, 583, 582, 581, 580, 579, 578, and 577.

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
598	1	WADE	DELVALLE	WADE.DELVALLE@sakilacustomer.org	604	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
596	1	ENRIQUE	FORSYTHE	ENRIQUE.FORSYTHE@sakilacustomer.org	602	1	2006-02-14 22:04:37	2006-02-15 04:57:20
595	1	TERRENCE	GUNDERSON	TERRENCE.GUNDERSON@sakilacustomer.org	601	1	2006-02-14 22:04:37	2006-02-15 04:57:20
594	1	EDUARDO	HIATT	EDUARDO.HIATT@sakilacustomer.org	600	1	2006-02-14 22:04:37	2006-02-15 04:57:20
593	2	RENE	MCALISTER	RENE.MCALISTER@sakilacustomer.org	599	1	2006-02-14 22:04:37	2006-02-15 04:57:20
592	1	TERRANCE	ROUSH	TERRANCE.ROUSH@sakilacustomer.org	598	0	2006-02-14 22:04:37	2006-02-15 04:57:20
591	1	KENT	ARSENAULT	KENT.ARSENAULT@sakilacustomer.org	597	1	2006-02-14 22:04:37	2006-02-15 04:57:20
590	2	SETH	HANNON	SETH.HANNON@sakilacustomer.org	596	1	2006-02-14 22:04:37	2006-02-15 04:57:20
589	1	TRACY	HERRMANN	TRACY.HERRMANN@sakilacustomer.org	595	1	2006-02-14 22:04:37	2006-02-15 04:57:20
588	1	MARION	OCAMPO	MARION.OCAMPO@sakilacustomer.org	594	1	2006-02-14 22:04:37	2006-02-15 04:57:20
587	1	SERGIO	STANFIELD	SERGIO.STANFIELD@sakilacustomer.org	593	1	2006-02-14 22:04:37	2006-02-15 04:57:20
586	1	KIRK	STCLAIR	KIRK.STCLAIR@sakilacustomer.org	592	1	2006-02-14 22:04:37	2006-02-15 04:57:20
585	1	PERRY	SWAFFORD	PERRY.SWAFFORD@sakilacustomer.org	591	1	2006-02-14 22:04:37	2006-02-15 04:57:20
584	2	SALVADOR	TEEL	SALVADOR.TEEL@sakilacustomer.org	590	1	2006-02-14 22:04:37	2006-02-15 04:57:20
583	1	MARSHALL	THORN	MARSHALL.THORN@sakilacustomer.org	589	1	2006-02-14 22:04:37	2006-02-15 04:57:20
582	2	ANDY	VANHORN	ANDY.VANHORN@sakilacustomer.org	588	1	2006-02-14 22:04:37	2006-02-15 04:57:20
581	1	VIRGIL	WOFFORD	VIRGIL.WOFFORD@sakilacustomer.org	587	1	2006-02-14 22:04:37	2006-02-15 04:57:20
580	1	ROSS	GREY	ROSS.GREY@sakilacustomer.org	586	1	2006-02-14 22:04:37	2006-02-15 04:57:20
579	2	DARYL	LARUE	DARYL.LARUE@sakilacustomer.org	585	1	2006-02-14 22:04:37	2006-02-15 04:57:20
578	2	LUMPKIN	WILLARD	WILLARD.LUMPKIN@sakilacustomer.org	584	1	2006-02-14 22:04:37	2006-02-15 04:57:20
577	2	CLIFTON	MALCOLM	CLIFTON.MALCOLM@sakilacustomer.org	583	1	2006-02-14 22:04:37	2006-02-15 04:57:20

### 三、思考题

- 1) 如果 insert 一条数据到 actor 表，但 actor\_id 和已有数据重复，会发生什么？同学们请自己尝试一下。

考虑执行如下语句：

```
INSERT INTO actor (actor_id, first_name, last_name, last_update)
VALUES (1, 'PINSKY', 'ROBIN', '2021-09-21 16:10:17')
```

执行结果如下：

```
[2021-09-21 16:10:21] [23000][1062] Duplicate entry '1' for key
'actor.PRIMARY'
[2021-09-21 16:10:21] [23000][1062] Duplicate entry '1' for key
'actor.PRIMARY'
```

表示插入的数据的字段与现有某行数据的字段重复，插入失败。

- 2) insert 语句还用了一个函数 NOW()，是做什么的呢？

NOW()函数以“YYYY-MM-DD HH:mm:ss” 的格式返回该条指令执行时的时间。