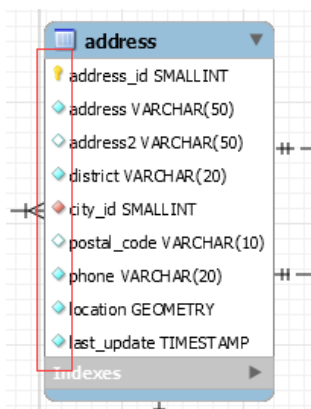


实验一报告

一、回答问题

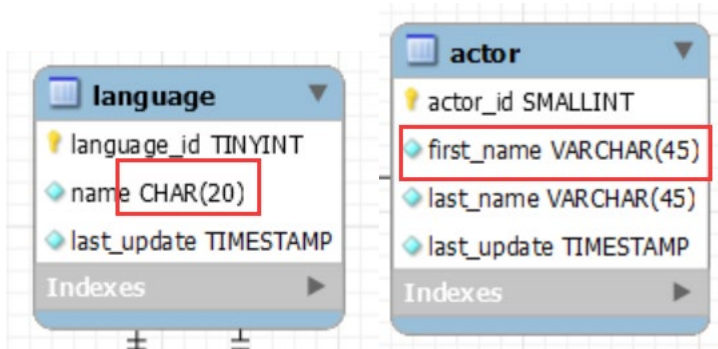
请一边熟悉 sakila 数据库，一边回答以下问题：

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？
(观察字段的属性)



标记	意义
	Primary key
	not null
	其他
	说明该 attribute 和其他表共享

2. char 和 varchar 类型的区别是什么？



- 1.CHAR 的长度是固定的，无论其内部的存储了多少个字符，例 CHAR(4) 的存储大小一直是 4 个 byte
- 2.VARCHAR 的长度不是固定的，和 CHAR 不同。
- 3.VARCHAR 的长度上限比 CHAR 更大，VARCHAR 最大有 65535，CHAR 最

大有 255。

3. 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？

表 `film_actor` 直接体现了影片和演员关系。其内部存储的是 `film` 的 `id` 和演员的 `id`。

`film` 名称和演员名称分别存储在 `film` 表和 `actor` 表内。

若要根据影片名得到演员的名字，需要访问 `film`, `film_actor`, `actor` 这三张表。

4. 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？

查找以下几个表：`customer`, `rental`, `inventory`, `film`

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）

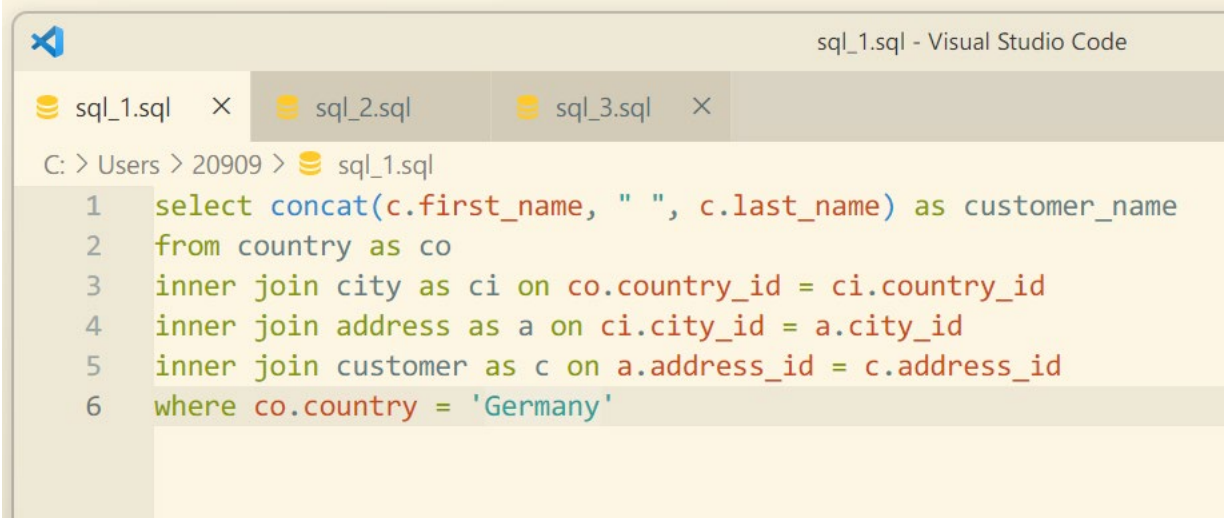
1、请列出所有 `country` 是“Germany”的客户的信息，显示 `customer_id`、客户姓名、地址、所在区域，所在城市（注意：客户姓名请以 `first_name+空格+last_name` 的格式，例如：SISSY SOBIESKI）；

```
select concat(c.first_name, " ", c.last_name) as customer_name
from country as co
inner join city as ci on co.country_id = ci.country_id
inner join address as a on ci.city_id = a.city_id
inner join customer as c on a.address_id = c.address_id
where co.country = 'Germany'
```

```
mysql> source C:/Users/20909/sql_1.sql
```

customer_name
GRACE ELLIS
MIGUEL BETANCOURT
VICKIE BREWER
ALMA AUSTIN
COLLEEN BURTON
VANESSA SIMS
VICKI FIELDS

7 rows in set (0.00 sec)



```
1 select concat(c.first_name, " ", c.last_name) as customer_name
2 from country as co
3 inner join city as ci on co.country_id = ci.country_id
4 inner join address as a on ci.city_id = a.city_id
5 inner join customer as c on a.address_id = c.address_id
6 where co.country = 'Germany'
```

2、 列出属于“Music”类型并以“A”开头的电影名；

```
select title from film as f
, category as c
, film_category as fc
where f.film_id = fc.film_id
and fc.category_id = c.category_id
and left(f.title, 1) = 'A';
```

```
mysql> source C:/Users/20909/sql_2.sql
+-----+
| title |
+-----+
| AMADEUS HOLY |
| AMERICAN CIRCUS |
| ANTITRUST TOMATOES |
| ARK RIDGEMONT |
| ALTER VICTORY |
| ANACONDA CONFESSIONS |
| ARGONAUTS TOWN |
| ALICE FANTASIA |
| ARIZONA BANG |
+-----+

C: > Users > 20909 > sql_2.sql
1 select title from film as f
2   , category as c
3   , film_category as fc
4 where f.film_id = fc.film_id
5 and fc.category_id = c.category_id
6 and left(f.title, 1) = 'A';

+-----+
| ARMAGEDDON LOST |
| ATTACKS HATE |
| ALADDIN CALENDAR |
| ANONYMOUS HUMAN |
| ARTIST COLDBLOODED |
| ARSENIC INDEPENDENCE |
+-----+
46 rows in set (0.01 sec)
```

- 3、找出租 DVD 花费的总费用在 160 至 170 之间的客户，列出他们的 first_name, last_name 和每个人花费的金额；

```
select c.first_name, c.last_name, sum(p.amount)
from customer as c
inner join payment as p on c.customer_id = p.customer_id
group by c.customer_id
having sum(p.amount) > 160 and sum(p.amount) < 170;
```

SQL File 3"

```

1 select c.first_name, c.last_name, sum(p.amount)
2 from customer as c
3 inner join payment as p on c.customer_id = p.customer_id
4 group by c.customer_id
5 having sum(p.amount) > 160 and sum(p.amount) < 170;

```

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

first_name	last_name	sum(p.amount)
DIANE	COLLINS	169.65
TONYA	CHAPMAN	161.68
DAISY	BATES	162.62
LOUIS	LEONE	161.65
MIKE	WAY	166.65
CURTIS	IRBY	167.62
GORDON	ALLARD	160.68
ARNOLD	HAVENS	167.67

Result 6 x

Output

Action Output

#	Time	Action	Message
11	20:14:41	select distinct title from film, category where name = '...	46 row(s) returned
12	20:18:45	select c.first_name, c.last_name, sum(p.amount) from ...	8 row(s) returned

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；

```

select f.film_id AS id, f.title AS title, sum(p.amount) AS total_sales
from payment AS p
inner join rental AS r ON p.rental_id = r.rental_id
inner join inventory AS i ON r.inventory_id = i.inventory_id
inner join film AS f ON i.film_id = f.film_id
group by f.film_id
order by sum(p.amount) DESC
limit 1;

```

SQL File 3* x

Limit to 1000 rows

```

2  from payment AS p
3  inner join rental AS r ON p.rental_id = r.rental_id
4  inner join inventory AS i ON r.inventory_id = i.inventory_id
5  inner join film AS f ON i.film_id = f.film_id
6  group by f.film_id
7  order by sum(p.amount) DESC
8  limit 1;

```

Result Grid

	id	title	total_sales
▶	879	TELEGRAPH VOYAGE	231.73

Export: | Wrap Cell Content: | Fetch rows:

Result 7 x

Output

Action Output

	#	Time	Action	Message
✓	12	20:18:45	select c.first_name, c.last_name, sum(p.amount) from ...	8 row(s) returned
✓	13	20:23:53	select f.film_id AS id, f.title AS title, sum(p.amount) AS ...	1 row(s) returned

5、 哪些演员出演的电影超过 40 部？ 请列出演员名、出演的电影数；

```

select a.first_name AS first_name
, a.last_name AS last_name
, count(f.film_id) AS total_films
from actor AS a
inner join film_actor AS fa ON a.actor_id = fa.actor_id
inner join film AS f ON fa.film_id = f.film_id
group by a.actor_id
having count(f.film_id) > 40
order by count(f.film_id) DESC;

```

SQL File 3*

```

1 • select a.first_name AS first_name
2   , a.last_name AS last_name
3   , count(f.film_id) AS total_films
4 from actor AS a
5 inner join film_actor AS fa ON a.actor_id = fa.actor_id
6 inner join film AS f ON fa.film_id = f.film_id
7 group by a.actor_id
8 having count(f.film_id) > 40
9 order by count(f.film_id) DESC;

```

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

	first_name	last_name	total_films
▶	GINA	DEGENERES	42
	WALTER	TORN	41

Result 8

Output

Action Output

#	Time	Action	Message
13	20:23:53	select f.film_id AS id, f.title AS title, sum(p.amount) AS ...	1 row(s) returned
14	20:25:11	select a.first_name AS first_name , a.last_name AS la...	2 row(s) returned

6、 请找出没有租借过电影《NATURAL STOCK》的顾客姓名；

```

select c.first_name AS first_name
, c.last_name AS last_name
from customer AS c
where not exists (
    select *
    from rental AS r
    inner join inventory AS i ON r.inventory_id = i.inventory_id
    inner join film AS f ON i.film_id = f.film_id
    where f.film_id = 'NATURAL STOCK' and r.customer_id = c.customer_id
);

```


SQL File 3* x

Limit to 1000 rows

```

1 • select c.first_name AS first_name
2   , c.last_name AS last_name
3   from customer AS c
4   where not exists (
5       select *
6       from rental AS r
7       inner join inventory AS i ON r.inventory_id = i.inventory_id
8       inner join film AS f ON i.film_id = f.film_id
9       where f.film_id = 'NATURAL STOCK' and r.customer_id = c.customer_id
10  );

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [FA](#)

	first_name	last_name
▶	MARY	SMITH
	PATRICIA	JOHNSON
	LINDA	WILLIAMS
	BARBARA	JONES
	ELIZABETH	BROWN
	JENNIFER	DAVIS

customer 11 x

Output

Action Output

	#	Time	Action	Message
✓	16	20:28:03	select c.first_name AS first_name , c.last_name AS la...	600 row(s) returned
✓	17	20:28:13	select c.first_name AS first_name , c.last_name AS la...	600 row(s) returned

7、 查询既演过《ELEPHANT TROJAN》又演过《DOGMA FAMILY》的演员，列出其姓名；

```

select a.first_name AS first_name
, a.last_name AS last_name
from actor AS a
inner join film_actor AS fa ON a.actor_id = fa.actor_id
inner join film AS f ON fa.film_id = f.film_id
where f.title = 'ELEPHANT TROJAN' or f.title = 'DOGMA FAMILY'
group by a.actor_id
having count(f.film_id) = 2;

```


SQL File 3* x

Limit to 1000 rows

```

1 • select a.first_name AS first_name
2   , a.last_name AS last_name
3   from actor AS a
4   inner join film_actor AS fa ON a.actor_id = fa.actor_id
5   inner join film AS f ON fa.film_id = f.film_id
6   where f.title = 'ELEPHANT TROJAN' or f.title = 'DOGMA FAMILY'
7   group by a.actor_id
8   having count(f.film_id) = 2;

```

Result Grid

	first_name	last_name
▶	GINA	DEGENERES

Result 12 x

Output

Action Output

	#	Time	Action	Message
✓	17	20:28:13	select c.first_name AS first_name , c.last_name AS la...	600 row(s) returned
✓	18	20:28:46	select a.first_name AS first_name , a.last_name AS la...	1 row(s) returned

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；

```

select c.name AS category_name
, c.category_id AS category_id
, count(f.film_id) AS total_films
from film AS f
inner join film_category AS fc ON f.film_id = fc.film_id
inner join category AS c ON fc.category_id = c.category_id
group by c.category_id;

```

Limit to 1000 rows

```

1 • select c.name AS category_name
2     , c.category_id AS category_id
3     , count(f.film_id) AS total_films
4 from film AS f
5 inner join film_category AS fc ON f.film_id = fc.film_id
6 inner join category AS c ON fc.category_id = c.category_id
7 group by c.category_id;

```

category_name	category_id	total_films
Action	1	64
Animation	2	66
Children	3	60
Classics	4	57
Comedy	5	58
Documentary	6	68
Drama	7	62
Family	8	69
Foreign	9	73
Games	10	61

Result 13 x

Output

Action Output

#	Time	Action	Message
18	20:28:46	select a.first_name AS first_name , a.last_name AS la...	1 row(s) returned
19	20:29:32	select c.name AS category_name , c.category_id AS ...	16 row(s) returned

9、 有哪些影片是 2 个商店都有库存的？显示影片名。

```

select f.film_id AS film_id
from film AS f
inner join inventory AS i ON f.film_id = i.film_id
where not exists (
    select *
    from rental AS r
    where r.return_date is null and i.inventory_id = r.inventory_id
)
or not exists(
    select *
    from rental AS r
    where r.inventory_id = r.inventory_id
)
group by f.film_id

```

having count(distinct i.store_id) = 2;

```
1 • select f.film_id AS film_id
2   from film AS f
3   inner join inventory AS i ON f.film_id = i.film_id
4   where not exists (
5       select *
6       from rental AS r
7       where r.return_date is null and i.inventory_id = r.inventory_id
8   )
9   or not exists(
10      select *
11      from rental AS r
12      where r.inventory_id = r.inventory_id
13  )
14  group by f.film_id
15  having count(distinct i.store_id) = 2;
```

Result Grid

film_id
73
74
77
78
79

Result 14 x

Output

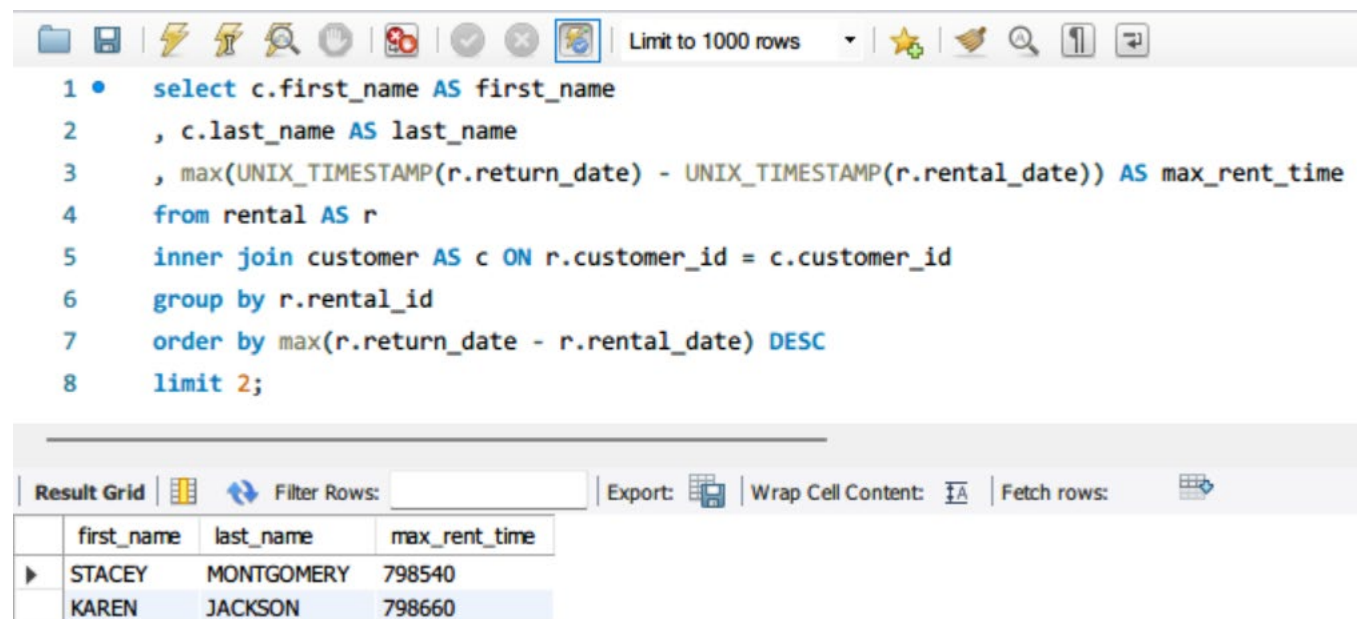
Action Output

#	Time	Action	Message
19	20:29:32	select c.name AS category_name , c.category_id AS ...	16 row(s) returned
20	20:30:10	select f.film_id AS film_id from film AS f inner join inven...	562 row(s) returned

10、 查询单次租借影片时间最长的 2 位客户，列出其 first_name、last_name 和当次租借时长（单位秒）；

```
select c.first_name AS first_name
, c.last_name AS last_name
, max(UNIX_TIMESTAMP(r.return_date) - UNIX_TIMESTAMP(r.rental_date)) AS max_rent_time
from rental AS r
inner join customer AS c ON r.customer_id = c.customer_id
group by r.rental_id
order by max(r.return_date - r.rental_date) DESC
```

limit 2;

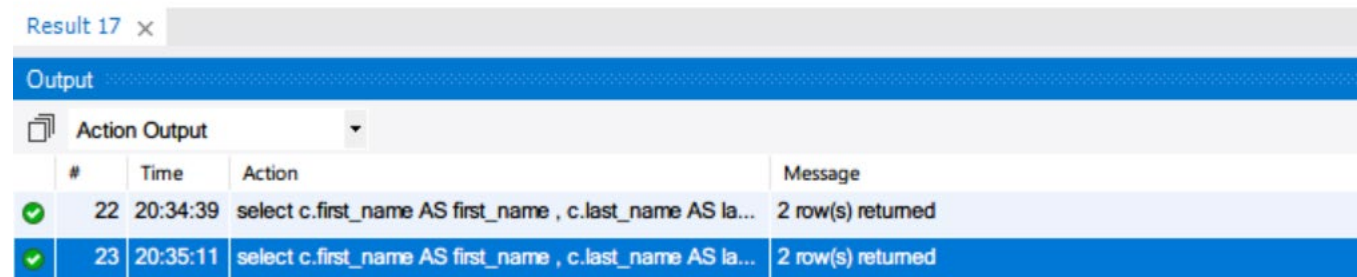


The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to "Limit to 1000 rows". The SQL editor contains the following query:

```
1 • select c.first_name AS first_name
2     , c.last_name AS last_name
3     , max(UNIX_TIMESTAMP(r.return_date) - UNIX_TIMESTAMP(r.rental_date)) AS max_rent_time
4 from rental AS r
5 inner join customer AS c ON r.customer_id = c.customer_id
6 group by r.rental_id
7 order by max(r.return_date - r.rental_date) DESC
8 limit 2;
```

Below the editor, the "Result Grid" tab is active, displaying the query results in a table:

	first_name	last_name	max_rent_time
▶	STACEY	MONTGOMERY	798540
	KAREN	JACKSON	798660



The screenshot shows the "Output" window with the "Action Output" tab selected. It displays a log of database actions:

#	Time	Action	Message
✓ 22	20:34:39	select c.first_name AS first_name , c.last_name AS la...	2 row(s) returned
✓ 23	20:35:11	select c.first_name AS first_name , c.last_name AS la...	2 row(s) returned

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

```
insert into customer (store_id, first_name, last_name, address_id, create_date, last_update)
values(1, 'walter', 'white', 1, NOW(), NOW());
```

The screenshot shows a SQL IDE window. The top toolbar includes icons for file operations, execution, and search. The main text area contains the following SQL code:

```
1 • insert into customer (store_id, first_name, last_name, address_id, create_date)
2 values(1, 'Walter', 'White', 1, NOW());
3 -- last_updated is set automatically
```

Below the code editor is an 'Output' panel. It has a dropdown menu set to 'Action Output'. The output is displayed in a table with four columns: #, Time, Action, and Message.

#	Time	Action	Message
23	20:35:11	select c.first_name AS first_name , c.last_name AS la...	2 row(s) returned
24	20:36:58	insert into customer (store_id, first_name, last_name, ...	1 row(s) affected

12、 修改刚才在 customer 表中新增的那条数据；

```
update customer
```

```
set last_update = NOW()
```

```
where customer_id = 600;
```

```
insert rental (rental_date, inventory_id, customer_id, return_date, staff_id)
```

```
values (now(), 1, 600, null, 1);
```

```
-- check insert result
```

```
select c.customer_id, c.first_name, c.last_name, r.rental_id
```

```
from customer as c
```

```
inner join rental as r on c.customer_id = r.customer_id
```

```
where c.customer_id = 601;
```

SQL File 3* x

Limit to 1000 rows

```

1 • update customer
2   set last_update = NOW()
3   where customer_id = 601;
4
5 • insert rental (rental_date, inventory_id, customer_id, return_date, staff_id)
6   values (now(), 1, 601, null, 1);
7   -- check insert result
8 • select c.customer_id, c.first_name, c.last_name, r.rental_id
9   from customer as c
10  inner join rental as r on c.customer_id = r.customer_id
11  where c.customer_id = 601;

```

Result Grid

customer_id	first_name	last_name	rental_id
601	Walter	White	16054

Result 19 x

Output

Action Output

#	Time	Action	Message
34	20:42:18	insert rental (rental_date, inventory_id, customer_id, r...	1 row(s) affected
35	20:42:18	select c.customer_id, c.first_name, c.last_name, r.rent...	1 row(s) returned

13、 删除第 11 步新增的那条数据。

```

delete from rental
where customer_id = 601;

```

```

delete from customer
where customer_id = 601;

```

```

-- check
select count(*) from customer
where customer_id = 601;

```


SQL File 3* x

Limit to 1000 rows

```

1 • delete from rental
2   where customer_id = 601;
3   -- 需先删除 rental 记录
4   delete from customer
5   where customer_id = 601;
6
7   -- check
8   select count(*) from customer
9   where customer_id = 601;

```

Result Grid

	count(*)
▶	0

Export: Wrap Cell Content:

Result 21 x

Output

Action Output

#	Time	Action	Message
✓ 38	20:47:57	delete from customer where customer_id = 601	1 row(s) affected
✓ 39	20:47:57	select count(*) from customer where customer_id = 601	1 row(s) returned

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，截图并分析原因。

```

mysql> insert actor (actor_id, first_name, last_name)
      → values (1, 'Jesse', 'Pinkman');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

```

Primary Key 不能有重复值。

- 2) insert 语句还用了一个函数 NOW()，是做什么的呢？

更新 last_updated 的值。