

牛客网：数据库 sql 实战

```
mysql> desc employees;
```

Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO	PRI	NULL	
birth_date	date	NO		NULL	
first_name	varchar(14)	NO		NULL	
last_name	varchar(16)	NO		NULL	
gender	char(1)	NO		NULL	
hire_date	date	NO		NULL	

1、查找最晚入职员工的所有信息

```
select * from employees where hire_date = (select max(hire_date) from employees);
```

2、查找入职员工时间排名倒数第三的员工所有

```
select * from employees where hire_date = (select hire_date from employees order by  
hire_date desc limit 2, 1);
```

limit m, n 表示从第 m+1 条开始，取 n 条数据

```
mysql> desc dept_manager;
```

Field	Type	Null	Key	Default	Extra
dept_no	char(4)	NO	PRI	NULL	
emp_no	int(11)	NO	PRI	NULL	
from_date	date	NO		NULL	
to_date	date	NO		NULL	

```
mysql> desc salaries;
```

Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO	PRI	NULL	
salary	int(11)	NO		NULL	
from_date	date	NO	PRI	NULL	
to_date	date	NO		NULL	

3、查找各个部门当前(to_date='9999-01-01')领导当前薪水详情以及其对应部门编号 dept_no

```
select s.*, d.dept_no from salaries s, dept_manager d  
where s.emp_no = d.emp_no and s.to_date = '9999-01-01' and d.to_date = '9999-01-01';
```

```
mysql> desc dept_emp;
```

Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO	PRI	NULL	
dept_no	char(4)	NO	PRI	NULL	
from_date	date	NO		NULL	
to_date	date	NO		NULL	

4、查找所有已经分配部门的员工的 last_name 和 first_name

内连接:

```
select e.last_name, e.first_name, d.dept_no from employees e inner join dept_emp d on
e.emp_no = d.emp_no;
```

自然连接:

```
select last_name, first_name, dept_no from dept_emp natural join employees;
```

5、查找所有员工的 last_name 和 first_name 以及对应部门编号 dept_no，也包括展示没有分配具体部门的员工

```
select e.last_name, e.first_name, d.dept_no from employees e left join dept_emp d on
d.emp_no = e.emp_no;
```

6、查找所有员工入职时候的薪水情况，给出 emp_no 以及 salary， 并按照 emp_no 进行逆序

```
select e.emp_no, s.salary from employees e, salaries s
where e.emp_no = s.emp_no and e.hire_date = s.from_date order by e.emp_no desc;
```

7、查找薪水涨幅超过 15 次的员工号 emp_no 以及其对应的涨幅次数 t

```
select emp_no, count(*) from salaries group by emp_no having count(*) > 15;
```

8、找出所有员工当前(to_date='9999-01-01')具体的薪水 salary 情况，对于相同的薪水只显示一次,并按照逆序显示

```
select distinct salary from salaries where to_date = '9999-01-01' order by salary desc;
```

9、获取所有部门当前 manager 的当前薪水情况，给出 dept_no,emp_no 以及 salary，当前表示 to_date='9999-01-01'

```
select d.dept_no, d.emp_no, s.salary from salaries s, dept_manager d
where d.emp_no = s.emp_no and d.to_date = '9999-01-01' and s.to_date = '9999-01-01';
```

10、获取所有非 manager 的员工 emp_no

```
select e.emp_no from employees e where e.emp_no not in (select emp_no from
dept_manager);
```

11、获取所有员工当前的 manager，如果当前的 manager 是自己的话结果不显示，当前

表示 to_date='9999-01-01'。结果第一列给出当前员工的 emp_no,第二列给出其 manager 对应的 manager_no。

```
select d.emp_no, m.emp_no from dept_emp d, dept_manager m where d.dept_no = m.dept_no and d.emp_no <> m.emp_no and d.to_date = '9999-01-01' and m.to_date = '9999-01-01';
```

12、获取所有部门中当前员工薪水最高的相关信息，给出 dept_no, emp_no 以及其对应的 salary

```
select d.dept_no, s.emp_no, max(s.salary) as salary from dept_emp d, salaries s
where s.emp_no = d.emp_no and d.to_date = '9999-01-01' and s.to_date = '9999-01-01'
group by d.dept_no;
```

```
mysql> desc titles;
```

Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO		NULL	
title	varchar(50)	NO		NULL	
from_date	date	NO		NULL	
to_date	date	YES		NULL	

13、从 titles 表获取按照 title 进行分组，每组个数大于等于 2，给出 title 以及对应的数目 t。

```
select title, count(*) from titles group by title having count(*) >= 2;
```

14、从 titles 表获取按照 title 进行分组，每组个数大于等于 2，给出 title 以及对应的数目 t，注意对于重复的 emp_no 进行忽略。

```
select distinct title, count(distinct emp_no) t from titles group by title having t >= 2;
```

15、查找 employees 表所有 emp_no 为奇数，且 last_name 不为 Mary 的员工信息，并按照 hire_date 逆序排列

```
select * from employees where emp_no%2 = 1 and last_name <> 'Mary' order by hire_date desc;
```

16、统计出当前各个 title 类型对应的员工当前薪水对应的平均工资。结果给出 title 以及平均工资 avg。

```
select t.title, avg(s.salary) from titles t, salaries s
where t.emp_no = s.emp_no and t.to_date = '9999-01-01' and s.to_date = '9999-01-01'
group by t.title;
```

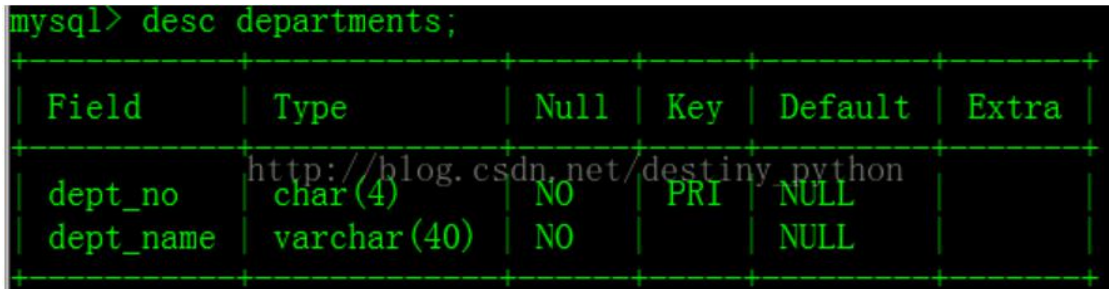
17、获取当前 (to_date='9999-01-01') 薪水第二多的员工的 emp_no 以及其对应的薪水 salary

```
select emp_no, salary from salaries where to_date = '9999-01-01' order by salary desc limit
```

1,1;

18、查找当前薪水(to_date='9999-01-01')排名第二多的员工编号 emp_no、薪水 salary、last_name 以及 first_name，不准使用 order by

```
select e.emp_no, max(s.salary) as salary, e.last_name, e.first_name from employees e,
salaries s
where e.emp_no = s.emp_no and s.to_date = '9999-01-01' and s.salary != (select max(salary)
from salaries where to_date = '9999-01-01');
```



```
mysql> desc departments;
```

Field	Type	Null	Key	Default	Extra
dept_no	char(4)	NO	PRI	NULL	
dept_name	varchar(40)	NO		NULL	

19、查找所有员工的 last_name 和 first_name 以及对应的 dept_name，也包括暂时没有分配部门的员工

```
select e.last_name, e.first_name, dm.dept_name from employees e left join dept_emp d on
e.emp_no = d.emp_no left join departments dm on d.dept_no = dm.dept_no;
```

20、查找员工编号 emp_no 为 10001 其自入职以来的薪水 salary 涨幅值 growth

```
select (select salary from salaries where emp_no = 10001 order by to_date desc limit 1) -
(select salary from salaries where emp_no = 10001 order by to_date limit 1) growth;
```

21、查找所有员工自入职以来的薪水涨幅情况，给出员工编号 emp_no 以及其对应的薪水涨幅 growth，并按照 growth 进行升序

```
select sta.emp_no, (cur.salary - sta.salary) growth from
(select e.emp_no, s.salary from employees e, salaries s where s.emp_no = e.emp_no and
s.to_date = '9999-01-01') cur, #查找目前的工资
(select e.emp_no, s.salary from employees e, salaries s where s.from_date = e.hire_date and
s.emp_no = e.emp_no) sta #查找入职时候的工资
where cur.emp_no = sta.emp_no order by growth;
```

22、统计各个部门对应员工涨幅的次数总和，给出部门编码 dept_no、部门名称 dept_name 以及次数 sum

```
select dm.dept_no, dm.dept_name, count(s.salary) sum from
salaries s inner join dept_emp d on s.emp_no = d.emp_no
inner join departments dm on d.dept_no = dm.dept_no
group by d.dept_no;
```

23、对所有员工的当前(to_date='9999-01-01')薪水按照 salary 进行按照 1-N 的排名，相同 salary 并列且按照 emp_no 升序排列

```
select s1.emp_no, s1.salary, count(distinct s2.salary) rank
```

```

from salaries s1, salaries s2
where s1.to_date = '9999-01-01' and s2.to_date = '9999-01-01'
and s1.salary <= s2.salary
group by s1.emp_no
order by s1.salary desc, s1.emp_no;

```

24、获取所有非 manager 员工当前的薪水情况，给出 dept_no、emp_no 以及 salary，当前表示 to_date='9999-01-01'

```

select d.dept_no, e.emp_no, s.salary
from employees e inner join salaries s on e.emp_no = s.emp_no and s.to_date =
'9999-01-01'
inner join dept_emp d on s.emp_no = d.emp_no
where d.emp_no not in (select emp_no from dept_manager);

```

25、获取员工其当前的薪水比其 manager 当前薪水还高的相关信息，当前表示 to_date='9999-01-01', 结果第一列给出员工的 emp_no，第二列给出其 manager 的 manager_no，第三列给出该员工当前的薪水 emp_salary, 第四列给该员工对应的 manager 当前的薪水 manager_salary

```

select s1.emp_no emp_no, s2.emp_no manager_no, s1.salary emp_salary, s2.salary
manager_salary
from (select d.emp_no, d.dept_no, s.salary from dept_emp d, salaries s
where d.emp_no = s.emp_no and s.to_date = '9999-01-01' and
d.emp_no not in (select emp_no from dept_manager)) s1,
(select dm.emp_no, dm.dept_no, s.salary from dept_manager dm, salaries s
where dm.emp_no = s.emp_no and s.to_date = '9999-01-01') s2
where s1.salary > s2.salary and s1.dept_no=s2.dept_no;

```

26、汇总各个部门当前员工的 title 类型的分配数目，结果给出部门编号 dept_no、dept_name、其当前员工所有的 title 以及该类型 title 对应的数目 count

```

select dm.dept_no, dm.dept_name, t.title, count(t.title) count
from dept_emp d inner join titles t on d.emp_no = t.emp_no and d.to_date = '9999-01-01'
and t.to_date = '9999-01-01'
inner join departments dm on d.dept_no = dm.dept_no
group by d.dept_no, t.title;

```

27、给出每个员工每年薪水涨幅超过 5000 的员工编号 emp_no、薪水变更开始日期 from_date 以及薪水涨幅值 salary_growth，并按照 salary_growth 逆序排列。

提示：在 sqlite 中获取 datetime 时间对应的年份函数为 strftime('%Y', to_date)

```

select s2.emp_no, s1.from_date, (s1.salary - s2.salary) salary_growth
from salaries s2, salaries s1
where s2.emp_no = s1.emp_no
and salary_growth > 5000
and ((strftime("%Y", s1.to_date) - strftime("%Y", s2.to_date)) = 1

```

```
or (strftime("%Y", s1.from_date) - strftime("%Y", s2.from_date)) = 1)
order by salary_growth desc;
```

表 film		表 category		表 film_category	
字段	说明	字段	说明	字段	说明
<u>film_id</u>	电影 id	<u>category_id</u>	电影分类 id	<u>film_id</u>	电影 id
Title	电影名称	name	电影分类名称	<u>category_id</u>	电影分类 id
description	电影描述信息	<u>last_update</u>	电影分类最后更新时间		电影 id 和分类 id 对应关系最后更新时间

28、查找描述信息中包含 robot 的电影对应的分类名称以及电影数目，而且还需要该分类对应电影数量>=2 部

```
select c.name, count(f.film_id) count
from (film f inner join film_category fc on f.film_id = fc.film_id)
inner join category c on c.category_id = fc.category_id
where f.description like '%robot%'
group by c.name
having count>=2;
```

29、使用 join 查询方式找出没有分类的电影 id 以及名称

```
select f.film_id, f.title from
film f left join film_category fc on f.film_id = fc.film_id
where fc.category_id is null;
```

30、使用子查询的方式找出属于 Action 分类的所有电影对应的 title,description

```
select f.title, f.description from
(select fc.film_id from film_category fc, category c
where fc.category_id = c.category_id and c.name = 'Action') a,
film f where f.film_id = a.film_id;
```

31、获取 select * from employees 对应的执行计划

```
explain select * from employees;
```

32、将 employees 表的所有员工的 last_name 和 first_name 拼接起来作为 Name，中间以一个空格区分

```
select last_name || ' ' || first_name name from employees;
```

33、创建一个 actor 表，包含如下列信息

列表	类型	是否为 NUL	含义
<u>actor_id</u>	<u>smallint(5)</u>	not null	主键 id
<u>first_name</u>	<u>varchar(45)</u>	not null	名字
<u>last_name</u>	<u>varchar(45)</u>	not null	姓氏
<u>last_update</u>	timestamp	not null	最后更新时间，默认是系统的当前时间

create table if not exists actor

```
(actor_id smallint(5) not null,  
first_name varchar(45) not null,  
last_name varchar(45) not null,  
last_update timestamp not null default (datetime('now','localtime'))  
PRIMARY KEY(actor_id));
```

34、对于表 actor 批量插入如下数据

```
insert into actor values(1,'PENELOPE','GUINNESS','2006-02-15 12:34:33'),  
(2,'NICK','WAHLBERG','2006-02-15 12:34:33');
```

35、对于表 actor 批量插入如下数据,如果数据已经存在,请忽略,不使用 replace 操作

```
insert OR ignore into actor values(3,'ED','CHASE','2006-02-15 12:34:33');
```

#备注: mysql 中需要删除 or

36、创建一个 actor_name 表,将 actor 表中的所有 first_name 以及 last_name 导入改表。

列表	类型	是否为 NULL	含义
first_name	varchar(45)	not null	名字
last_name	varchar(45)	not null	姓氏

create table

actor_name(first_name varchar(45) not null,

last_name varchar(45) not null);

```
insert into actor_name select first_name, last_name from actor;
```

37、针对表 actor 中 first_name 创建唯一索引 uniq_idx_firstname, 对 last_name 创建普通索引 idx_lastname

```
CREATE UNIQUE INDEX uniq_idx_firstname ON actor(first_name);
```

```
CREATE INDEX idx_lastname ON actor(last_name);
```

38、针对 actor 表创建视图 actor_name_view, 只包含 first_name 以及 last_name 两列, 并对这两列重新命名, first_name 为 first_name_v, last_name 修改为 last_name_v:

```
create view actor_name_view as select first_name first_name_v, last_name last_name_v  
from actor;
```

39、针对 salaries 表 emp_no 字段创建索引 idx_emp_no, 查询 emp_no 为 10005, 使用强制索引。

```
select * from salaries indexed by idx_emp_no where emp_no = 10005;
```

#在 mysql 中用 force index select * from salaries force index idx_emp_no where emp_no = 10005;

40、针对 actor 表,现在在 last_update 后面新增加一列名字为 create_date,类型为 datetime, NOT NULL, 默认值为'0000-00-00 00:00:00'

```
alter table actor add create_date datetime not null default '0000-00-00 00:00:00';
```

41、构造一个触发器 audit_log, 在向 employees 表中插入一条数据的时候,触发插入相

关的数据到 audit 中。

```
create trigger audit_log after
insert on employees_test
begin
insert into audit values(new.id,new.name);
end;
```

42、删除 emp_no 重复的记录，只保留最小的 id 对应的记录。

思路：第一步：按 emp_no 分组，选出每组最小的 id；

第二步：删除数据，id 不在上述 id 内。

```
delete from titles_test where id not in
(select min(id) from titles_test group by emp_no);
```

43、将所有 to_date 为 9999-01-01 的全部更新为 NULL,且 from_date 更新为 2001-01-01。

```
update titles_test set to_date = null,from_date = '2001-01-01';
```

44、将 id=5 以及 emp_no=10001 的行数据替换成 id=5 以及 emp_no=10005,其他数据保持不变，使用 replace 实现。

```
update titles_test set emp_no=replace(emp_no,10001,10005) where id=5;
```

45、将 titles_test 表名修改为 titles_2017。

```
alter table titles_test rename to titles_2017;
```

46、在 audit 表上创建外键约束，其 emp_no 对应 employees_test 表的主键 id。

```
ALTER TABLE audit ADD FOREIGN KEY (emp_no) REFERENCES employees_test (id);
```

47、create view emp_v as select * from employees where emp_no >10005;

如何获取 emp_v 和 employees 有相同的数据？

```
select e.* from employees e, emp_v ev where e.emp_no = ev.emp_no;
```

```
mysql> desc emp_bonus;
```

Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO		NULL	
received	datetime	NO		NULL	
btype	smallint(6)	NO		NULL	

48、将所有获取奖金的员工当前的薪水增加 10%。

```
update salaries set salary = salary*1.1 where emp_no in
```



```
(select s.emp_no from salaries s, emp_bonus e
where s.emp_no=e.emp_no and s.to_date='9999-01-01');
49、针对库中的所有表生成 selectcount(*)对应的 SQL 语句
```

关键点: 在 SQLite 系统表 sqlite_master 中可以获得所有表的索引, 其中字段 name 是所有表的名字, 而且对于自己创建的表而言, 字段 type 永远是 'table'

```
SELECT "select count(*) from " || name || ";" cnts
FROM sqlite_master WHERE type = 'table';
```

50、将 employees 表中的所有员工的 last_name 和 first_name 通过(,)连接起来。

```
select last_name||" "||first_name from employees;
```

51、查找字符串'10,A,B' 中逗号','出现的次数 cnt。

```
select (length('10,A,B') - length(replace('10,A,B',','))) cnt;
```

52、获取 Employees 中的 first_name, 查询按照 first_name 最后两个字母, 按照升序进行排列

```
select first_name from employees order by substr(first_name,-2);
```

53、按照 dept_no 进行汇总, 属于同一个部门的 emp_no 按照逗号进行连接, 结果给出 dept_no 以及连接出的结果 employees

本题要用到 SQLite 的聚合函数 group_concat(X,Y), 其中 X 是要连接的字段, Y 是连接时用的符号, 可省略, 默认为逗号。

```
select dept_no, group_concat(emp_no) employees from dept_emp group by dept_no;
```

54、查找排除当前最大、最小 salary 之后的员工的平均工资 avg_salary。

```
SELECT AVG(salary) avg_salary FROM salaries
WHERE to_date = '9999-01-01'
AND salary NOT IN (SELECT MAX(salary) FROM salaries WHERE to_date = '9999-01-01')
AND salary NOT IN (SELECT MIN(salary) FROM salaries WHERE to_date = '9999-01-01');
```

55、分页查询 employees 表, 每 5 行一页, 返回第 2 页的数据

```
select * from employees limit 5,5;
```

56、获取所有员工的 emp_no、部门编号 dept_no 以及对应的 bonus 类型 btype 和 received, 没有分配具体的员工不显示

```
select d.emp_no, d.dept_no, eb.btype, eb.received from
employees e inner join dept_emp d on e.emp_no = d.emp_no
left join emp_bonus eb on e.emp_no=eb.emp_no;
```

57、使用含有关键字 exists 查找未分配具体部门的员工的所有信息。

```
SELECT * FROM employees WHERE NOT EXISTS
```

```
(SELECT emp_no FROM dept_emp WHERE emp_no = employees.emp_no);
```

58、存在如下的视图：

```
create view emp_v as select * from employees where emp_no >10005;
```

获取 employees 中的行数据，且这些行也存在于 emp_v 中。注意不能使用 intersect 关键字。

```
select * from employees where emp_no>10005;
```

59、给出 emp_no、first_name、last_name、奖金类型 btype、对应的当前薪水情况 salary 以及奖金金额 bonus。bonus 类型 btype 为 1 其奖金为薪水 salary 的 10%，btype 为 2 其奖金为薪水的 20%，其他类型均为薪水的 30%。当前薪水表示 to_date='9999-01-01'

```
select e.emp_no, e.first_name, e.last_name, eb.btype, s.salary,  
(case eb.btype  
when 1 then s.salary*0.1  
when 2 then s.salary*0.2  
else s.salary*0.3 end) bonus  
from employees e inner join salaries s on e.emp_no = s.emp_no  
inner join emp_bonus eb on e.emp_no = eb.emp_no  
and s.to_date = '9999-01-01';
```

60、按照 salary 的累计和 running_total，其中 running_total 为前面员工的 salary 累计和，其他以此类推。具体结果如下 Demo 展示

```
select s1.emp_no, s1.salary,  
(select sum(s2.salary) from salaries s2  
where s2.emp_no <= s1.emp_no and s2.to_date='9999-01-01') running_total  
from salaries s1 where s1.to_date='9999-01-01';
```

61、对于 employees 表中，给出按 first_name 升序排列的奇数行的 first_name

```
select e1.first_name from employees e1  
where (select count(*) from employees e2  
where e1.first_name >= e2.first_name)%2=1;
```