AA

查找最晚入职的员工：

查找最晚入职员工的所有信息  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));

select \*

from employees

order by hire\_date desc

limit 0,1（分页）

AA

查找入职员工时间排名倒数第三的员工所有：

CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));

select \*

from employees

order by hire\_date desc

limit 2,1

查询各个部门当前（to-date=‘9999-01-01’）领导当前薪水详情以及各个部门对应的部门编号dept-no：

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  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));

CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));

答案：

select s.\*,d.dept\_no from salaries s,dept\_manager d

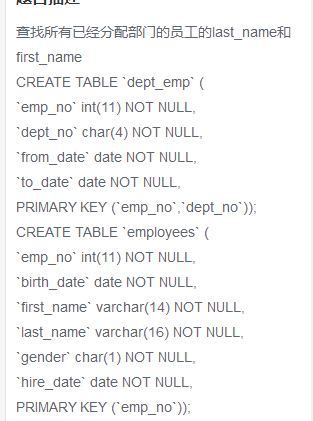
where d.to\_date = '9999-01-01'

and s.to\_date = '9999-01-01'

and s.emp\_no = d.emp\_no

查找所有已经分配部门的员工的last\_name和first——name

查找所有已经分配部门的员工的last\_name和first\_name  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));



select a.last\_name, a.first\_name, b.dept\_no

from employees a

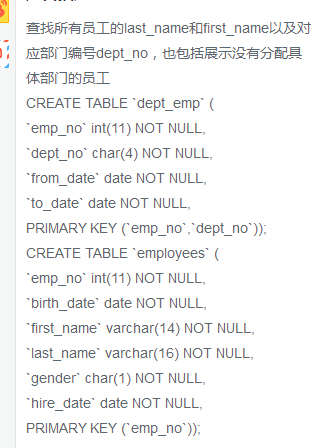
inner join dept\_emp b

where a.emp\_no=b.emp\_no;

查找所有员工的last\_name和first\_name以及对应部门编号dept\_no，也包括没有展示没有分配具体部门的员工:

## **题目描述**

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  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));



select a.last\_name, a.first\_name, b.dept\_no

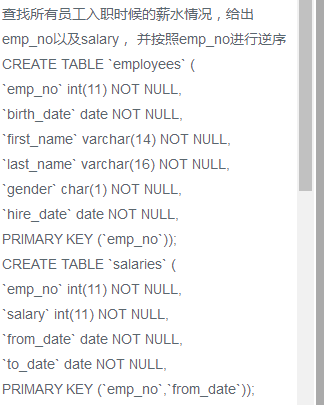
from employees a

left join dept\_emp b

on a.emp\_no=b.emp\_no;

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：

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  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select e.emp\_no,s.salary

from employees e,salaries s

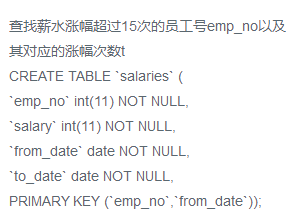
where e.emp\_no=s.emp\_no

and e.hire\_date=s.from\_date

order by e.emp\_no desc

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：

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  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select a.\*

from(

select emp\_no,count(\*) as t

from salaries

group by emp\_no) as a

where t>15

查找所有员工当前（to\_data=’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：

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  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select distinct s.salary

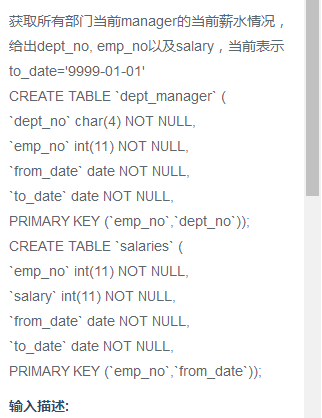
from salaries s

where s.to\_date='9999-01-01'

order by s.salary desc

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，当前拜师to\_date=’9999-01-01’

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'  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



注意:查找表的顺序

select dept\_manager.dept\_no,dept\_manager.emp\_no,salaries.salary

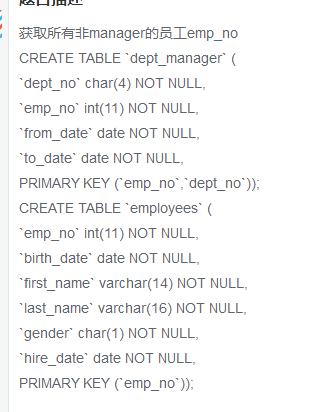
from salaries,dept\_manager,

where dept\_manager.to\_date='9999-01-01'

and salaries.to\_date='9999-01-01'

and dept\_manager.emp\_no=salaries.emp\_no

获取所有非manager的员工emp\_no：



select e.emp\_no

from employees e

left join dept\_manager d

on e.emp\_no=d.emp\_no

where d.emp\_no is null

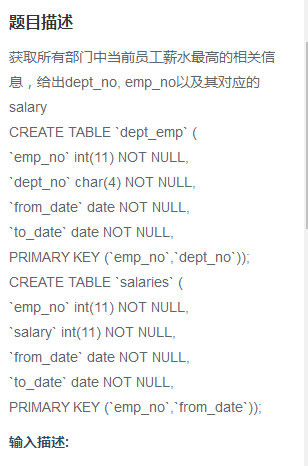
获取所有员工当前的manager，如果当前的manager是自己的话结果不显示，当前表示的to\_date=‘9999-01-01’，结果第一列给出当前员工的emp\_no,第二列给出其manager对应的manager\_no：

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。  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));



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以及对应的salary:

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  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select a.dept\_no,a.emp\_no,max(a.salary)

from

(

select d.dept\_no,d.emp\_no,s.salary

from dept\_emp d,salaries s

where d.emp\_no=s.emp\_no

and d.to\_date= s.to\_date

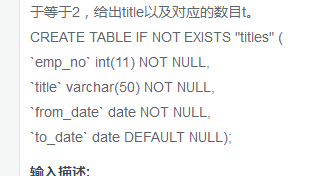
)

as a

group by a.dept\_no

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:

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。  
CREATE TABLE IF NOT EXISTS "titles" (  
`emp\_no` int(11) NOT NULL,  
`title` varchar(50) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date DEFAULT NULL);



select title,count(title)as t

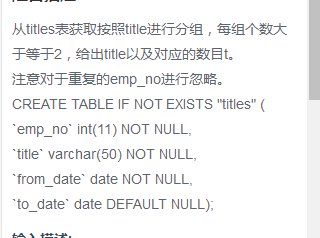
from titles

group by title

having t>1

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:

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。  
CREATE TABLE IF NOT EXISTS "titles" (  
`emp\_no` int(11) NOT NULL,  
`title` varchar(50) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date DEFAULT NULL);



select title,count(distinct emp\_no)as t

from titles

group by title

having t>1

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进行逆序排列:

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  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));



select \*

from employees

where emp\_no%2!=0

and last\_name not like 'Mary'

order by hire\_date desc

统计当前各个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:

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。  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));  
CREATE TABLE IF NOT EXISTS "titles" (  
`emp\_no` int(11) NOT NULL,  
`title` varchar(50) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date DEFAULT NULL);



select a.title,avg(a.salary)

from

(

select \*

from salaries s,titles t

where s.emp\_no=t.emp\_no

and s.to\_date='9999-01-01'

and t.to\_date='9999-01-01'

) as a

group by a.title

获取当前（to\_date=’9999-01-01’）薪水第二多的emp\_no以及对应的薪水salary:

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  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select emp\_no,salary

from salaries

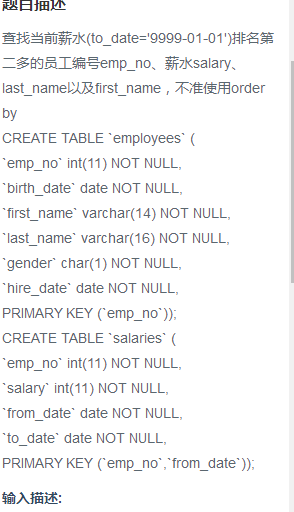
where to\_date='9999-01-01'

order by salary desc

limit 1,1

查找当前薪水（to\_date=’9999-0-1’）排名第二多的员工编号emp\_no,薪水salary，last\_name、first\_name .不准使用order by:

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  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select e.emp\_no,max(s.salary),e.last\_name,e.first\_name

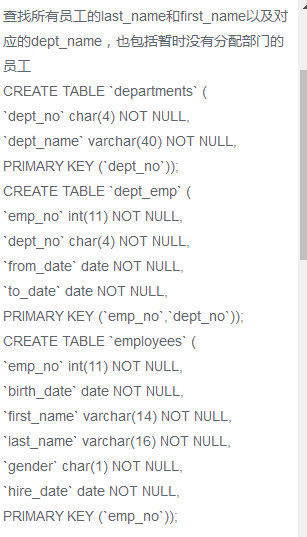
from employees e,salaries s

where e.emp\_no=s.emp\_no

and s.salary not in(select max(salary)

from salaries)

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分的:

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  
CREATE TABLE `departments` (  
`dept\_no` char(4) NOT NULL,  
`dept\_name` varchar(40) NOT NULL,  
PRIMARY KEY (`dept\_no`));  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));员工：

查找员工编号emp\_now为1000其自入职以来的salary涨幅值growth:

查找员工编号emp\_now为10001其自入职以来的薪水salary涨幅值growth  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select(

( select max(salary )

from salaries

where emp\_no='10001'

)

-

(

select min(salary)

from salaries

where emp\_no='10001'

)

)

as growth

查找所有员工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:

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\_noy以及其对应的薪水涨幅growth，并按照growth进行升序  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select a.emp\_no, a.csalary-b.hsalary as growth from

(select e.emp\_no, salary as csalary from salaries s, employees e

where e.emp\_no=s.emp\_no and s.to\_date ='9999-01-01') a,

(select e.emp\_no, salary as hsalary from salaries s, employees e

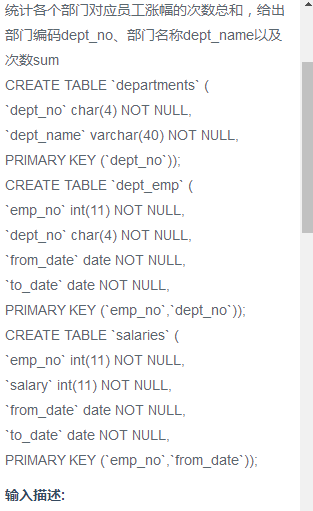
where e.emp\_no=s.emp\_no and s.from\_date =e.hire\_date) b

where a.emp\_no=b.emp\_no

order by growth asc

统计各个部门对应员工涨幅的次数综合，给出部门编码dept\_no、部门名称，dept\_name以及次数sum：

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  
CREATE TABLE `departments` (  
`dept\_no` char(4) NOT NULL,  
`dept\_name` varchar(40) NOT NULL,  
PRIMARY KEY (`dept\_no`));  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select d.dept\_no ,dept.dept\_name,count(salary)

from salaries s ,dept\_emp d,departments dept

where s.emp\_no = d.emp\_no

and d.dept\_no=dept.dept\_no

group by dept.dept\_no

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：

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  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select s1.emp\_no,s1.salary,count(distinct s2.salary) as rank

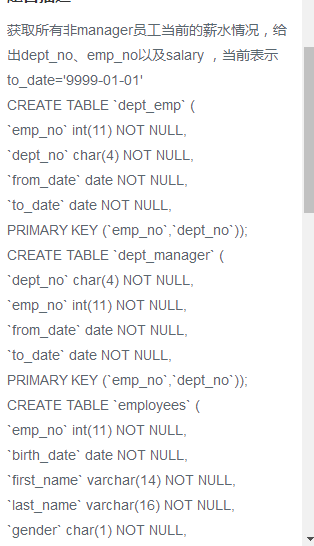
from salaries s1,salaries s2 where s1.salary<=s2.salary

and s1.to\_date='9999-01-01' and s2.to\_date='9999-01-01'

group by s1.emp\_no order by rank

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’:

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'  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select d.dept\_no,e.emp\_no,s.salary

from dept\_emp d,employees e,salaries s

where d.emp\_no = e.emp\_no and e.emp\_no = s.emp\_no

and e.emp\_no not in(select emp\_no from dept\_manager where to\_date='9999-01-01')

and s.to\_date='9999-01-01' and d.to\_date='9999-01-01'

获取员工当前薪水比其manager当前的薪水还高的相关信息，当前表示to\_date=‘9999-01-01’:

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  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));

输入描述:

select a.emp\_no,b.emp\_no as manager\_no,a.emp\_salary,b.manager\_salary

from

(select de.emp\_no,de.dept\_no,s1.salary as emp\_salary

from salaries s1

inner join dept\_emp de on de.emp\_no=s1.emp\_no

where de.to\_date='9999-01-01'

and s1.to\_date='9999-01-01' ) as a,

(select dm.emp\_no,dm.dept\_no,s2.salary as manager\_salary

from salaries s2

inner join dept\_manager dm

on dm.emp\_no=s2.emp\_no

where dm.to\_date='9999-01-01'

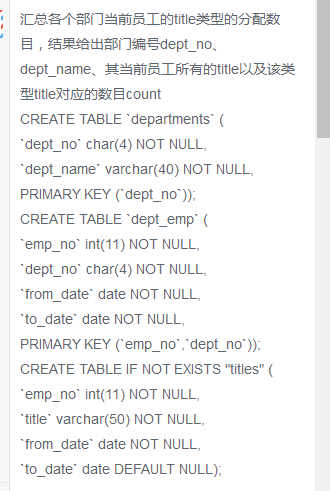
and s2.to\_date='9999-01-01') as b

where a.dept\_no=b.dept\_no

and a.emp\_salary>b.manager\_salary;

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,qid其当前员工所有的title以及该类型title 对应的count：

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  
CREATE TABLE `departments` (  
`dept\_no` char(4) NOT NULL,  
`dept\_name` varchar(40) NOT NULL,  
PRIMARY KEY (`dept\_no`));  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE IF NOT EXISTS "titles" (  
`emp\_no` int(11) NOT NULL,  
`title` varchar(50) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date DEFAULT NULL);



SELECT de.dept\_no, dp.dept\_name, t.title, COUNT(t.title) AS count

FROM titles AS t INNER JOIN dept\_emp AS de

ON t.emp\_no = de.emp\_no

AND de.to\_date = '9999-01-01'

AND t.to\_date = '9999-01-01'

INNER JOIN departments AS dp

ON de.dept\_no = dp.dept\_no

GROUP BY de.dept\_no, t.title

给出每个员工每年薪水涨幅超过5000的员工编号emp\_no、薪水 变更日期，from\_date以及薪水涨幅salary\_growth,并按照salary\_growth逆序排列：

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)

CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



select t1.emp\_no,t1.from\_date,(t1.salary-t2.salary) as salary\_growth

from

(select emp\_no,to\_date,salary,from\_date

from salaries as s1

) t1

inner join

(

select emp\_no,salary,to\_date,from\_date

from salaries as s2

) t2

on t1.emp\_no=t2.emp\_no

where (strftime('%Y',t1.to\_date)-strftime('%Y',t2.to\_date)=1 or

strftime('%Y',t1.from\_date)-strftime('%Y',t2.from\_date)=1) and

t1.salary-t2.salary >5000

order by salary\_growth desc

查找描述信息中包括robot的电影对应的分类名称及电影数目，而且还需要该分类对应的电影数量>=5：

film表

|  |  |
| --- | --- |
| 字段 | 说明 |
| film\_id | 电影id |
| title | 电影名称 |
| description | 电影描述信息 |

CREATE TABLE IF NOT EXISTS film (

film\_id smallint(5)  NOT NULL DEFAULT '0',

title varchar(255) NOT NULL,

description text,

PRIMARY KEY (film\_id));

category表

|  |  |
| --- | --- |
| 字段 | 说明 |
| category\_id | 电影分类id |
| name | 电影分类名称 |
| last\_update | 电影分类最后更新时间 |

CREATE TABLE category  (

category\_id  tinyint(3)  NOT NULL ,

name  varchar(25) NOT NULL, `last\_update` timestamp,

PRIMARY KEY ( category\_id ));

film\_category表

|  |  |
| --- | --- |
| 字段 | 说明 |
| film\_id | 电影id |
| category\_id | 电影分类id |
| last\_update | 电影id和分类id对应关系的最后更新时间 |

CREATE TABLE film\_category  (

film\_id  smallint(5)  NOT NULL,

category\_id  tinyint(3)  NOT NULL, `last\_update` timestamp);

查找描述信息中包括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>=5部

select c.name, count(f.film\_id) as t

from film f, category c, film\_category fc

where f.description like '%robot%'

and f.film\_id = fc.film\_id

and c.category\_id = fc.category\_id

group by c.category\_id

having t>=2

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 AS F

LEFT JOIN film\_category AS FC

on F.film\_id = FC.film\_id

WHERE FC.category\_id is null

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 title,description from film

where film\_id in

(select film\_id from film\_category

where category\_id in

(select category\_id from category

where name like 'action'))

获取select \* from employees对应的执行计划：

EXPLAIN SELECT \* FROM employees

在SQLite数据库中，可以用 "EXPLAIN" 关键字或 "EXPLAIN QUERY PLAN" 短语，用于描述表的细节，具体说明与用法可参考：

<http://www.runoob.com/sqlite/sqlite-explain.html>

<http://www.sqlite.org/lang_explain.html>

将employees表中的所有员工的last\_name和first\_name拼接起来作为name,中间以一个空格区分：

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  
CREATE TABLE `employees` ( `emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));



SELECT last\_name||' '||first\_name AS Name

FROM employees

说明：不同数据库连接字符串的方法不完全相同，MySQL、SQL Server、Oracle等数据库支持CONCAT方法，而本题所用的SQLite数据库只支持用连接符号"||"来连接字符串

创建一个actor表，包含以下信息：

创建一个actor表，包含如下列信息

|  |  |  |  |
| --- | --- | --- | --- |
| 列表 | 类型 | 是否为NULL | 含义 |
| actor\_id | smallint(5) | not null | 主键id |
| first\_name | varchar(45) | not null | 名字 |
| last\_name | varchar(45) | not null | 姓氏 |
| last\_update | timestamp | not null | 最后更新时间，默认是系统的当前时间 |



);

CREATE TABLE actor

(

actor\_id smallint(5) NOT NULL PRIMARY KEY,

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)

)

注意：主键设置为在后面添加primary key或者在后面标注

对于actor批量插入如下数据：

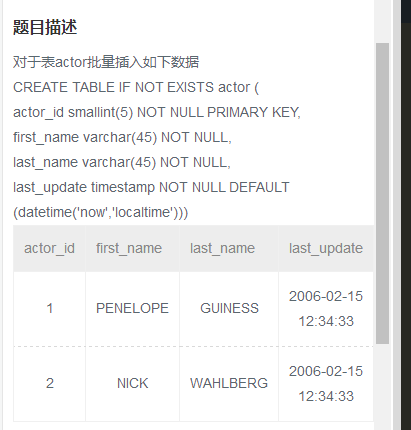
对于表actor批量插入如下数据  
CREATE TABLE IF NOT EXISTS actor (  
actor\_id smallint(5) NOT NULL PRIMARY KEY,  
first\_name varchar(45) NOT NULL,  
last\_name varchar(45) NOT NULL,  
last\_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))

|  |  |  |  |
| --- | --- | --- | --- |
| actor\_id | first\_name | last\_name | last\_update |
| 1 | PENELOPE | GUINESS | 2006-02-15 12:34:33 |
| 2 | NICK | WAHLBERG | 2006-02-15 12:34:33 |



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  
CREATE TABLE IF NOT EXISTS actor (  
actor\_id smallint(5) NOT NULL PRIMARY KEY,  
first\_name varchar(45) NOT NULL,  
last\_name varchar(45) NOT NULL,  
last\_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))

|  |  |  |  |
| --- | --- | --- | --- |
| actor\_id | first\_name | last\_name | last\_update |
| '3' | 'ED' | 'CHASE' | '2006-02-15 12:34:33' |



insert into actor values

(1, 'PENELOPE', 'GUINESS', '2006-02-15 12:34:33'),

(2, 'NICK', 'WAHLBERG', '2006-02-15 12:34:33')

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：

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  
CREATE TABLE IF NOT EXISTS actor (  
actor\_id smallint(5) NOT NULL PRIMARY KEY,  
first\_name varchar(45) NOT NULL,  
last\_name varchar(45) NOT NULL,  
last\_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))

|  |  |  |  |
| --- | --- | --- | --- |
| actor\_id | first\_name | last\_name | last\_update |
| '3' | 'ED' | 'CHASE' | '2006-02-15 12:34:33' |



insert or ignore into actor

values(3,'ED','CHASE','2006-02-15 12:34:33');

在 SQLite 中，用 INSERT OR IGNORE 来插入记录，或忽略插入与表内UNIQUE字段都相同的记录

|  |  |
| --- | --- |
| 1 | INSERT OR IGNORE INTO actor VALUES (3, 'ED', 'CHASE', '2006-02-15 12:34:33') |

用 INSERT OR REPLACE 来插入记录，或更新替代与表内UNIQUE字段都相同的记录

|  |  |
| --- | --- |
| 1 | INSERT OR REPLACE INTO actor VALUES (3, 'ED', 'CHASE', '2006-02-15 12:34:33') |

对于如下表actor，其对应的数据为：对于如下表actor，其对应的数据为:

|  |  |  |  |
| --- | --- | --- | --- |
| actor\_id | first\_name | last\_name | last\_update |
| 1 | PENELOPE | GUINESS | 2006-02-15 12:34:33 |
| 2 | NICK | WAHLBERG | 2006-02-15 12:34:33 |

创建一个actor\_name表，将actor表中的所有first\_name以及last\_name导入改表。 actor\_name表结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 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,

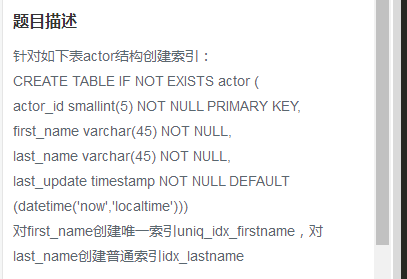
last\_name varchar(45) not null

);

insert into actor\_name select first\_name,last\_name from actor

针对如下表actor结构创建索引：

针对如下表actor结构创建索引：  
CREATE TABLE IF NOT EXISTS actor (  
actor\_id smallint(5) NOT NULL PRIMARY KEY,  
first\_name varchar(45) NOT NULL,  
last\_name varchar(45) NOT NULL,  
last\_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))  
对first\_name创建唯一索引uniq\_idx\_firstname，对last\_name创建普通索引idx\_lastname

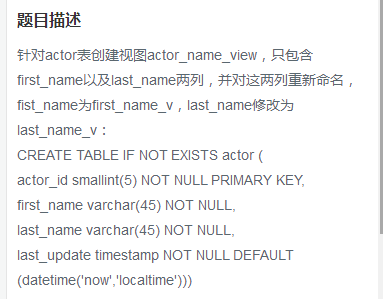


create unique index uniq\_idx\_firstname on actor(first\_name);

create index idx\_lastname on actor(last\_name);

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:

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，fist\_name为first\_name\_v，last\_name修改为last\_name\_v：  
CREATE TABLE IF NOT EXISTS actor (  
actor\_id smallint(5) NOT NULL PRIMARY KEY,  
first\_name varchar(45) NOT NULL,  
last\_name varchar(45) NOT NULL,  
last\_update timestamp NOT NULL DEFAULT (datetime('now','localtime')))



create view actor\_name\_view as

select first\_name as fist\_name\_v, last\_name as last\_name\_v

from actor

针对salaries表emp\_no字段创建索引idx\_emp\_no，查询emp\_no为1005，使用强制索引：

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。  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));  
create index idx\_emp\_no on salaries(emp\_no);



select \* from salaries indexed by idx\_emp\_no

where emp\_no=10005

SQLite中，使用 INDEXED BY 语句进行强制索引查询，可参考：

[http://www.runoob.com/sqlite/sqlite-indexed-by.html](http://www.runoob.com/sqlite/sqlite-indexed-by.html" \t "https://www.nowcoder.com/profile/1427705/_blank)

|  |  |
| --- | --- |
| 1 | SELECT \* FROM salaries INDEXED BY idx\_emp\_no WHERE emp\_no = 10005 |

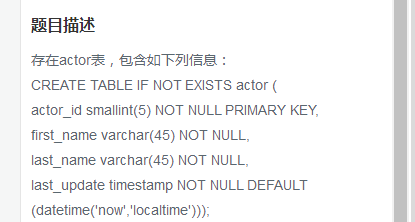
MySQL中，使用 FORCE INDEX 语句进行强制索引查询，可参考：

<http://www.jb51.net/article/49807.htm>

|  |  |
| --- | --- |
| 1 | SELECT \* FROM salaries FORCE INDEX idx\_emp\_no WHERE emp\_no = 10005 |

存在actor表，包含以下信息，现在在last\_update后面增加一列名字为create\_date，类型为datetime，not null,默认值为‘0000 00：00:00’:

存在actor表，包含如下列信息：  
CREATE TABLE IF NOT EXISTS actor (  
actor\_id smallint(5) NOT NULL PRIMARY KEY,  
first\_name varchar(45) NOT NULL,  
last\_name varchar(45) NOT NULL,  
last\_update timestamp NOT NULL DEFAULT (datetime('now','localtime')));  
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'

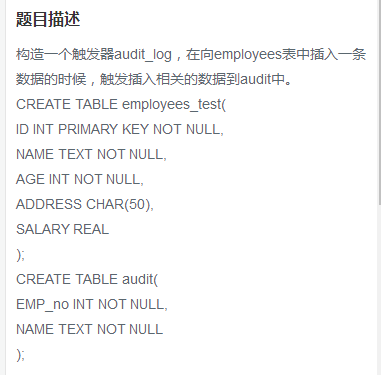
：

alter table actor

add column create\_date datetime NOT NULL DEFAULT '0000-00-00 00:00:00'

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：

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 TABLE employees\_test(  
ID INT PRIMARY KEY NOT NULL,  
NAME TEXT NOT NULL,  
AGE INT NOT NULL,  
ADDRESS CHAR(50),  
SALARY REAL  
);  
CREATE TABLE audit(  
EMP\_no INT NOT NULL,  
NAME TEXT NOT NULL  
);



create trigger audit\_log after insert on employees\_test

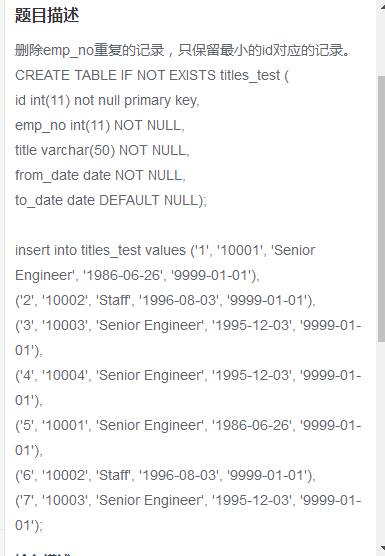
begin

insert into audit values(new.id,new.name);

end;

删除emp\_no重复的记录，只保留最小的id对应的记录：

删除emp\_no重复的记录，只保留最小的id对应的记录。  
CREATE TABLE IF NOT EXISTS titles\_test (  
id int(11) not null primary key,  
emp\_no int(11) NOT NULL,  
title varchar(50) NOT NULL,  
from\_date date NOT NULL,  
to\_date date DEFAULT NULL);  
  
insert into titles\_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');



DELETE FROM titles\_test

WHERE id NOT IN (

SELECT

min(id) AS id

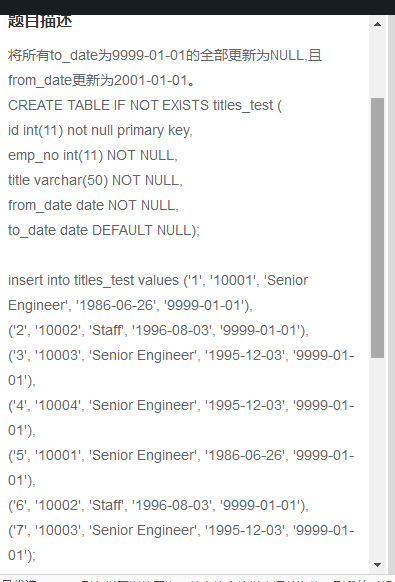
FROM titles\_test

GROUP BY emp\_no);

说明:本题思路如下：先用 GROUP BY 和 MIN() 选出每个 emp\_no 分组中最小的 id，然后用 DELETE FROM ... WHERE ... NOT IN ... 语句删除 “非每个分组最小id对应的所有记录”

将所有的to\_date为9999-01-01的全部更新为null，且from\_date更新为2001-01-01：

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。  
CREATE TABLE IF NOT EXISTS titles\_test (  
id int(11) not null primary key,  
emp\_no int(11) NOT NULL,  
title varchar(50) NOT NULL,  
from\_date date NOT NULL,  
to\_date date DEFAULT NULL);  
  
insert into titles\_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');



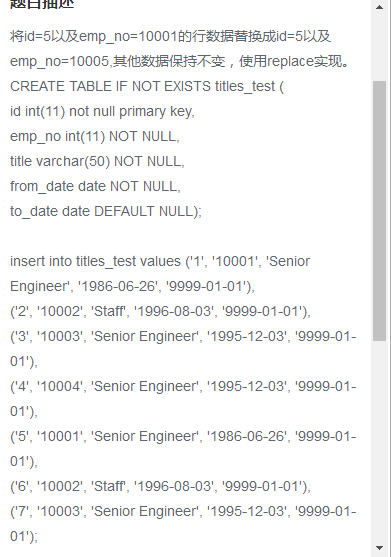
update titles\_test

set to\_date=null,from\_date='2001-01-01'

where to\_date='9999-01-01'

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：

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。  
CREATE TABLE IF NOT EXISTS titles\_test (  
id int(11) not null primary key,  
emp\_no int(11) NOT NULL,  
title varchar(50) NOT NULL,  
from\_date date NOT NULL,  
to\_date date DEFAULT NULL);  
  
insert into titles\_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');



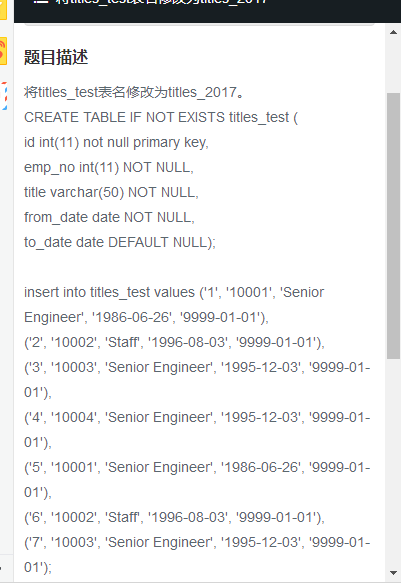
replace into titles\_test

values(5, 10005, 'Senior Engineer', '1986-06-26', '9999-01-01');



将titles\_test表名修改为titles\_2017：

将titles\_test表名修改为titles\_2017。  
CREATE TABLE IF NOT EXISTS titles\_test (  
id int(11) not null primary key,  
emp\_no int(11) NOT NULL,  
title varchar(50) NOT NULL,  
from\_date date NOT NULL,  
to\_date date DEFAULT NULL);  
  
insert into titles\_test values ('1', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('2', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('3', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('4', '10004', 'Senior Engineer', '1995-12-03', '9999-01-01'),  
('5', '10001', 'Senior Engineer', '1986-06-26', '9999-01-01'),  
('6', '10002', 'Staff', '1996-08-03', '9999-01-01'),  
('7', '10003', 'Senior Engineer', '1995-12-03', '9999-01-01');



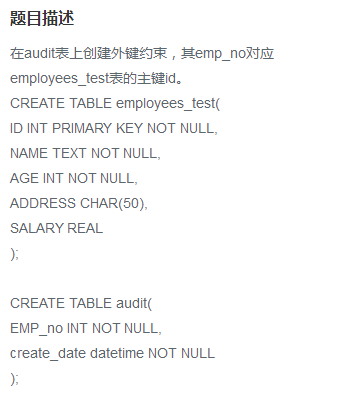
这是sqllite的写法：

alter table titles\_test

rename to titles\_2017

在audit表上创建外键约束，其emp\_no对应employees\_test表的主键：

在audit表上创建外键约束，其emp\_no对应employees\_test表的主键id。  
CREATE TABLE employees\_test(  
ID INT PRIMARY KEY NOT NULL,  
NAME TEXT NOT NULL,  
AGE INT NOT NULL,  
ADDRESS CHAR(50),  
SALARY REAL  
);  
  
CREATE TABLE audit(  
EMP\_no INT NOT NULL,  
create\_date datetime NOT NULL  
);



drop table audit;

create table audit(

EMP\_no int not null,

create\_date datetime not null,

foreign key(EMP\_no) references employees\_test(ID));

由于SQLite中不能通过 ALTER TABLE ... ADD FOREIGN KEY ... REFERENCES ... 语句来对已创建好的字段创建外键，因此只能先删除表，再重新建表的过程中创建外键。

存在如下的视图：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？



说明：获取相同数据，只需要两者的查询条件保持一致即可：

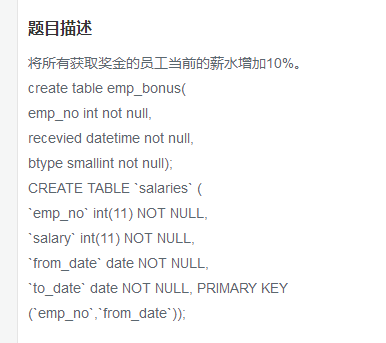
SELECT em.\*

FROM employees AS em, emp\_v AS ev

WHERE em.emp\_no = ev.emp\_no

将所有获取奖金员工的当前薪水增加10%：

将所有获取奖金的员工当前的薪水增加10%。  
create table emp\_bonus(  
emp\_no int not null,  
recevied datetime not null,  
btype smallint not null);  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL, PRIMARY KEY (`emp\_no`,`from\_date`));



update salaries

set salary =salary\*1.1

where emp\_no in

(select emp\_no

from emp\_bonus

)

针对库中的所有表生成 select count(\*)对应大的SQL语句：

针对库中的所有表生成select count(\*)对应的SQL语句  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
create table emp\_bonus(  
emp\_no int not null,  
recevied datetime not null,  
btype smallint not null);  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));



SELECT "select count(\*) from " || name || ";" AS cnts

FROM sqlite\_master WHERE type = 'table'

本题主要有以下两个关键点：

1、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'，详情可参考：

<http://blog.csdn.net/xingfeng0501/article/details/7804378>

2、在 SQLite 中用 “||” 符号连接字符串

|  |  |
| --- | --- |
| 1  2 | SELECT "select count(\*) from " || name || ";" AS cnts  FROM sqlite\_master WHERE type = 'table' |

将employees表中的所有员工的last\_name和first\_name通过（‘）连接起来：

将employees表中的所有员工的last\_name和first\_name通过(')连接起来。  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));



SELECT last\_name || "'" || first\_name FROM employees

在SQLite数据库中，只支持连接符号“||”来连接，不支持，函数连接。



查找字符串‘10，A，B’中逗号的出现的次数cnt:

select (length('10,A,B')-length(replace('10,A,B',',','')))

as cnt

Sql:

select len(text) - len(replace(text,',','')) from tb

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：

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  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));



SELECT first\_name

FROM employees

ORDER BY substr(first\_name,length(first\_name)-1)

SELECT first\_name

FROM employees

ORDER BY substr(first\_name,-2)

解析：本题考查 substr(X,Y,Z) 或 substr(X,Y) 函数的使用。其中X是要截取的字符串。Y是字符串的起始位置（注意第一个字符的位置为1，而不为0），取值范围是±(1~length(X))，当Y等于length(X)时，则截取最后一个字符；当Y等于负整数-n时，则从倒数第n个字符处截取。Z是要截取字符串的长度，取值范围是正整数，若Z省略，则从Y处一直截取到字符串末尾；若Z大于剩下的字符串长度，也是截取到字符串末尾为止。

按照dept\_no进行汇总，属于同一个部门的emp\_no按照逗号连接，结果给出dept\_no以及连接出的结果employees:

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  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));



CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));

SELECT dept\_no, group\_concat(emp\_no) AS employees

FROM dept\_emp

GROUP BY dept\_no

解析：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。此函数必须与 GROUP BY 配合使用。此题以 dept\_no 作为分组，将每个分组中不同的emp\_no用逗号连接起来（即可省略Y）。可参考：

[http://www.sqlite.org/lang\_aggfunc.html#groupconcat](http://www.sqlite.org/lang_aggfunc.html%23groupconcat)

<http://blog.csdn.net/langzxz/article/details/16807859>

group\_concat()函数返回X的非null值的连接后的字符串。如果给出了参数Y，将会在每个X之间用Y作为分隔符。如果省略了Y，“，”将作为默认的分隔符。每个元素连接的顺序是随机的。

查找排除当前最大，最小salary之后的员工的平均工资avg\_salary：

CREATE TABLE `salaries` ( `emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));

select avg(salary) as avg\_salary

from salaries

where to\_date='9999-01-01'

and salary not in (

select max(salary)

from salaries

)

and salary not in(

select min(salary)

from salaries

)

分页查询employees表，每5行一页，返回第二页的数据

CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));

select \*

from employees

limit 5,5

根据题意，每行5页，返回第2页的数据，即返回第6~10条记录，以下有两种方法可以解决：

方法一：利用 LIMIT 和 OFFSET 关键字。LIMIT 后的数字代表返回几条记录，OFFSET 后的数字代表从第几条记录开始返回（第一条记录序号为0），也可理解为跳过多少条记录后开始返回。

2.几种典型的分页sql，下面例子是每页50条，198\*50=9900，取第199页数据。

[IMG_256](http://www.cnblogs.com/zcttxs/archive/2012/04/01/javascript:void(0);)

--写法1，not in/top  
select top 50 \* from pagetest   
where id not in (select top 9900 id from pagetest order by id)  
order by id  
  
  
--写法2，not exists  
select top 50 \* from pagetest   
where not exists   
(select 1 from (select top 9900 id from pagetest order by id)a where a.id=pagetest.id)  
order by id  
  
--写法3，max/top  
select top 50 \* from pagetest  
where id>(select max(id) from (select top 9900 id from pagetest order by id)a)  
order by id  
  
--写法4，row\_number()  
select top 50 \* from   
(select row\_number()over(order by id)rownumber,\* from pagetest)a  
where rownumber>9900  
  
select \* from   
(select row\_number()over(order by id)rownumber,\* from pagetest)a  
where rownumber>9900 and rownumber<9951  
  
select \* from   
(select row\_number()over(order by id)rownumber,\* from pagetest)a  
where rownumber between 9901 and 9950  
  
--写法5，在csdn上一帖子看到的，row\_number() 变体，不基于已有字段产生记录序号，先按条件筛选以及排好序，再在结果集上给一常量列用于产生记录序号  
select \*  
from (  
 select row\_number()over(order by tempColumn)rownumber,\*  
 from (select top 9950 tempColumn=0,\* from pagetest where 1=1 order by id)a  
)b  
where rownumber>9900

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recevied，没有分配具体的员工不显示：

CREATE TABLE `dept\_emp` ( `emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `dept\_manager` (  
`dept\_no` char(4) NOT NULL,  
`emp\_no` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));

select de.emp\_no,de.dept\_no,eb.btype,eb.recevied

from dept\_emp de

left join emp\_bonus eb

on eb.emp\_no =de.emp\_no

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。  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));

select \*

from employees

where not exists

(select emp\_no

from dept\_emp

where emp\_no=employees.emp\_no)

存在如下的视图：  
create view emp\_v as select \* from employees where emp\_no >10005;  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
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 emp\_v

解析：

根据题意，不能使用 INTERSECT 关键字，但由于视图 emp\_v 的记录是从 employees 中导出的，因此要判断两者中相等的数据，只需要判断emp\_no相等即可。

方法一：用 WHERE 选取二者 emp\_no 相等的记录

|  |  |
| --- | --- |
| 1 | SELECT em.\* FROM employees AS em, emp\_v AS ev WHERE em.emp\_no = ev.emp\_no |

方法二：由于emp\_v的全部记录均由 employees 导出，因此可以投机取巧，直接输出 emp\_v 所有记录

|  |  |
| --- | --- |
| 1 | SELECT \* FROM emp\_v |

获取有奖金的员工相关信息。  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));  
CREATE TABLE `dept\_emp` (  
`emp\_no` int(11) NOT NULL,  
`dept\_no` char(4) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`dept\_no`));  
create table emp\_bonus(  
emp\_no int not null,  
recevied datetime not null,  
btype smallint not null);  
CREATE TABLE `salaries` (  
`emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL, PRIMARY KEY (`emp\_no`,`from\_date`));  
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, b.btype, s.salary,

(CASE b.btype

WHEN 1 THEN s.salary \* 0.1

WHEN 2 THEN s.salary \* 0.2

ELSE s.salary \* 0.3 END) AS bonus

FROM employees AS e

INNER JOIN emp\_bonus AS b

ON e.emp\_no = b.emp\_no

INNER JOIN salaries AS s

ON e.emp\_no = s.emp\_no

AND s.to\_date = '9999-01-01'

解析：本题主要考查 SQLite 中 CASE 表达式的用法。即当 btype = 1 时，得到 salary \* 0.1；当 btype = 2 时，得到 salary \* 0.2；其他情况得到 salary \* 0.3。详细用法请参考：

按照salary的累计和running\_total，其中running\_total为前两个员工的salary累计和，其他以此类推。 具体结果如下Demo展示。。  
CREATE TABLE `salaries` ( `emp\_no` int(11) NOT NULL,  
`salary` int(11) NOT NULL,  
`from\_date` date NOT NULL,  
`to\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`,`from\_date`));

SELECT s1.emp\_no, s1.salary,

(SELECT SUM(s2.salary)

FROM salaries AS s2

WHERE s2.emp\_no <= s1.emp\_no

AND s2.to\_date = '9999-01-01') AS running\_total

FROM salaries AS s1

WHERE s1.to\_date = '9999-01-01'

解析：本题的思路为复用 salaries 表进行子查询，最后输出求和结果。

1、输出的第三个字段，是由一个 SELECT 子查询构成。将子查询内复用的 salaries 表记为 s2，主查询的 salaries 表记为 s1，当主查询的 s1.emp\_no 确定时，对子查询中不大于 s1.emp\_no 的 s2.emp\_no 所对应的薪水求和

2、注意是对员工当前的薪水求和，所以在主查询和子查询内都要加限定条件 to\_date = '9999-01-01'

对于employees表中，给出奇数行的first\_name  
CREATE TABLE `employees` (  
`emp\_no` int(11) NOT NULL,  
`birth\_date` date NOT NULL,  
`first\_name` varchar(14) NOT NULL,  
`last\_name` varchar(16) NOT NULL,  
`gender` char(1) NOT NULL,  
`hire\_date` date NOT NULL,  
PRIMARY KEY (`emp\_no`));

SELECT e1.first\_name

FROM

(SELECT e2.first\_name,

(SELECT COUNT(\*)

FROM employees AS e3

WHERE e3.first\_name <= e2.first\_name)

AS rowid

FROM employees AS e2) AS e1

WHERE e1.rowid % 2 = 1

解析：本题的思路为复用 salaries 表进行子查询，最后输出求和结果。

1、输出的第三个字段，是由一个 SELECT 子查询构成。将子查询内复用的 salaries 表记为 s2，主查询的 salaries 表记为 s1，当主查询的 s1.emp\_no 确定时，对子查询中不大于 s1.emp\_no 的 s2.emp\_no 所对应的薪水求和

2、注意是对员工当前的薪水求和，所以在主查询和子查询内都要加限定条件 to\_date = '9999-01-01'