# SQL基础(MYSQ的高级查询DML和DQL)

注意事项(这都是在图形化界面写的，终端要看好符号，例如’’;等)

1. MYSQL和ORACLE 相差不大，这两个的区别已备注.
2. 最好不要用中文添加表名或者字段(学习的时候没注意，最后不好改了)(表被’’修饰是MQSQL自带)
3. 原生SQL是数据库的基础，不过都相差不大,下面是SQL基础.

查询语句select:

知识点：内连接(Inner Join)，外连接(Outer Join)，自连接(Self- Join),Group By和having查询，子查询，相关子查询，获取最前面的指定行数记录，分页查询，查询结果操作等;

1. 连接查询(内连接 Inner Join)将多个表联合查询出多个表中的数据。

SELECT 字段名|表达式|分组函数 等等

FROM 表名，表名等等

WHERE 查询条件

GROUP BY 分组字段列表

HAVING 分组查询过滤条件

ORDER BY 排序字段列表

例如:

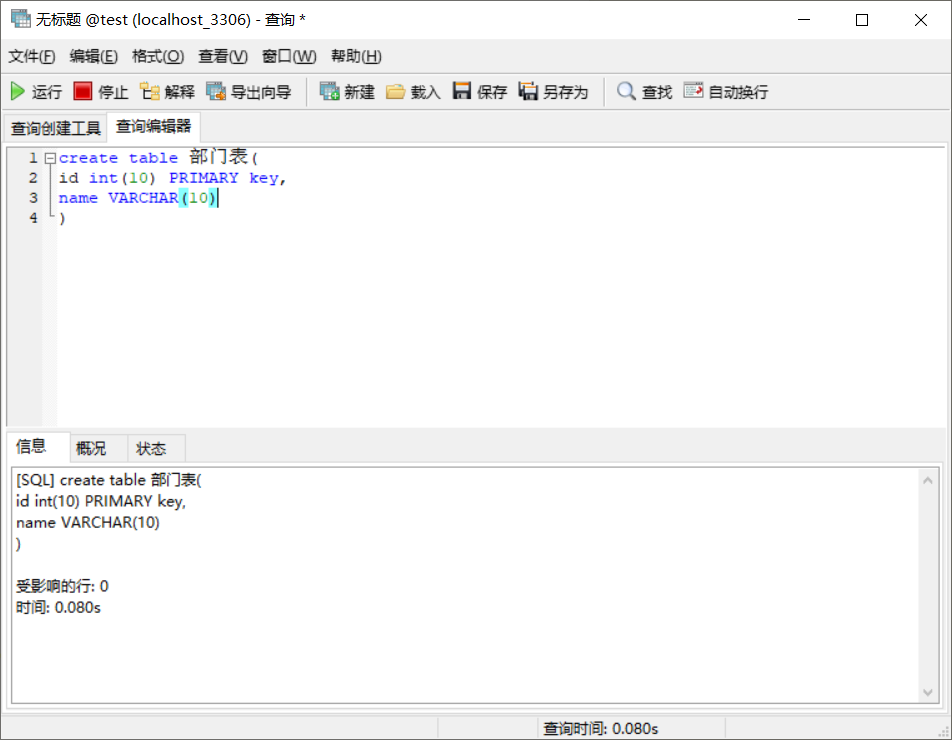
创建一个部门表

create table 部门表(

id int(10) PRIMARY key,

name VARCHAR(10)

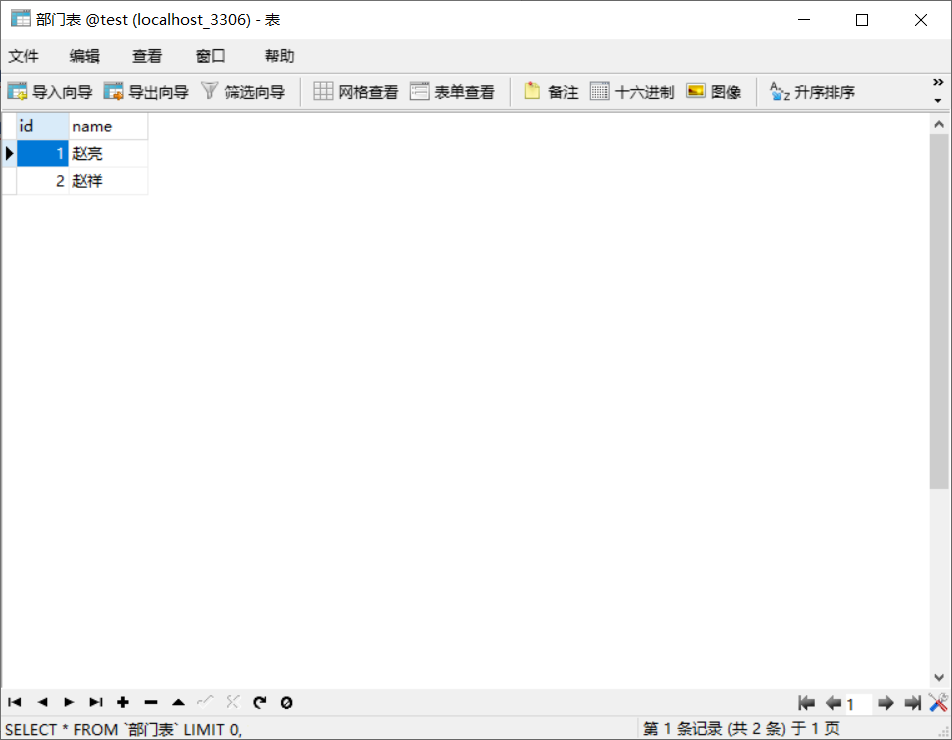
)



向部门插入数据

Insert into 部门表(id,name) values(1,’赵亮’);

Insert into 部门表(id,name) values(2,’赵祥’);



创建一个员工表

create table 员工表

(

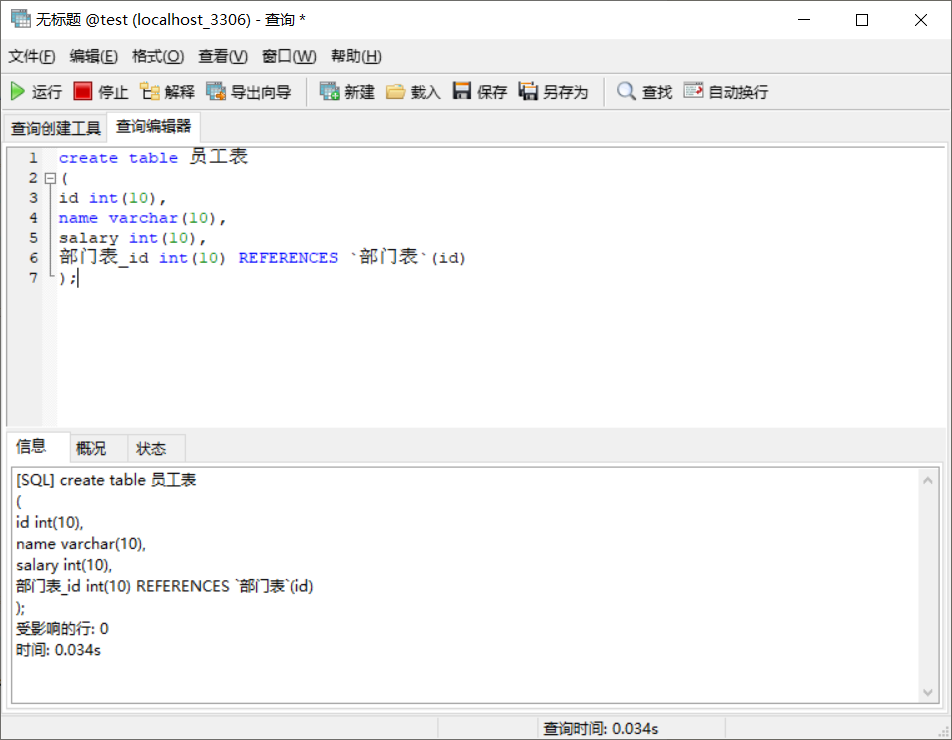
id int(10),

name varchar(10),

salary int(10),

部门表\_id int(10) REFERENCES `部门表`(id)

);



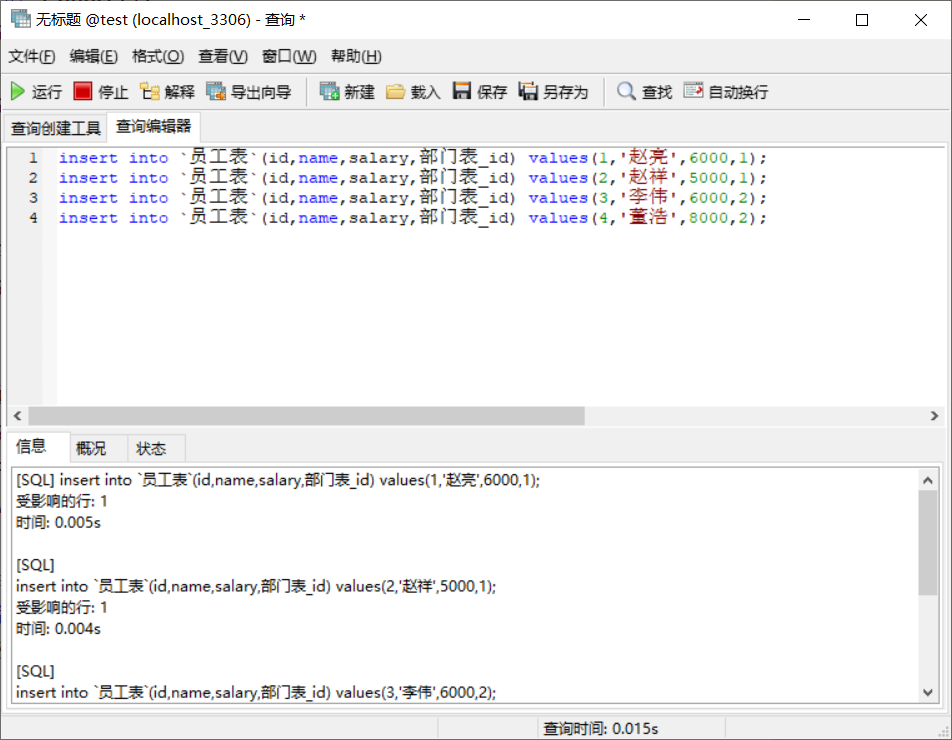
向员工表中插入数据

insert into `员工表`(id,name,salary,部门表\_id) values(1,'赵亮',6000,1);

insert into `员工表`(id,name,salary,部门表\_id) values(2,'赵祥',5000,1);

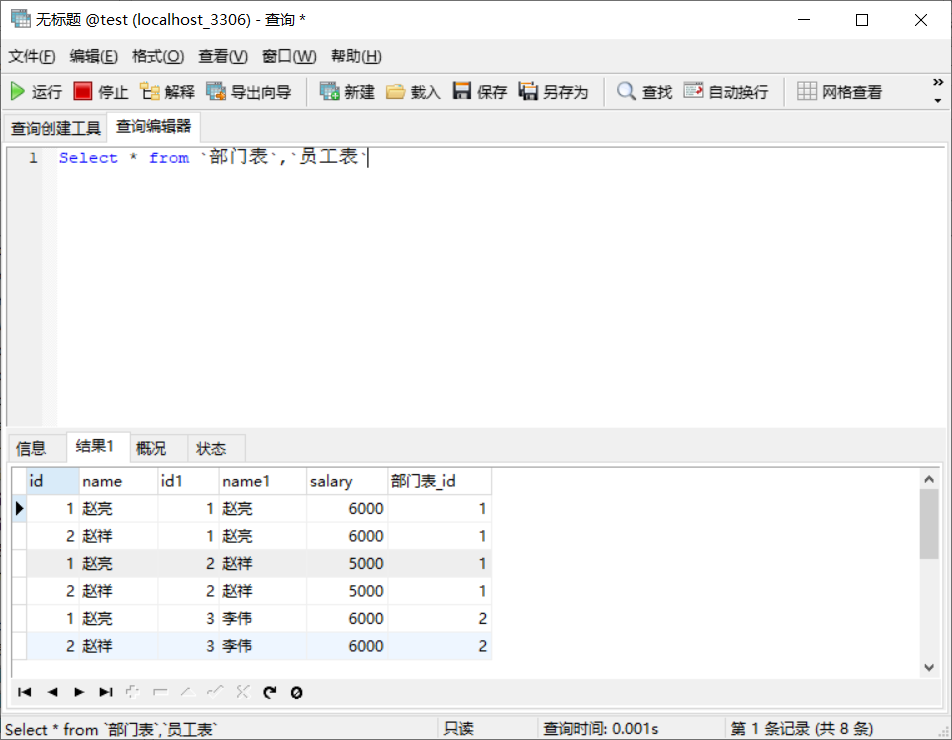
insert into `员工表`(id,name,salary,部门表\_id) values(3,'李伟',6000,2);

insert into `员工表`(id,name,salary,部门表\_id) values(4,'董浩',8000,2);



1. 笛卡尔积交叉连接，结果为两个表的乘积

SELECT \* FROM 部门表,员工表



1. 等值连接通过相同字段名为条件建立起来的连接

SELECT 字段|表达式

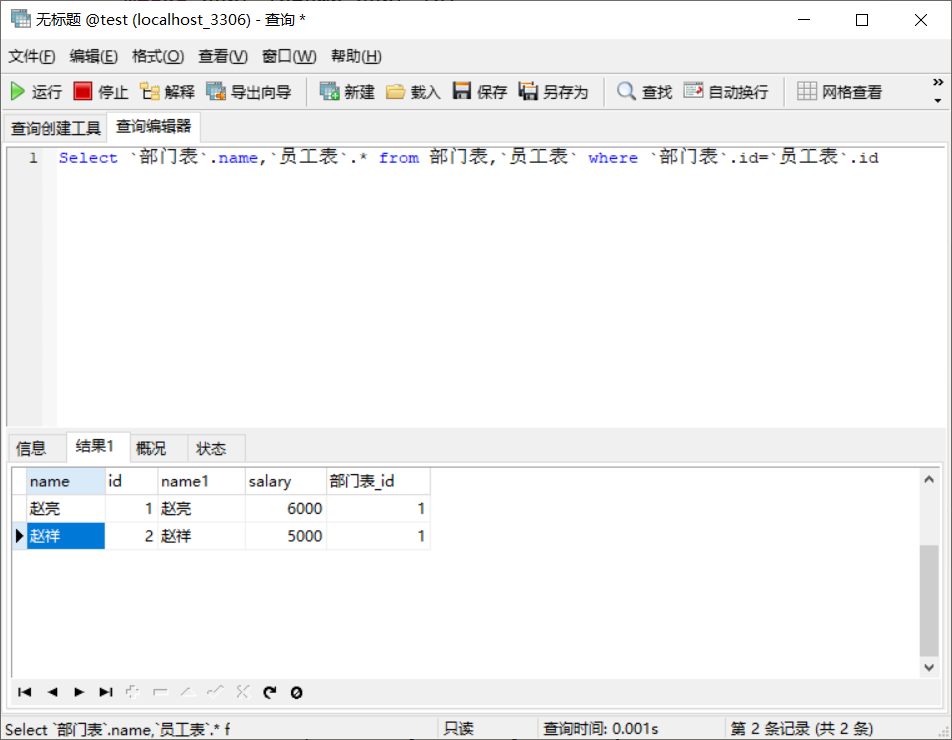
FROM 表名1，表名2

WHERE 表名1.列名=表名2.列名;

当连接的表中有相同字段时，字段一般都加上“表名”前缀，例如:

显示所有员工信息，和其所在的部门名称

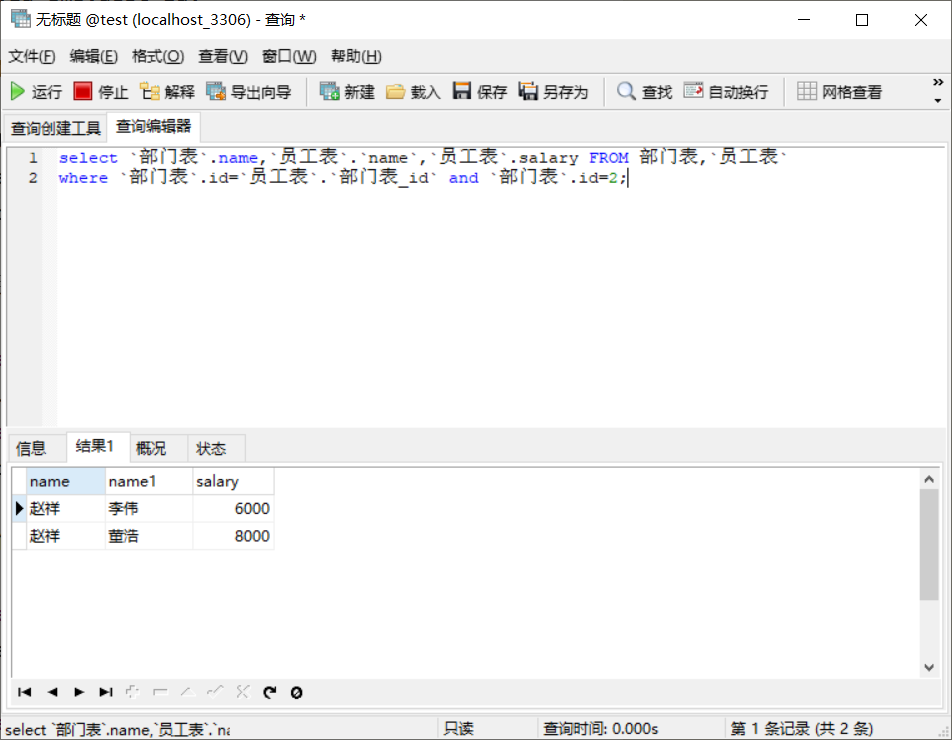
Select `部门表`.name,`员工表`.\* from 部门表,`员工表` where `部门表`.id=`员工表`.id



显示 赵亮 的所有员工，工资和其所在的部门名称

select `部门表`.name,`员工表`.`name`,`员工表`.salary FROM 部门表,`员工表`

where `部门表`.id=`员工表`.`部门表\_id` and `部门表`.id=2;



1. 连接中 表 使用 别名(自己起的名字)

显示 赵亮 的所有员工信息，和其所在的部门名称

select a.name,b.\* from `部门表` a JOIN `员工表` b ON a.id=b.`部门表\_id` and a.id=2

JOIN ON等价于WHERE，WHERE也可以SQL连接

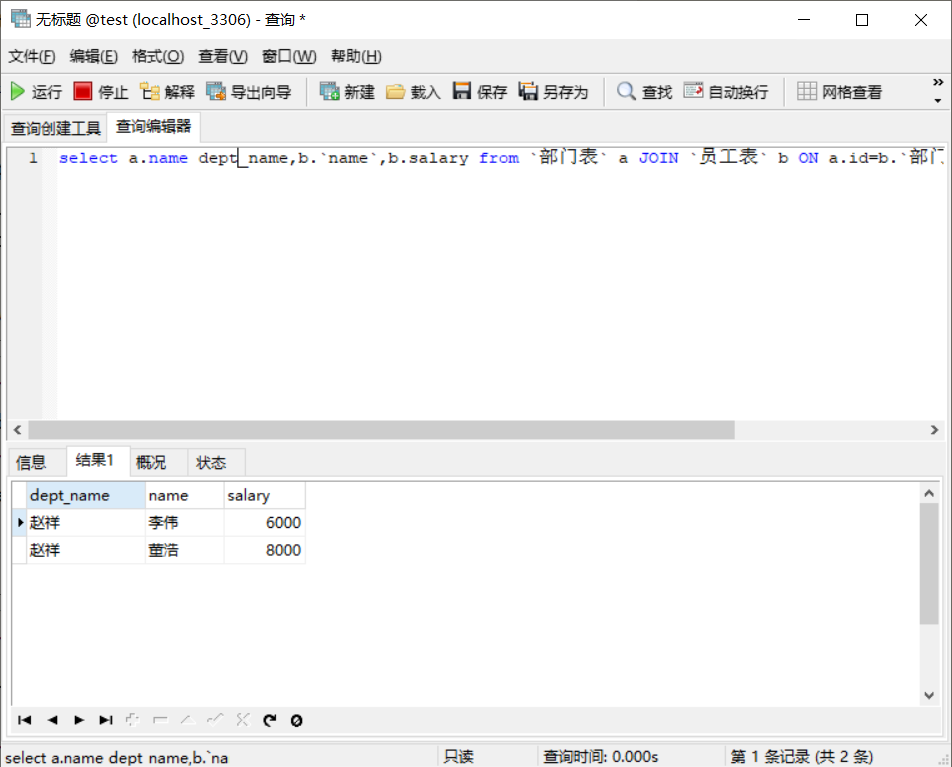


1. 连接中 字段 使用别名

显示 赵亮 的所有员工信息，和其所在的部门名称

select a.name dept\_name,b.`name`,b.salary

from `部门表` a JOIN `员工表` b ON a.id=b.`部门表\_id` and a.id=2



1. 多表等值连接

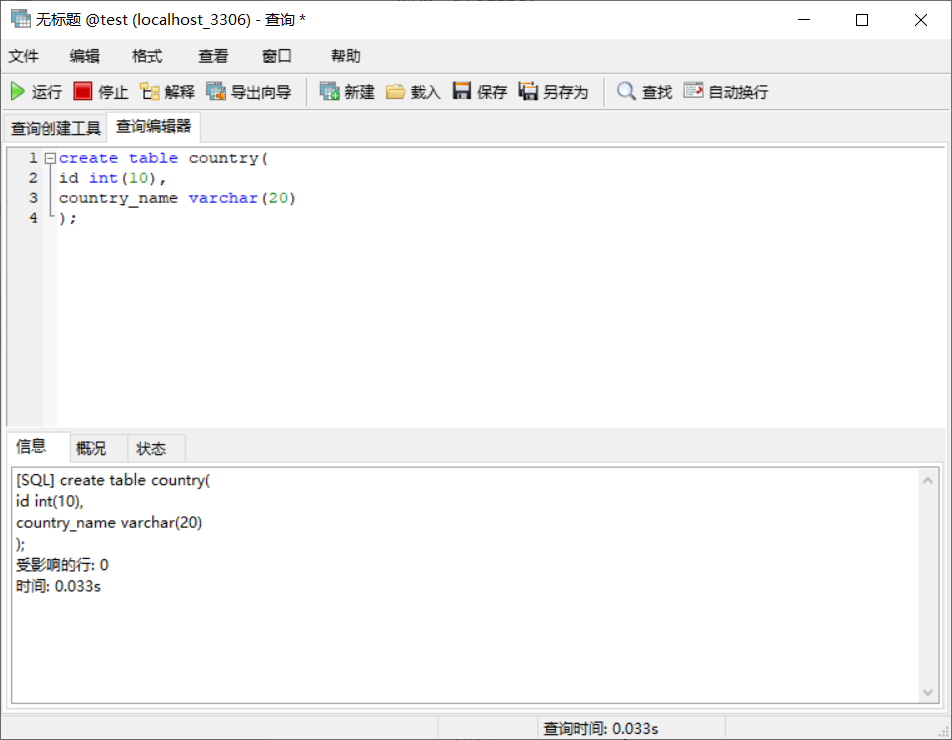
创建一个国家表 country并插入数据

create table country(

id int(10),

country\_name varchar(20)

);



插入数据

insert into country(id,country\_name) values(1,'中国');

insert into country(id,country\_name) values(2,'美国');



创建一个省份表province，并且插入数据

Create table province(

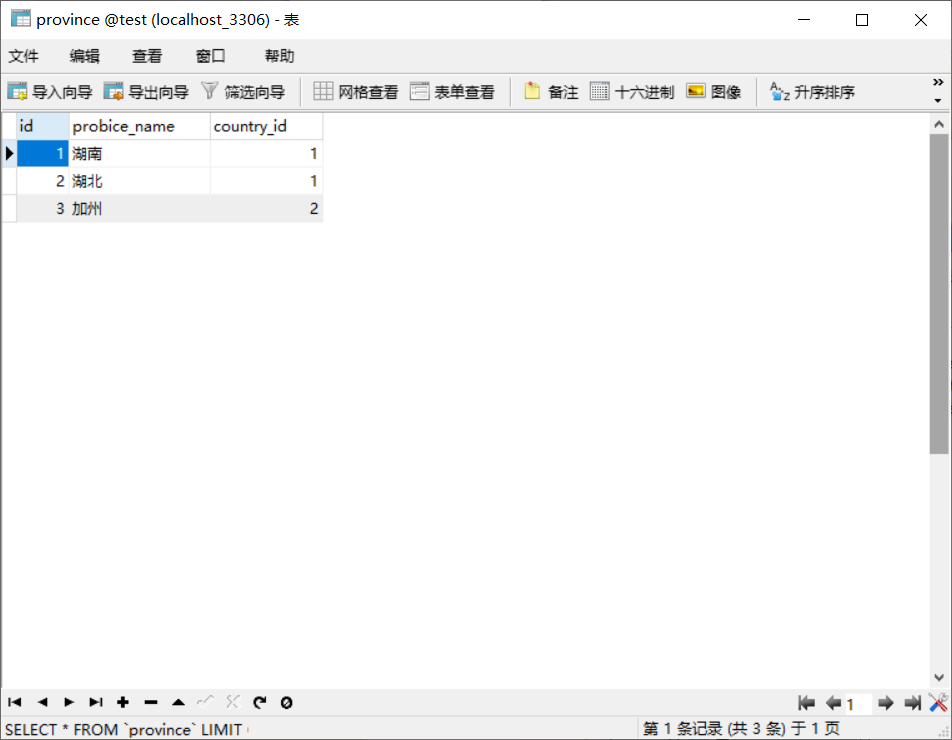
Id int(10),

Province\_name varchar(20),

Country\_id int(10)

);

数据自己照着插入



创建城市表并且插入数据

create table city(

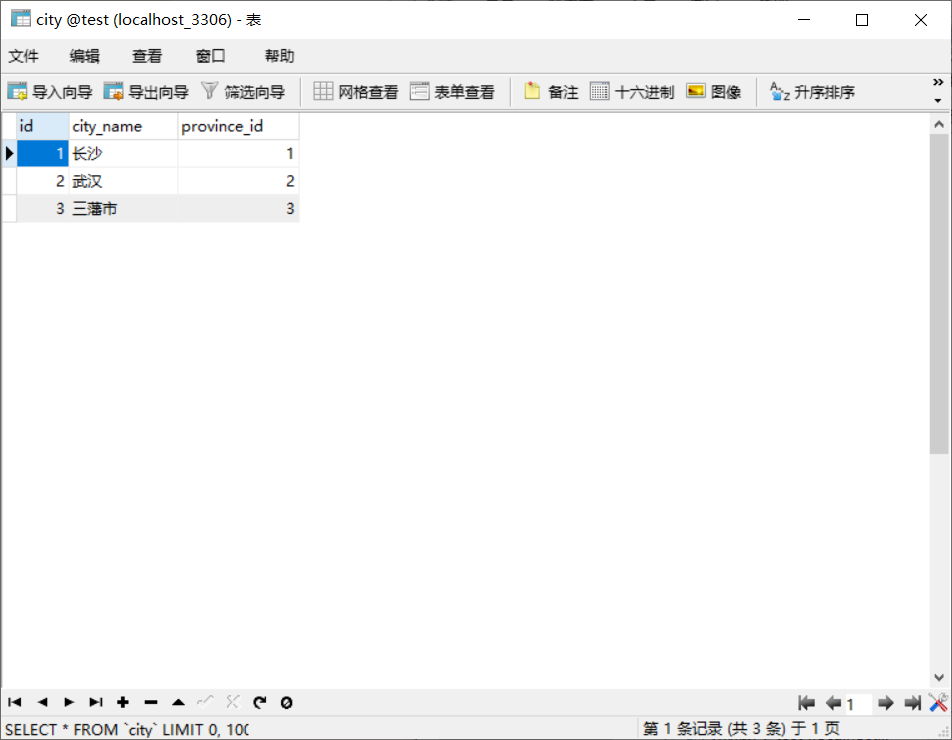
id int(10),

city\_name varchar(20),

province\_id int(10)

);

数据自己照着插



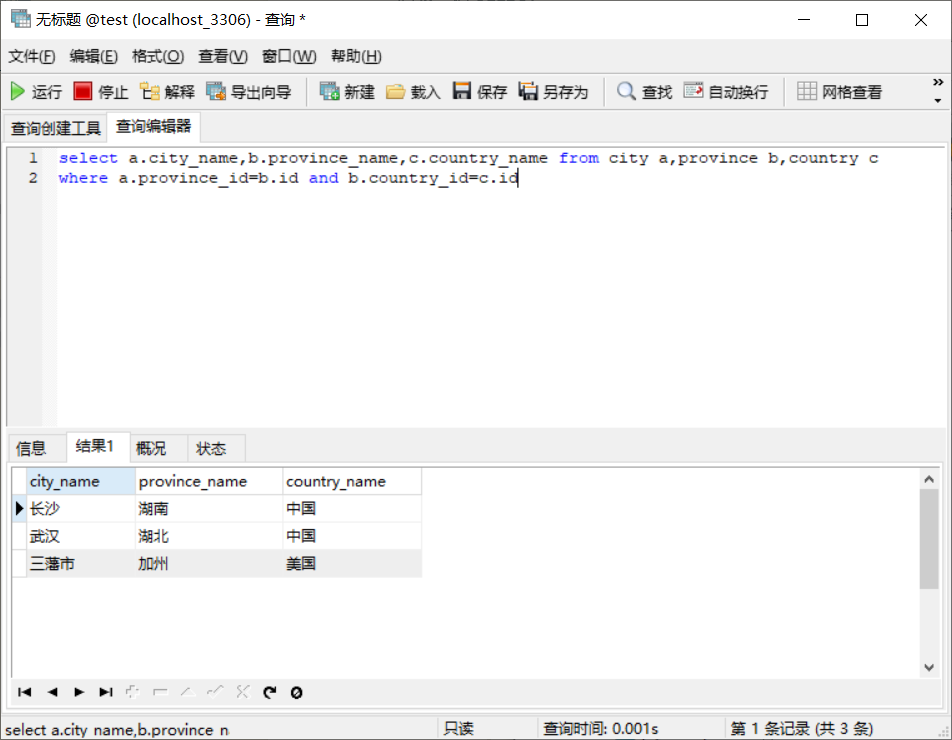
将Country province city 三个表联合查询出城市数据对应的省份和国家名

Select a.city\_name,b.province\_name,c.country\_name

from city a,province b,country c

where a.province\_id=b.id and b.country\_id=c.id

JOIN ON等价WHERE 自己测试



1. 非等值连接

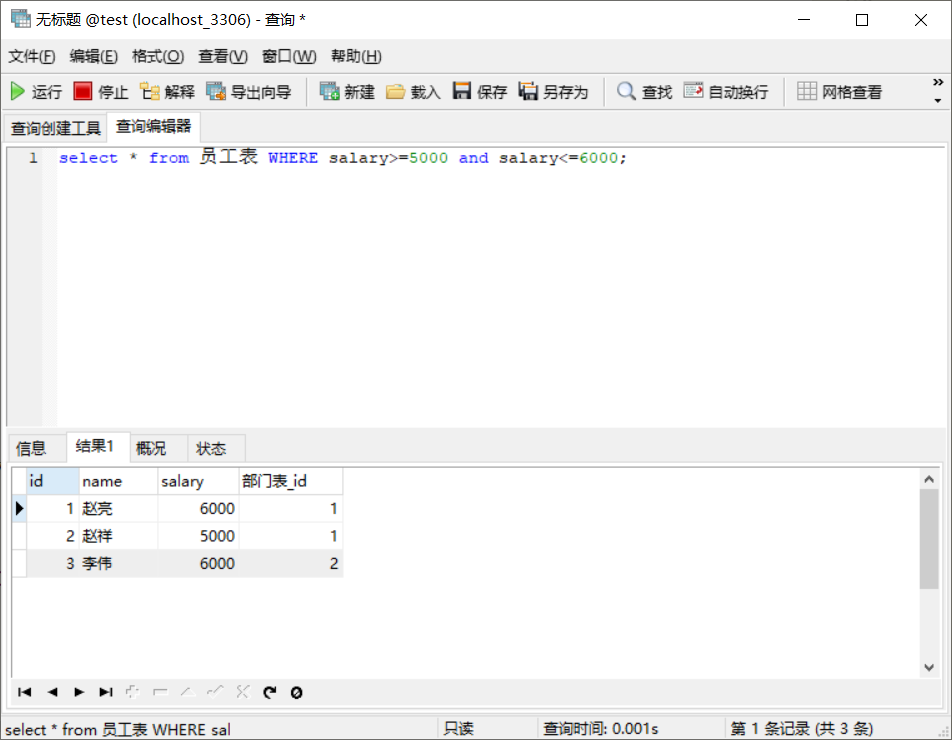
包括>,>=,<,<=,<>,BETWEEN AND

例如：

查询超出工资范围内5000到6000的所有员工信息

select \* from 员工表 WHERE salary>=5000 and salary<=6000;

select \* from 员工表 WHERE salary BETWEEN 5000 and 6000;



1. 外连接(Outer Join)

创建一个一张汽车表car并且插数据

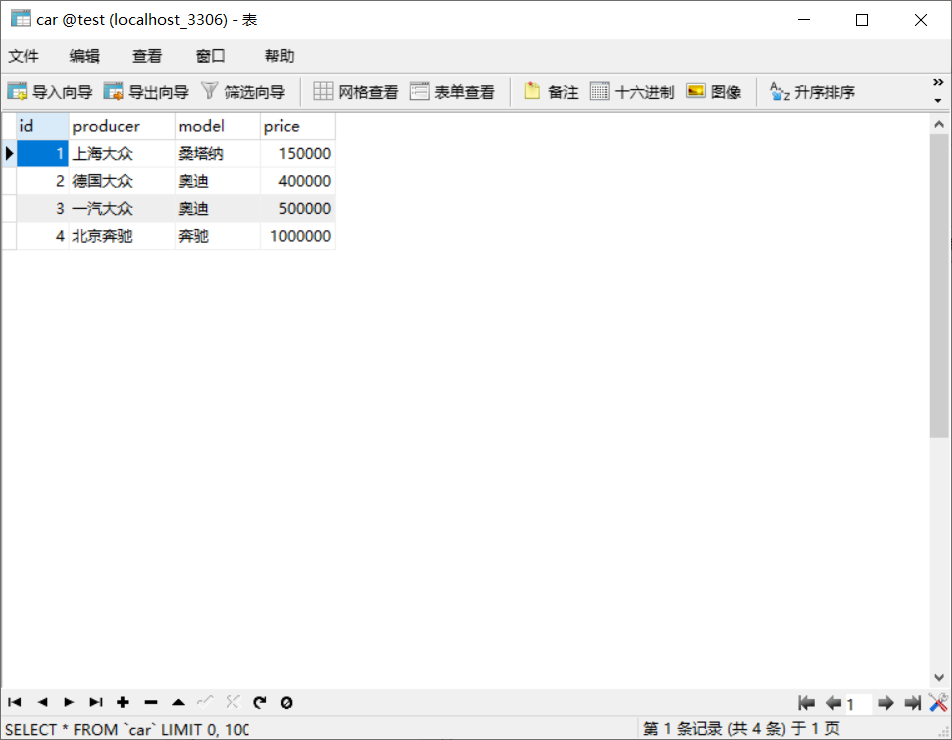
create table car(

id int(10),

producer varchar(15),

model varchar(10),

price float

); 

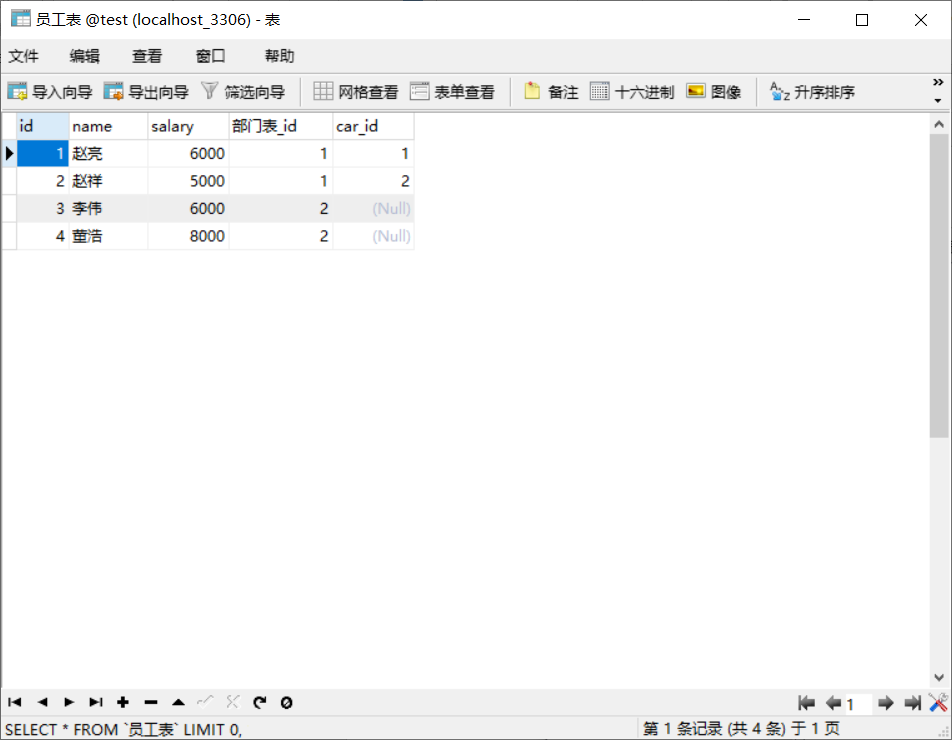
在员工表 添加一个上海大众 car\_id字段

Alter table 员工表 add(car\_id int(10));

设置赵亮拥有 上海大众 car\_id=1,赵祥拥有德国大众 car\_id=2

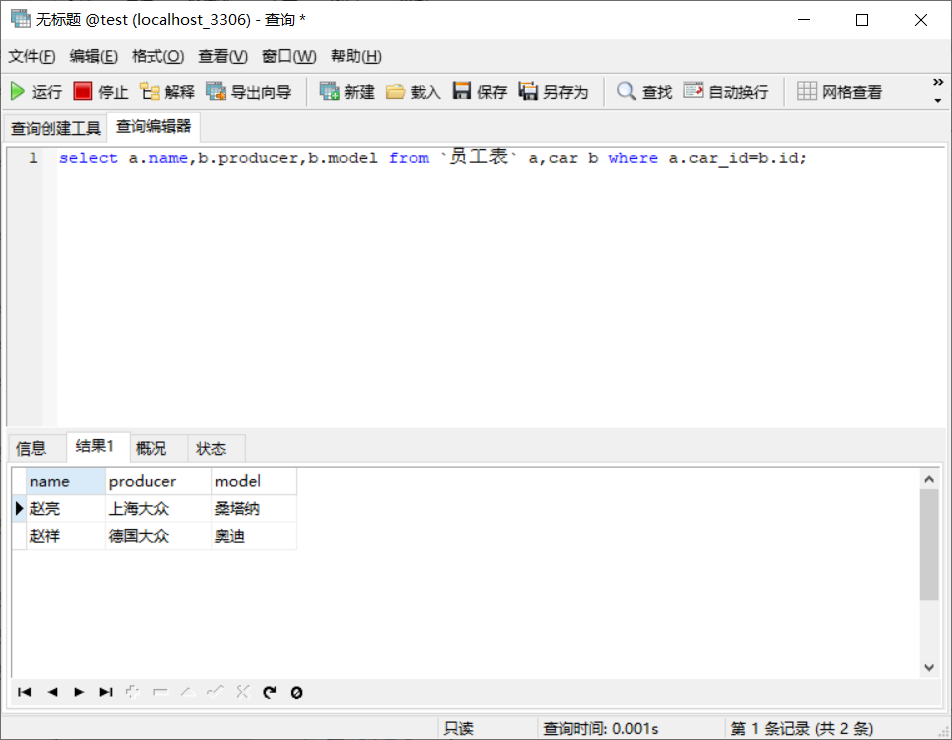
Update `员工表` set car\_id=1 where name='赵亮';

Update `员工表` set car\_id=2 where name='赵祥';



回顾等值连接 查出员工的姓名 和 员工拥有车的信息

select a.name,b.producer,b.model from `员工表` a,car b where a.car\_id=b.id;

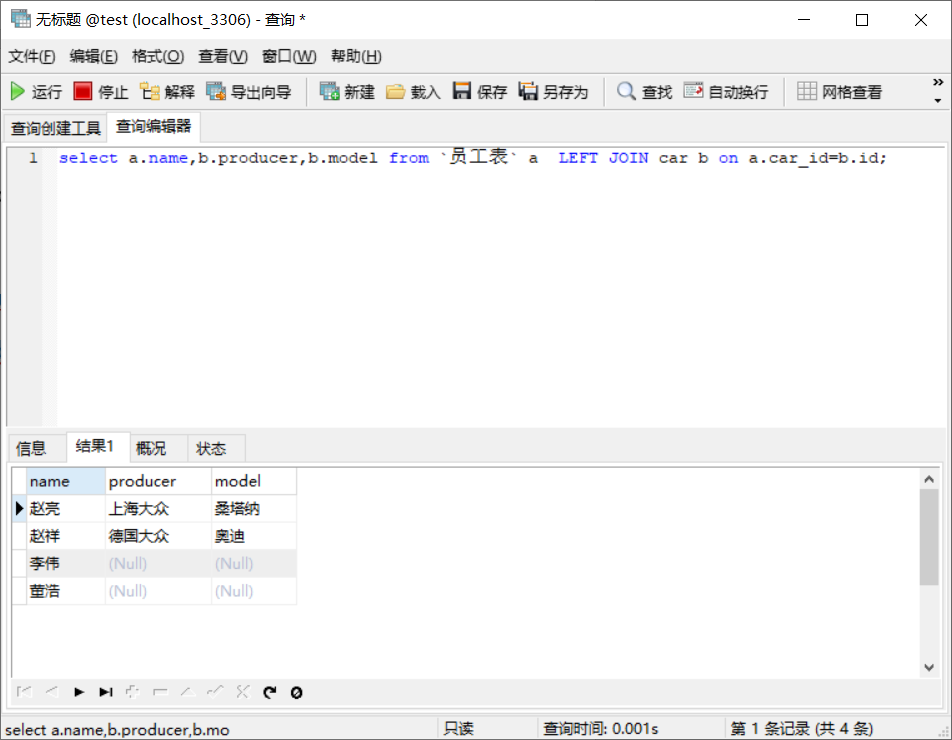


左外连接(Left Outer Join) 左边表不匹配的数据显示，右边表以NULL填充

select a.name,b.producer,b.model from `员工表` a LEFT JOIN car b on a.car\_id=b.id;

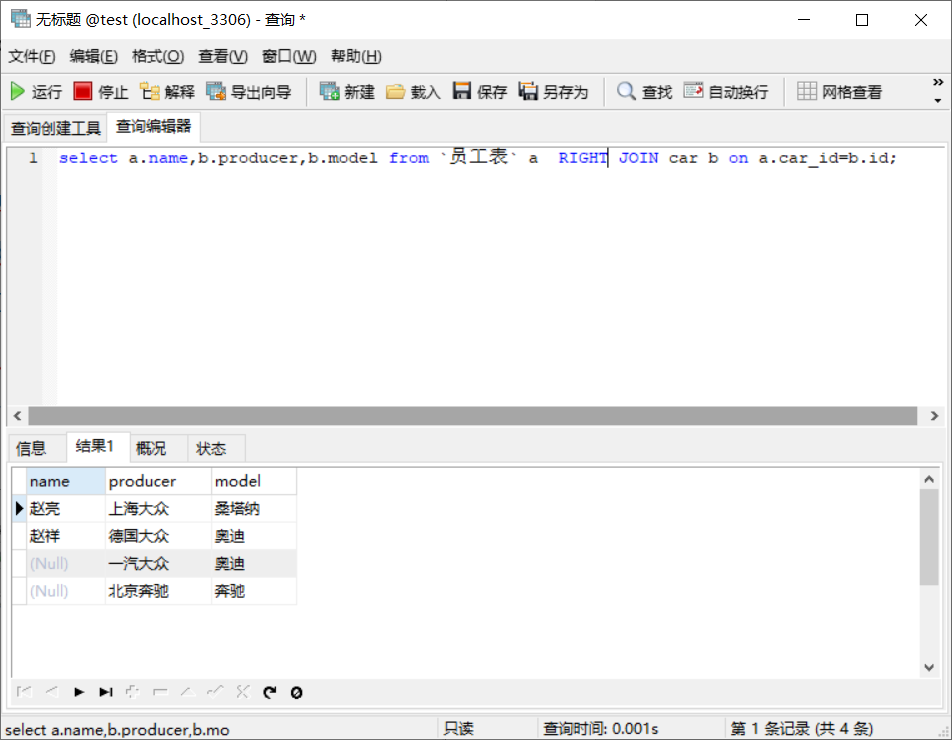
等价不过建议LEFT JOIN ON

select a.name,b.producer,b.model from `员工表` a,car b where a.car\_id=b.id(+);



右外连接(Right Outer JOIN) 左边表以NULL填充，右边表不匹配的数据显示

select a.name,b.producer,b.model from `员工表` a RIGHT JOIN car b on a.car\_id=b.id;



全外连接(FULL Join) 左右表不匹配的表的数据全显示

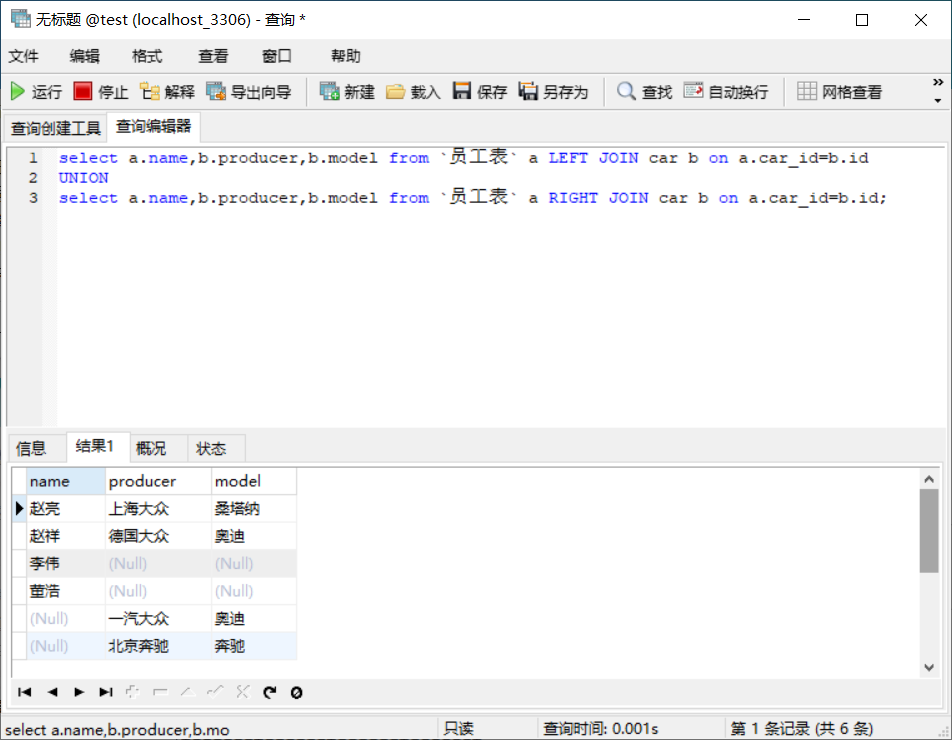
select a.name,b.producer,b.model from `员工表` a FULL JOIN car b on a.car\_id=b.id;(ORACLE用法)

下面是MYSQL用法

select a.name,b.producer,b.model from `员工表` a LEFT JOIN car b on a.car\_id=b.id

UNION

select a.name,b.producer,b.model from `员工表` a RIGHT JOIN car b on a.car\_id=b.id;



1. 自连接(Self Join)

自连接就是一个表 自己和自己关联查询

创建一个产品的分类表 category

并且分类的上级和子级

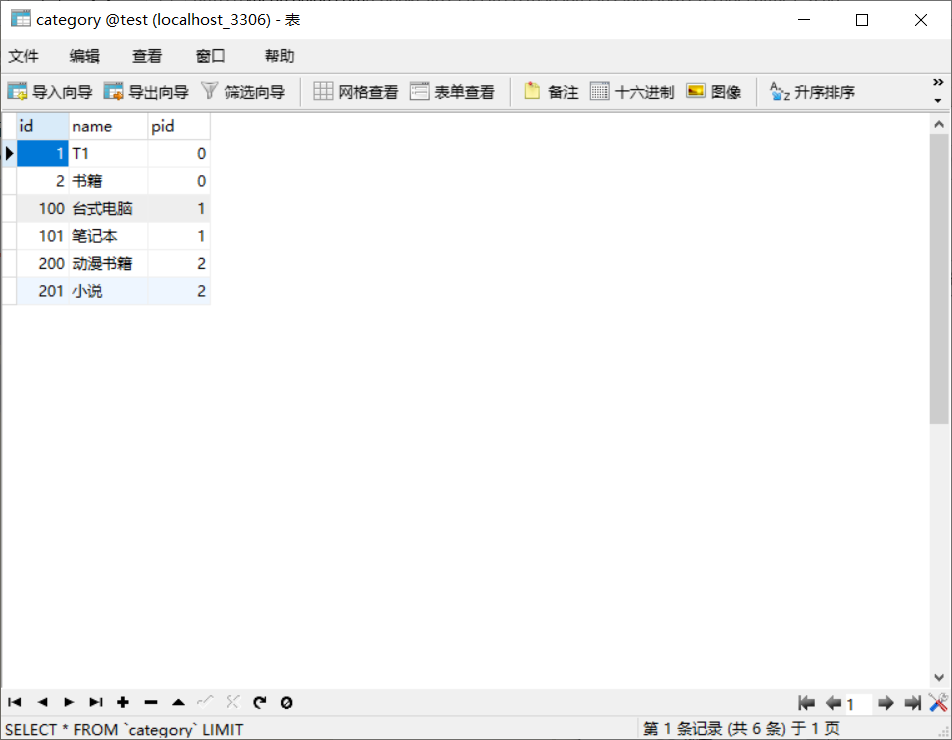
create table category(

id int(10),

name varchar(15),

pid int(10)

);

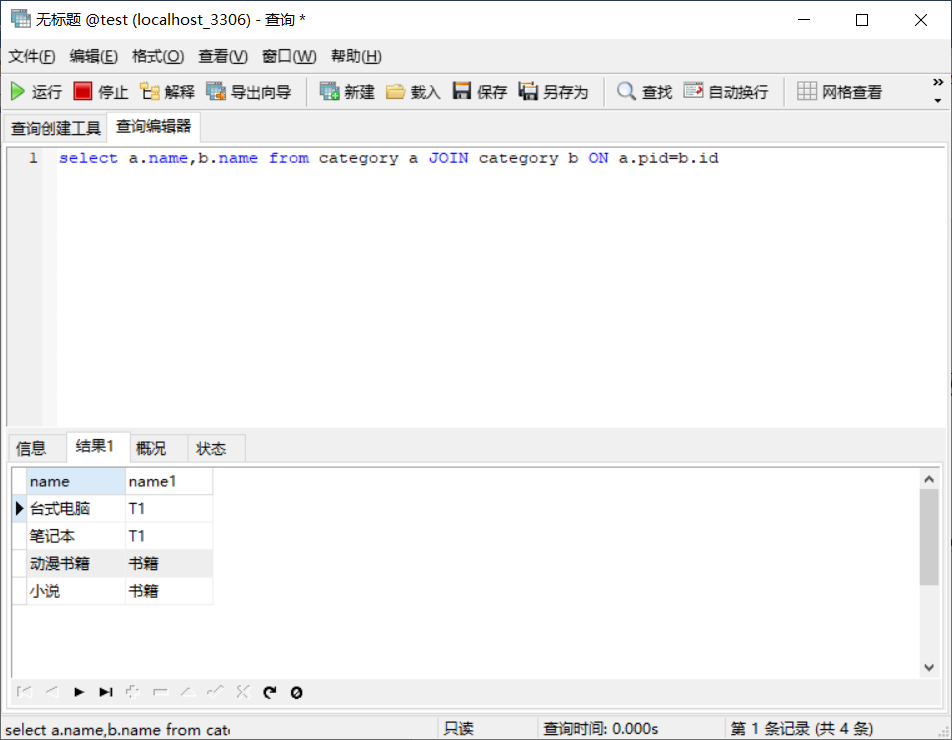


查询出分类信息 和 对应的上级分类名称

两者等价(牢记)

select a.name,b.name from category a,category b where a.pid=b.id

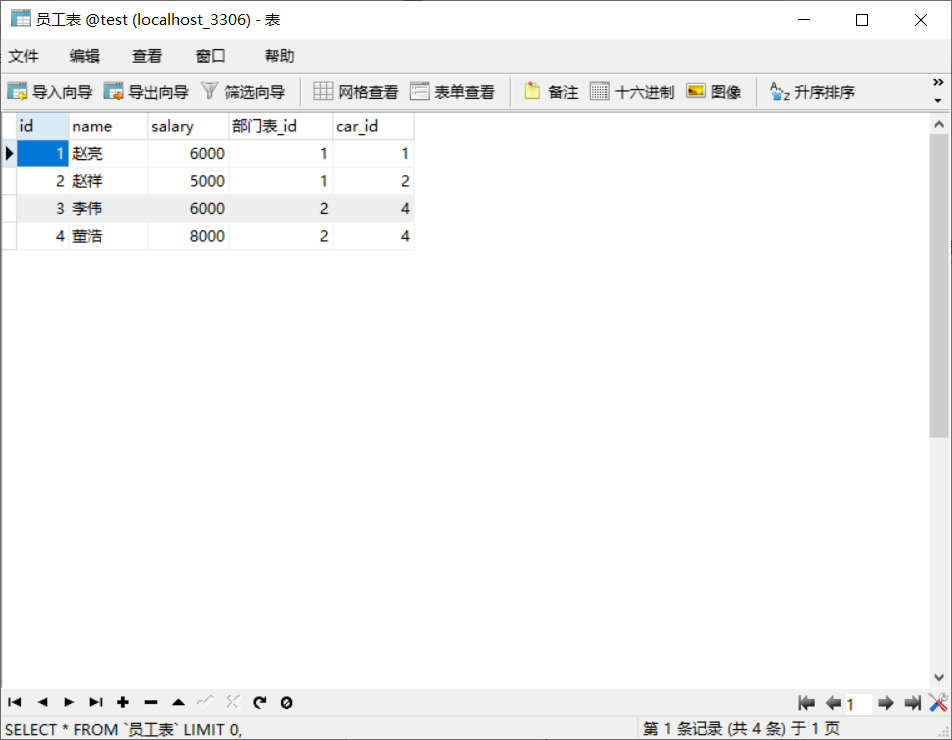
select a.name,b.name from category a JOIN category b ON a.pid=b.id



1. Group By 和 having查询
2. 基于多字段分组

//修改 员工表 car\_id

Update 员工表 set car\_id=4 where id IN(3,4);



//按照部门和员工拥有的车类型，统计人数

select 部门表\_id,car\_id,count(\*) from 员工表

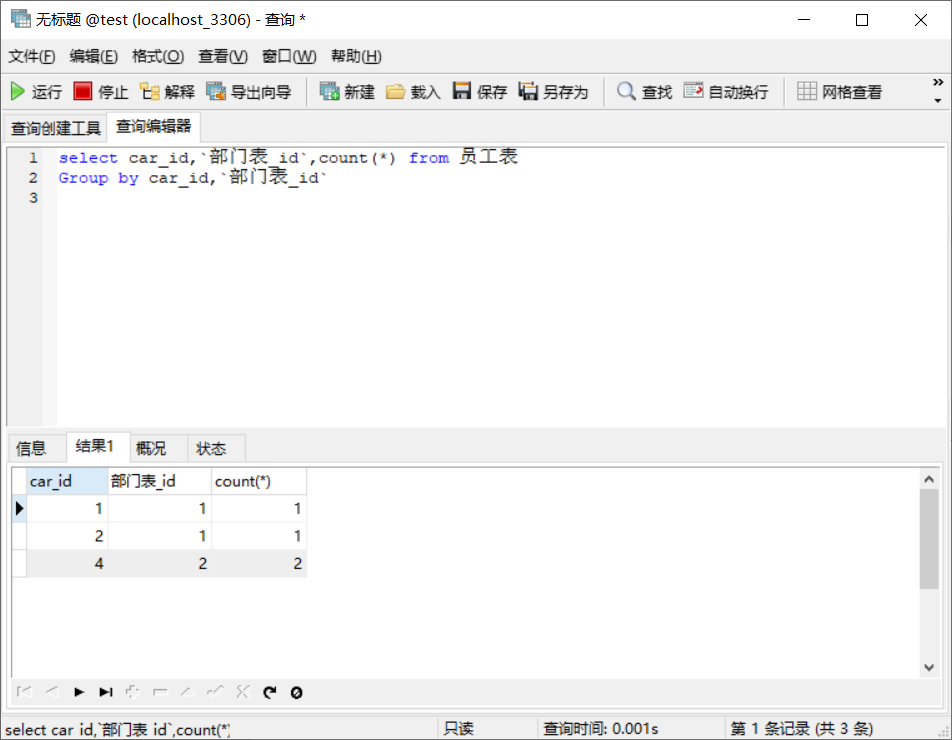
Group by `部门表\_id`,car\_id;



可以比对出如果两者交换顺序，group by 也有顺序，select后面的顺序为最终结果，如果前后两者顺序颠倒，以select后面的顺序为主.(提醒)

select car\_id,`部门表\_id`,count(\*) from 员工表

Group by car\_id,`部门表\_id`



2．HAVING(MYSQL允许Group By使用别名，下面是原生SQL不允许，发生冲突考研以SQL为主)

WHERE 语句不许用分组函数，由于处理顺序的原因

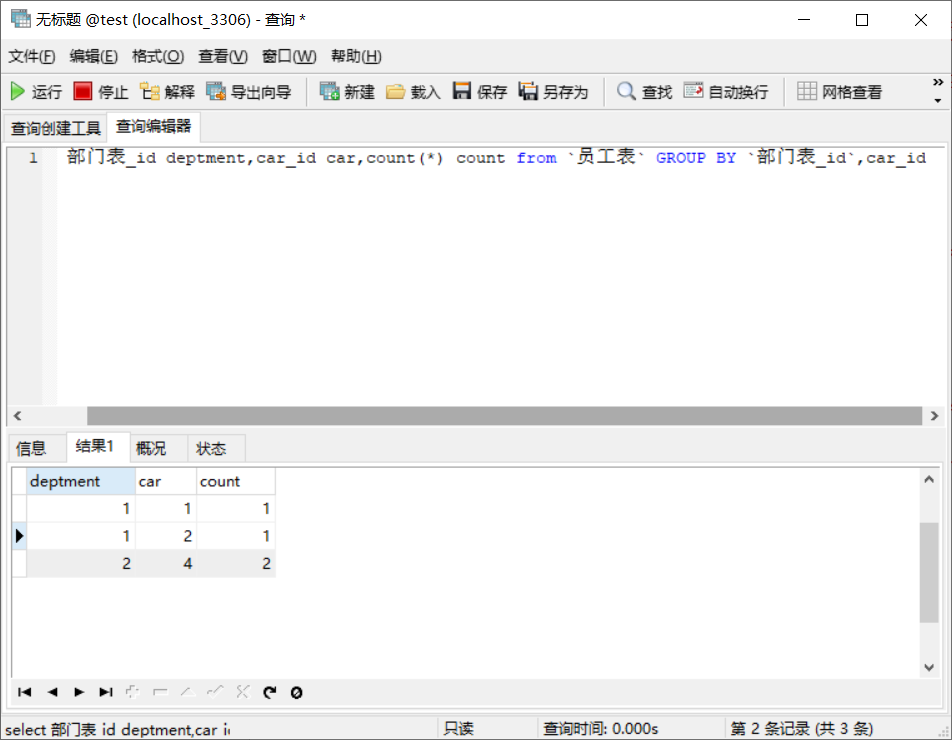
WHERE，GROUP，HAVING字句不允许使用SELECT 字段/表达式列表的别名(MYSQL允许GROUP BY)

//错误用法：(HAVING必须跟在Group BY后面做过滤)

select 部门表\_id deptment,car\_id car,count(\*) count from `员工表` GROUP BY car,deptment

//正确用法:

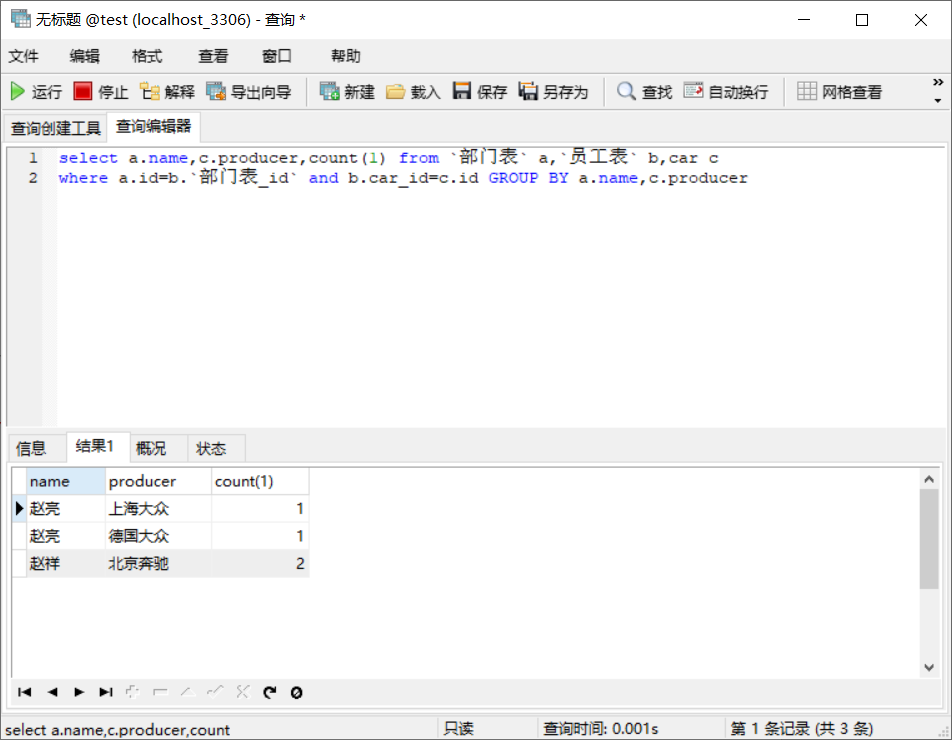
select 部门表\_id deptment,car\_id car,count(\*) count from `员工表` GROUP BY `部门表\_id`,car\_id



多表联合分组，按照部门和员工拥有的车类型，统计人数，显示部门名称和车名

select a.name,c.producer,count(1) from `部门表` a,`员工表` b,car c

where a.id=b.`部门表\_id` and b.car\_id=c.id GROUP BY a.name,c.producer



1. 子查询

SELECT 字段列表

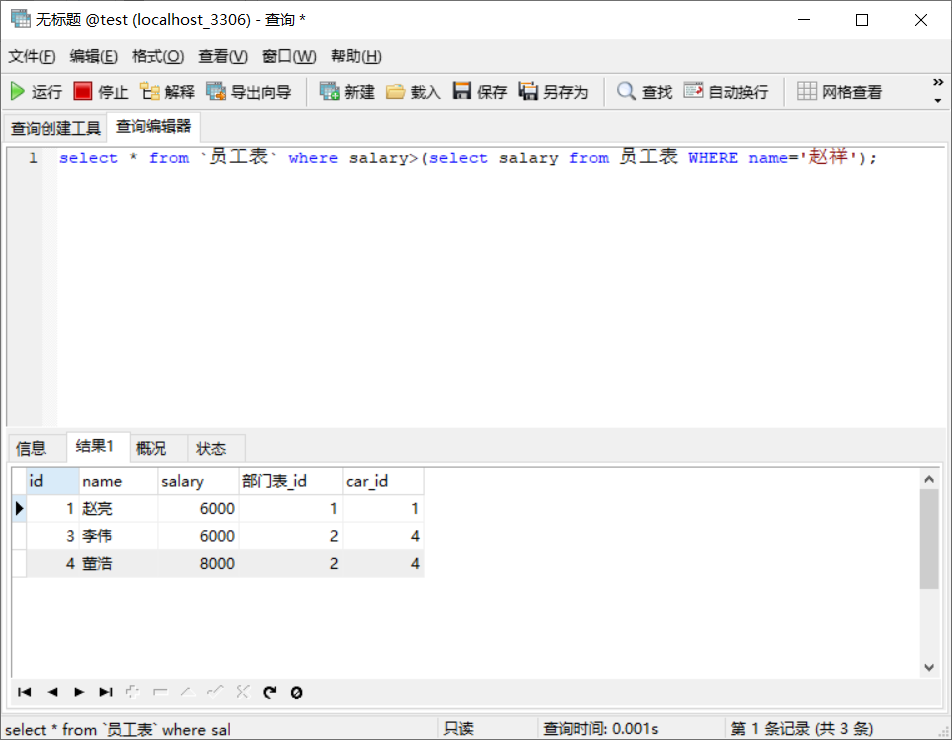
FROM 表名

WHERE 表达式，运算符，子查询;

例如：

查询员工的工资 大于 赵祥的工资 员工所有数据

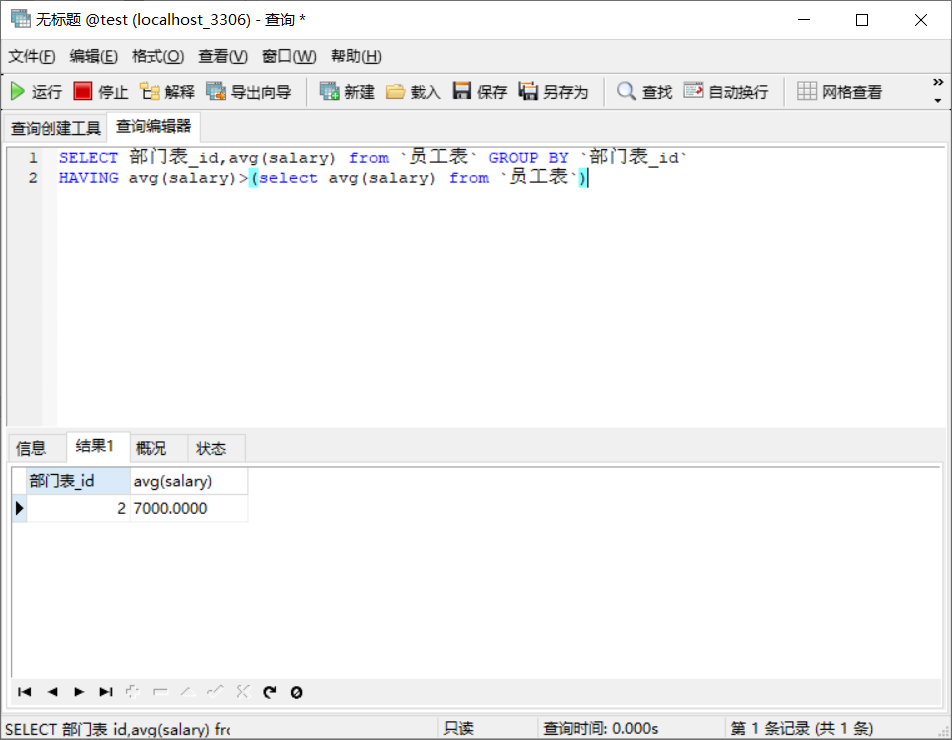
select \* from `员工表` where salary>(select salary from 员工表 WHERE name='赵祥');



查询出部门平均工资 高于 整体平均工资的部门编号

SELECT 部门表\_id,avg(salary) from `员工表` GROUP BY `部门表\_id`

HAVING avg(salary)>(select avg(salary) from `员工表`)

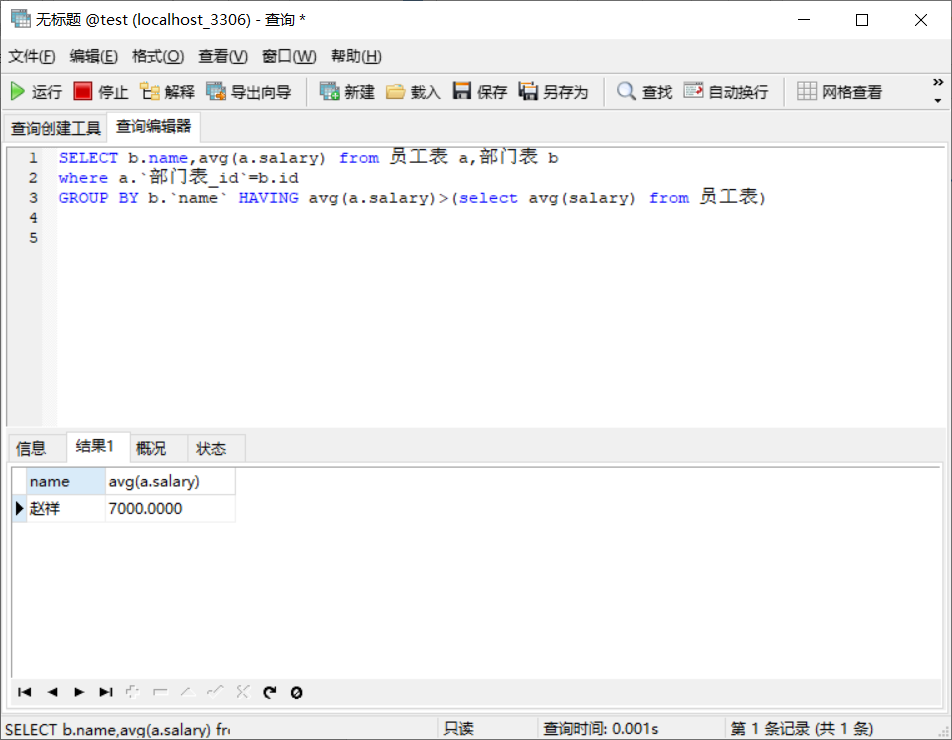


查询出部门平均工资 高于 整体平均工资的部名称

SELECT b.name,avg(a.salary) from 员工表 a,部门表b

where a.`部门表\_id`=b.id

GROUP BY b.`name` HAVING avg(a.salary)>(select avg(salary) from 员工表)



1. 单行子查询

返回单行单列的查询,可以用(=,>,<,>=,<=,<>)

1. 子查询返回单行单列的结果

查询工资员工的工资 大于 赵祥的工资 的员工的所有数据

select \* from `员工表` where salary>(select salary from 员工表 WHERE name='赵祥');

上面有不再测试

1. 子查询未返回任何行

查询员工的工资 大于 董浩的工资 的员工所有数据

select \* from `员工表` where salary>(select salary from 员工表 WHERE name='董浩');

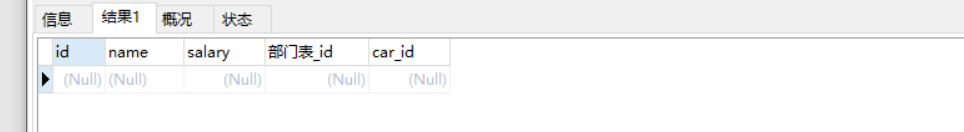
select \* from `员工表` where salary>NULL;

上面两者等价

注意要判断是否未空要用IS NULL或者 IS NOT NULL

select \* from `员工表` where salary IS NULL;

select \* from `员工表` where salary IS NOT NULL;



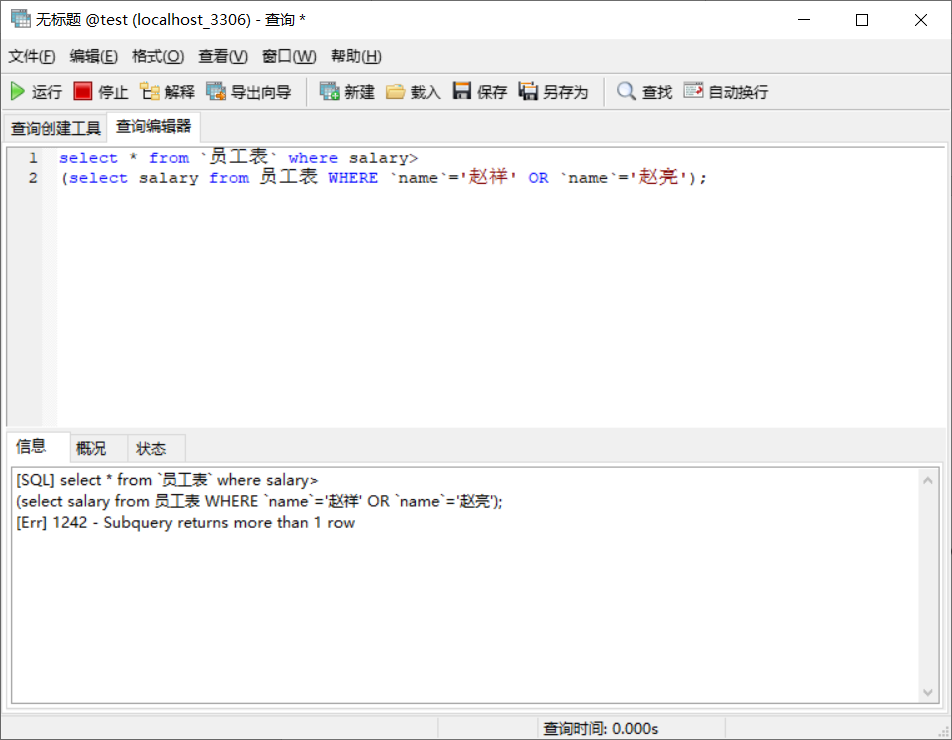
子查询返回多行结果

此时不允许用(=,<,>,<=,>=,<>)运算符

//错误示范:

select \* from `员工表` where salary>

(select salary from 员工表 WHERE `name`='赵祥' OR `name`='赵亮');



3.多行子查询

返回多行单例的查询结果，使用(IN,ALL,ANY,EXISTS)运算符

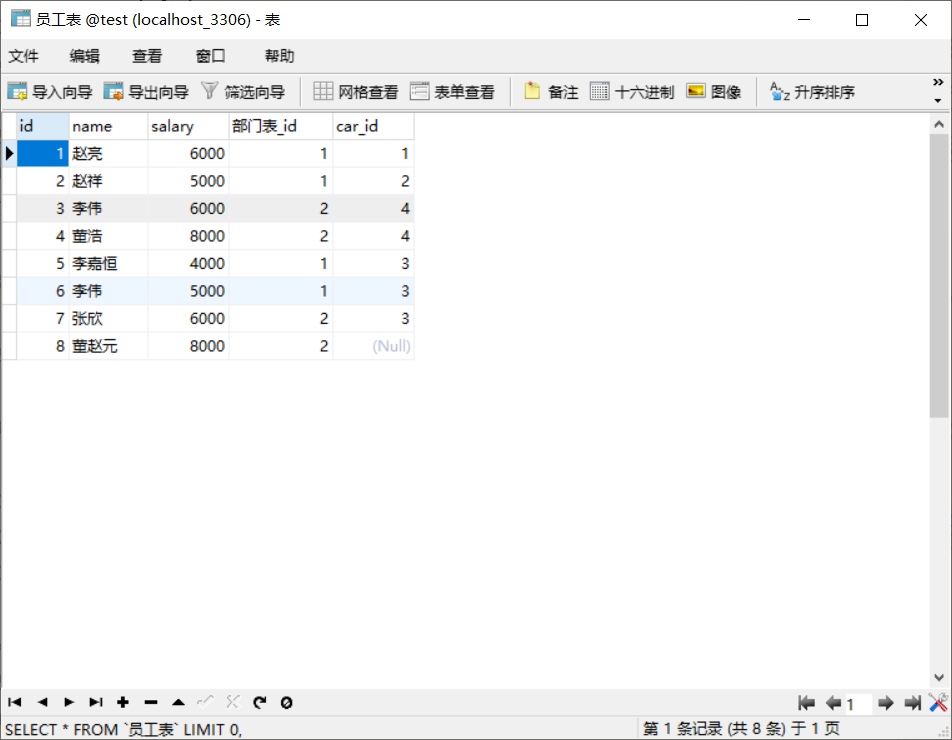
IN:等于列表中的任何一个 id IN(3,4)

ALL:和子查询返回的所有值比较 id>ALL(3,4)

ANY:和子查询返回的任意一个值比较 id>ANY(3,4)

EXISTS:判断子查询是否有返回结果 exists不为空NULL 为true否则为false

插入数据如下图所示:

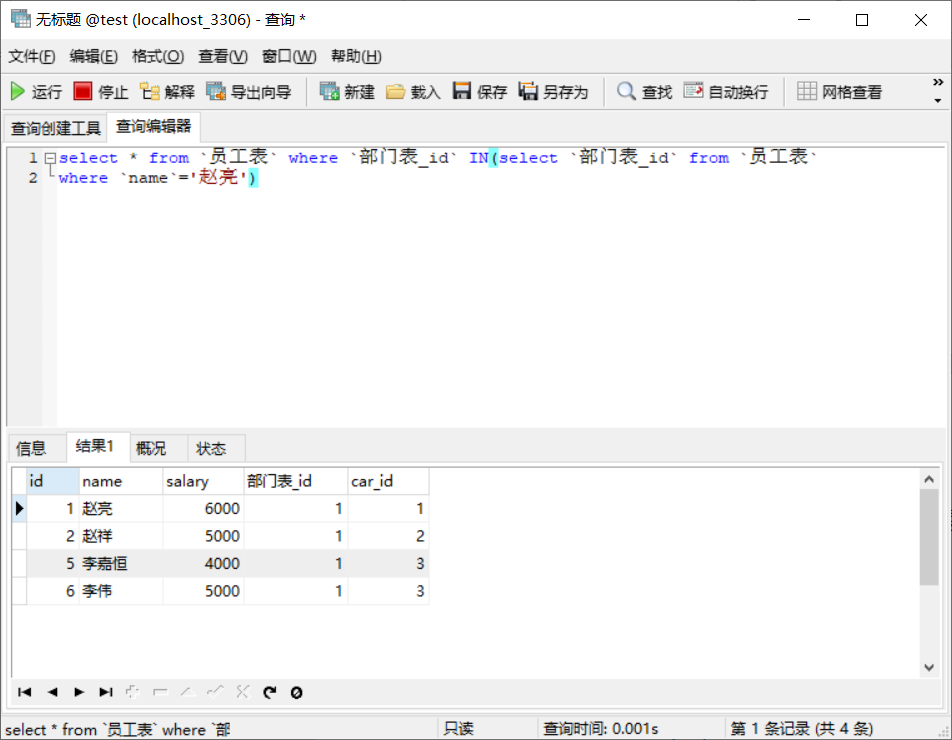


IN查询出 与赵亮部门相同的所有员工信息

select \* from `员工表` where `部门表\_id` IN(select `部门表\_id` from `员工表`

where `name`='赵亮')

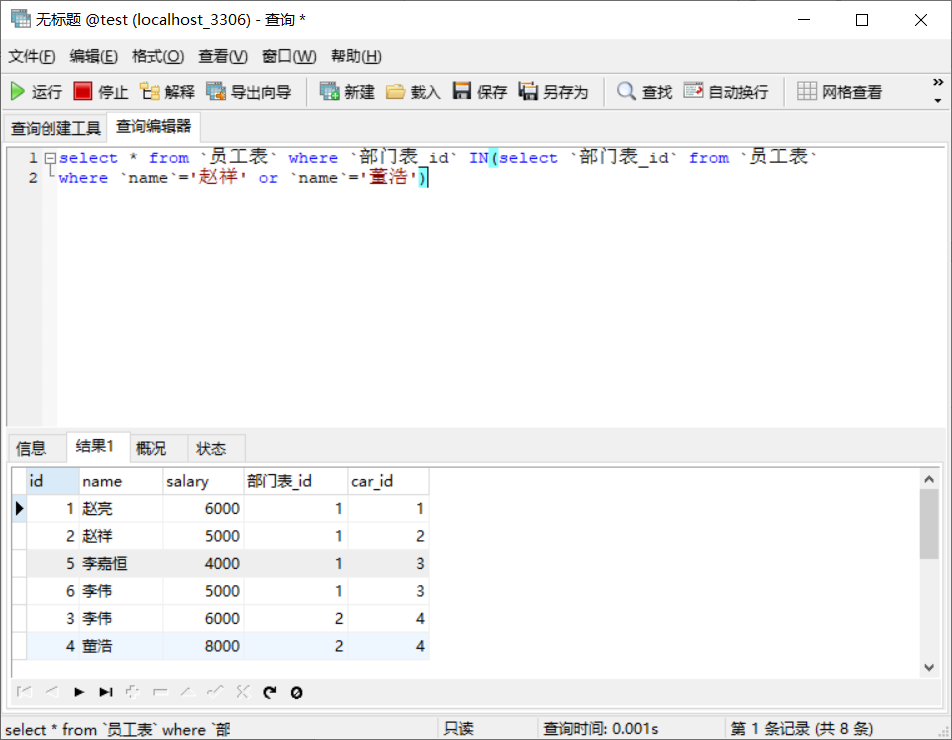
3



IN查询出 与赵祥 或 董浩 部门相同的所有员工信息

select \* from `员工表` where `部门表\_id` IN(select `部门表\_id` from `员工表`

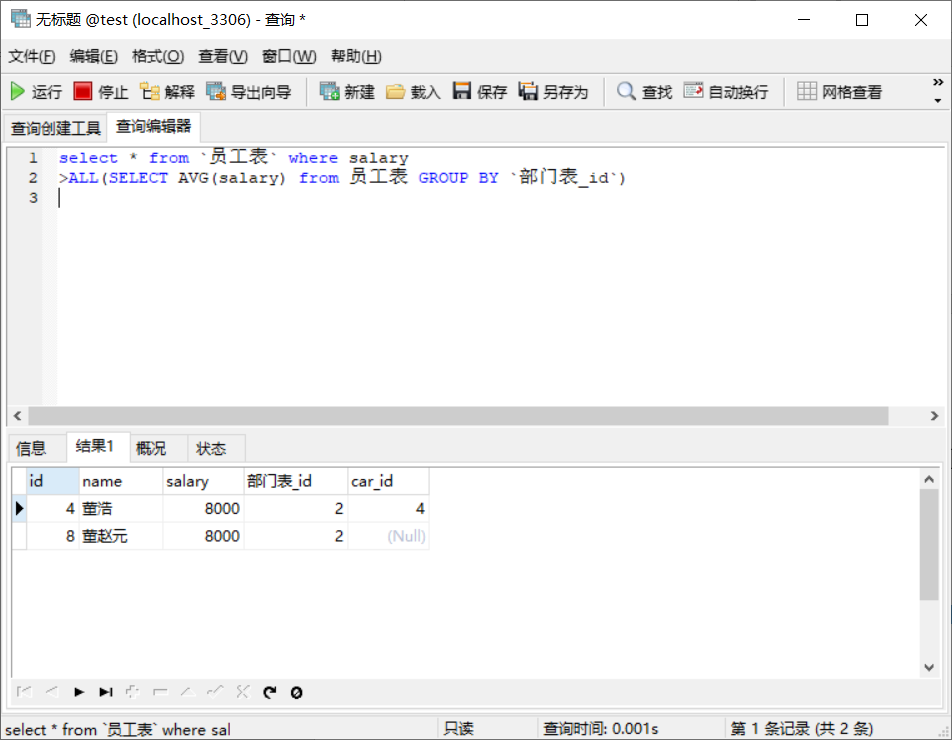
where `name`='赵祥' or `name`='董浩')



ALL查询查询工资 高于 所有的部门平均工资 的员工信息

select \* from `员工表` where salary

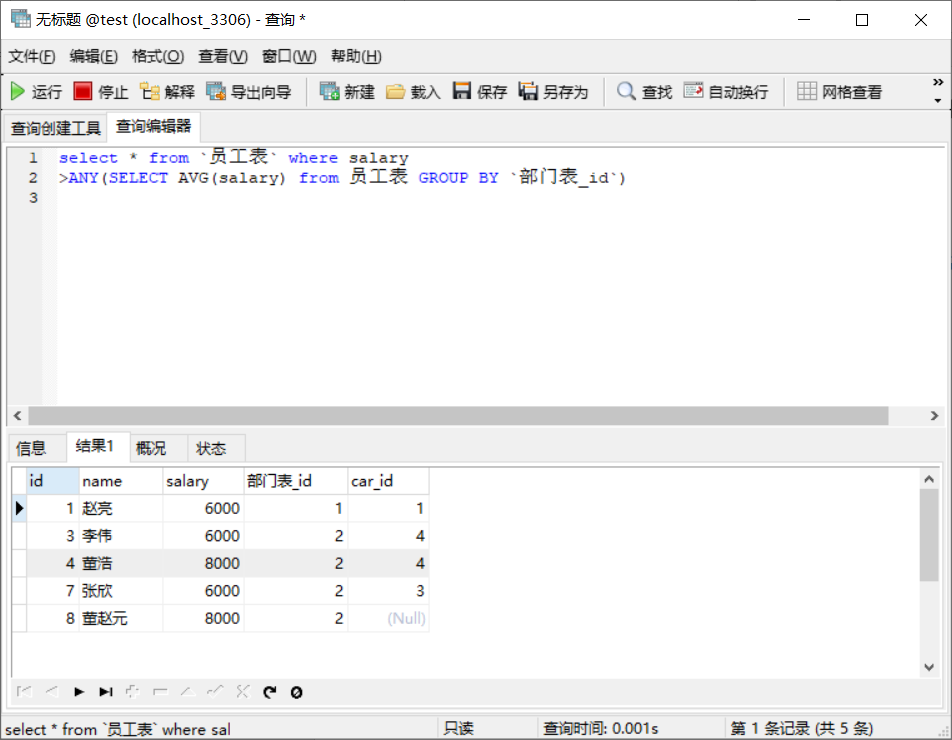
>ALL(SELECT AVG(salary) from 员工表 GROUP BY `部门表\_id`)



ANY查询工资 高于 任意部门的平均工资的员工信息

select \* from `员工表` where salary

>ANY(SELECT AVG(salary) from 员工表 GROUP BY `部门表\_id`)



EXISTS 判断子查询是否返回结果 TRUE，FALSE,如果赵亮所在的部门存在，返回所有员工信息

IN 和 EXISTS的区别

IN 不能处理NULL

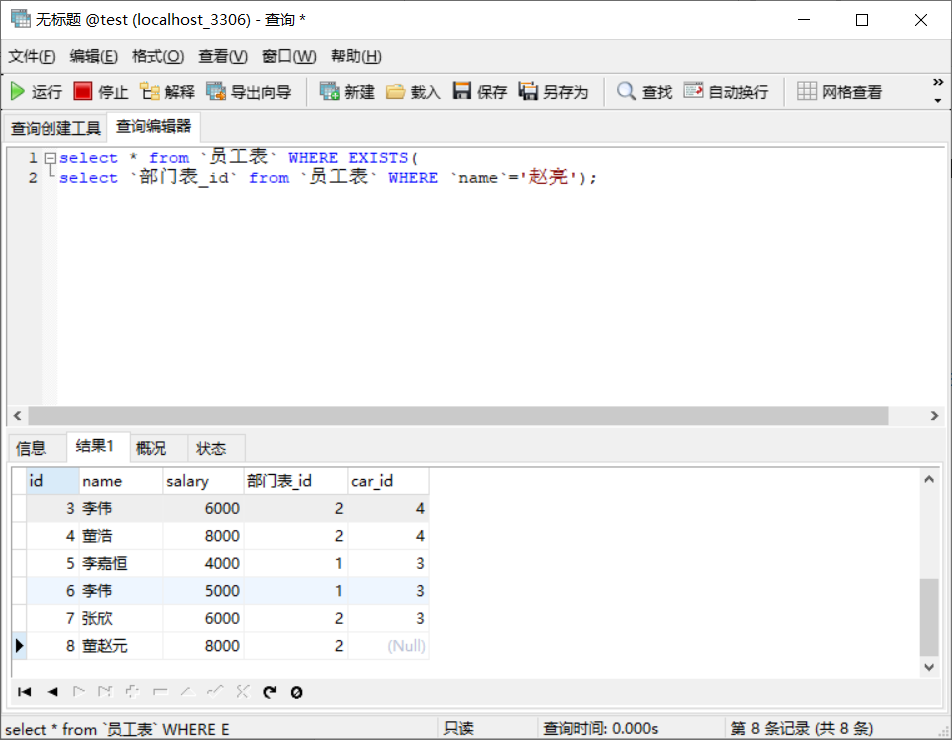
EXISTS 可以匹配NULL

主表数据小而从表数据非常大时，exists的查询效率高

从表数据量小而主表数据量非常大时，in查询效率高

select \* from `员工表` WHERE EXISTS(

select `部门表\_id` from `员工表` WHERE `name`='赵亮');



3.多列查询

如果是单行数据用(<,>,=,>=,<=)

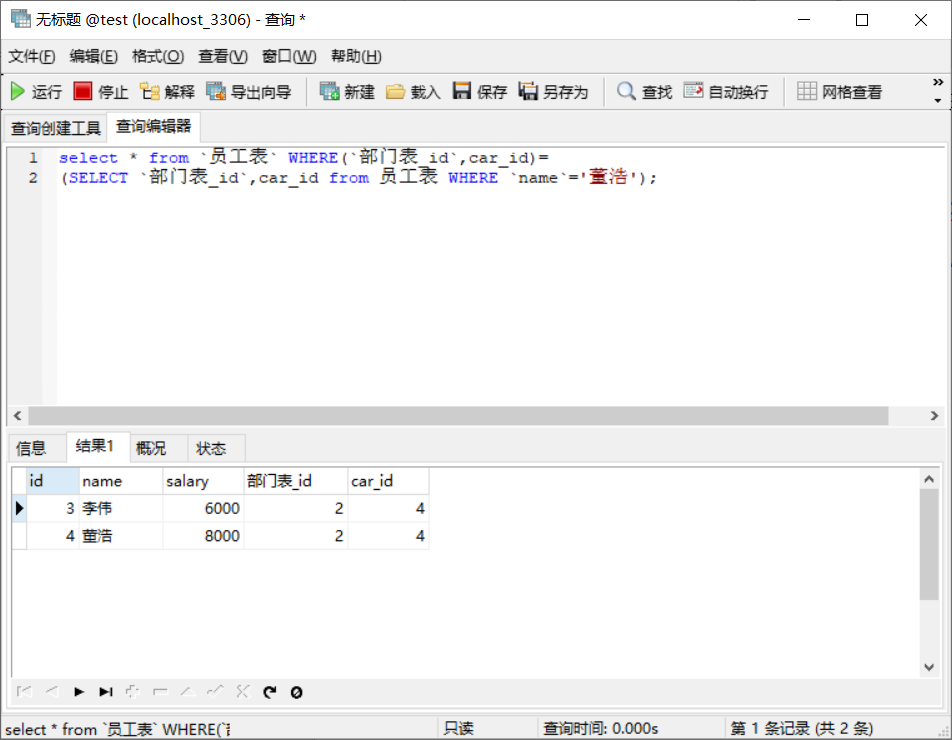
如果是多行数据用(IN,ALL,ANY)

例如:

查询出与 董浩 所在部门 和同一类型的车的 员工信息

select \* from `员工表` WHERE(`部门表\_id`,car\_id)=

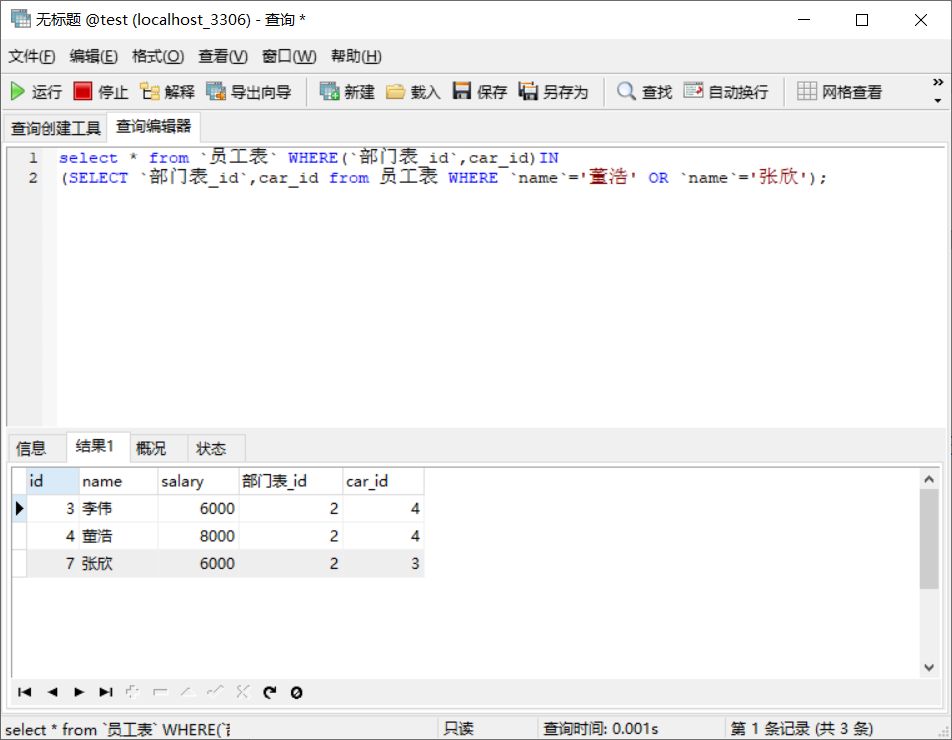
(SELECT `部门表\_id`,car\_id from 员工表 WHERE `name`='董浩');



查询出与 董浩和张欣 所在部门 和同一类型的车的 员工信息

select \* from `员工表` WHERE(`部门表\_id`,car\_id)IN(注意)

(SELECT `部门表\_id`,car\_id from 员工表 WHERE `name`='董浩' OR `name`='张欣');



7相关子查询

子查询要用到主查询的结果，也就是子查询和主查询相互依赖

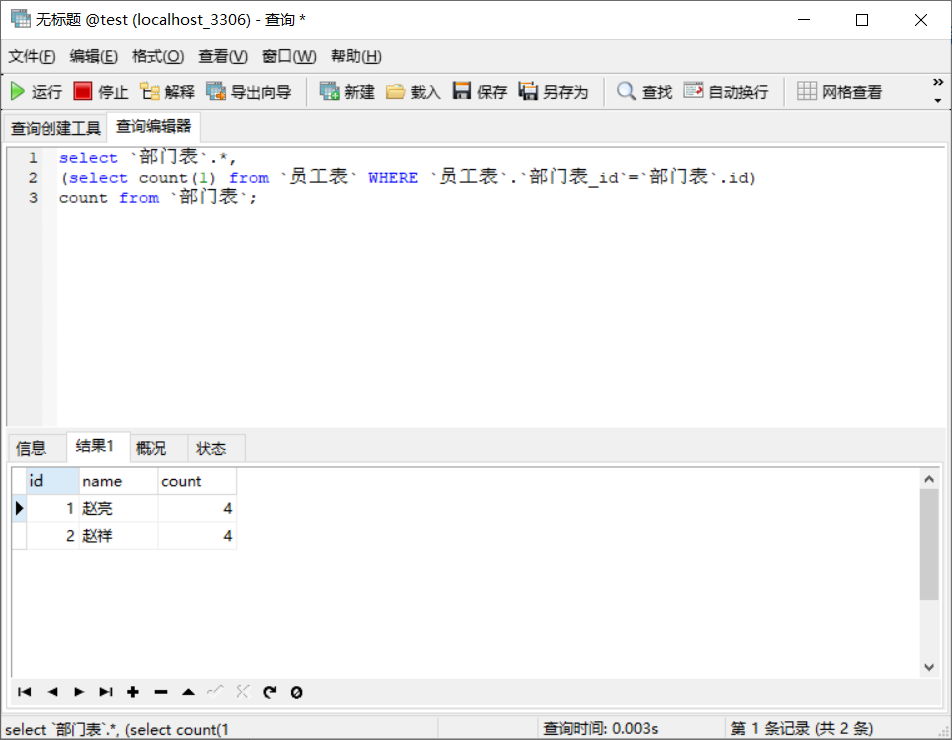
例如:

查询出所有的部门和对应的员工数量

select `部门表`.\*,

(select count(1) from `员工表` WHERE `员工表`.`部门表\_id`=`部门表`.id)

count from `部门表`;

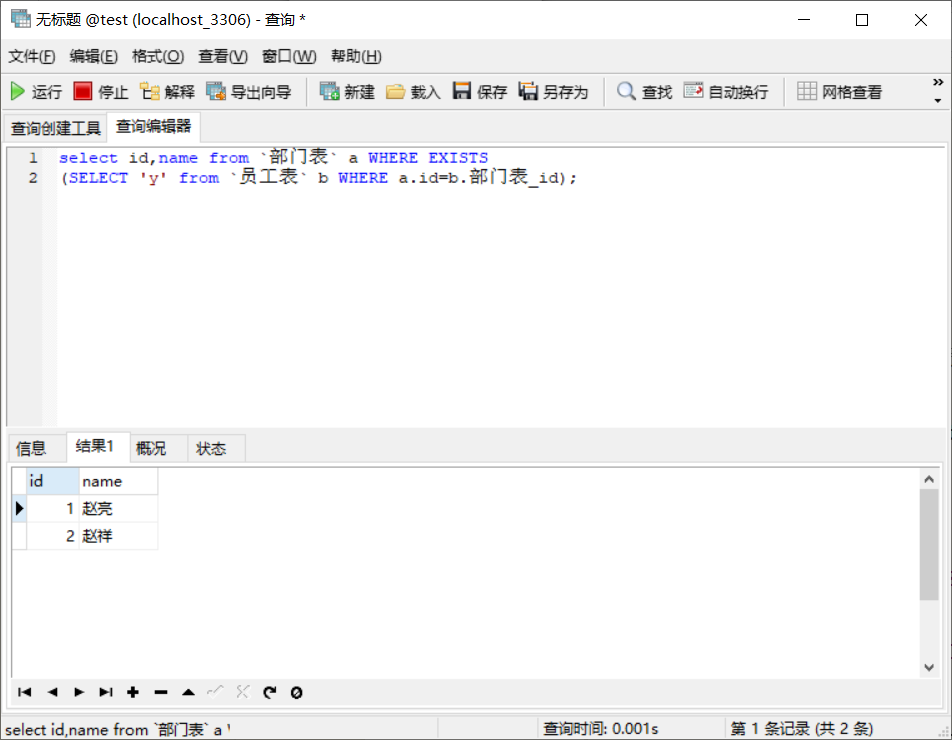


返回已经雇佣了员工的部门

insert into `部门表` values(3,'质量')

select id,name from `部门表` a WHERE EXISTS

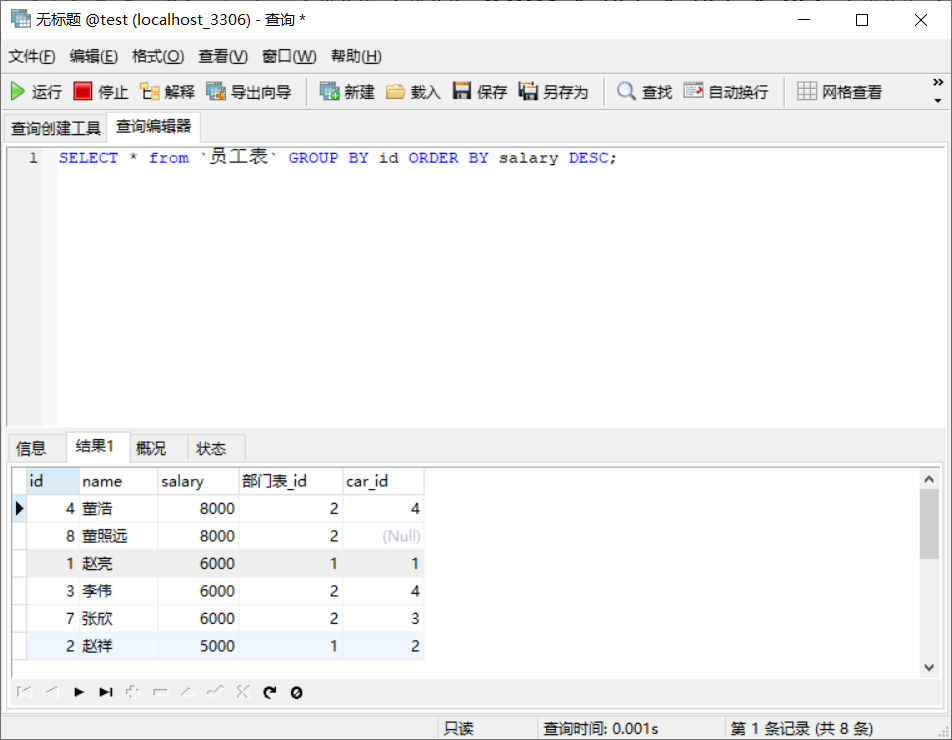
(SELECT 'y' from `员工表` b WHERE a.id=b.部门表\_id);



8获取最前面的limit行记录

1. 返回全部查询结果 工资降序排列

SELECT \* from `员工表` GROUP BY id ORDER BY salary DESC;



2. imit,n,m

1.非排序查询(无 ORDER BY)

返回员工前5条数据

SELECT \* from `员工表` LIMIT 5;

Oracle(SQL) 是SELECT ROWNUM，员工表.\* from `员工表` where LIMIT 5;

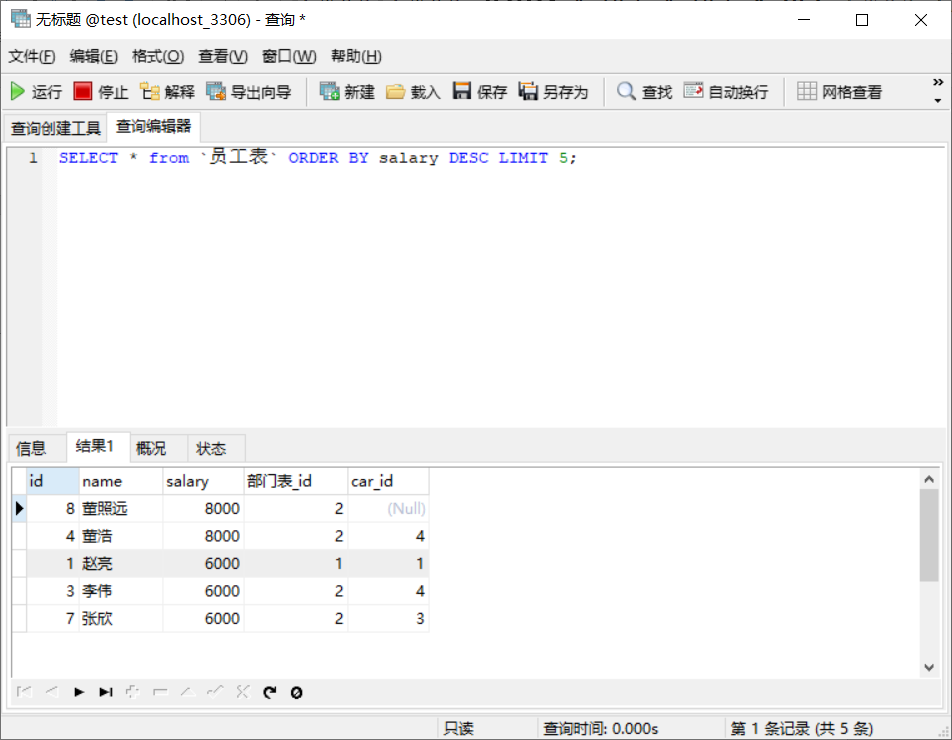


2.排序查询(GROUP BY)

返回工资最高的前5名员工信息

SELECT \* from `员工表` ORDER BY salary DESC LIMIT 5;

Oracle(SQL) 是SELECT ROWNUM，员工表.\* from `员工表` where LIMIT 5 ORDER BY salary DESC;



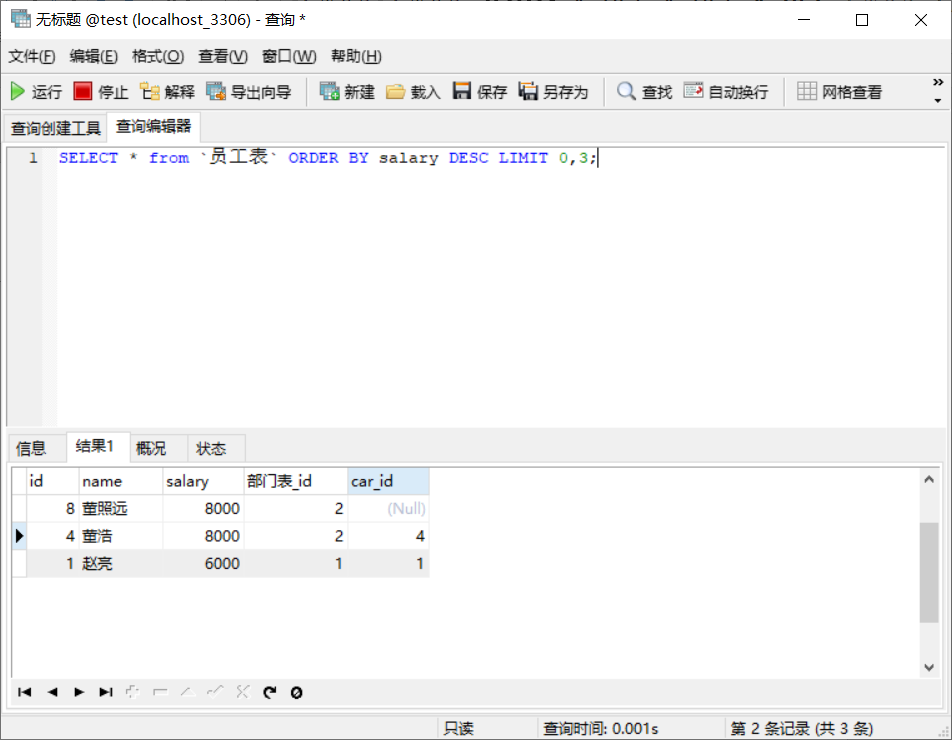
9.分页查询

例如:

按照工资排倒序，返回员工信息，每页显示三条记录，当前显示第一页

SELECT \* from `员工表` ORDER BY salary DESC LIMIT 0,3;

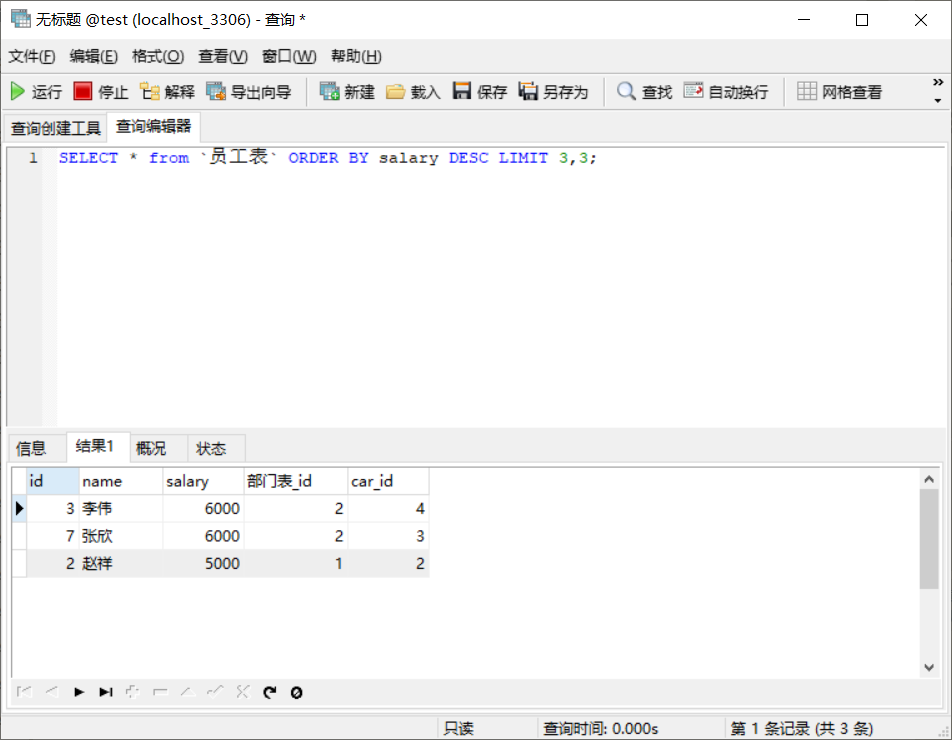
Oracle(SQL) 是SELECT ROWNUM，员工表.\* from `员工表` where LIMIT 0,3 ORDER BY salary DESC;



按照工资排倒序 返回员工信息 每页显示3条记录，当前显示第二页

SELECT \* from `员工表` ORDER BY salary DESC LIMIT 3,3;

Oracle(SQL) 是SELECT ROWNUM，员工表.\* from `员工表` where LIMIT 3,3 ORDER BY salary DESC;



10.查询结果的集合操作

SELECT 语句1 UNION[ALL] SELECT 语句2

ORDER BY(排序) 子句

创建一个账户account表并且插入数据

create table account(

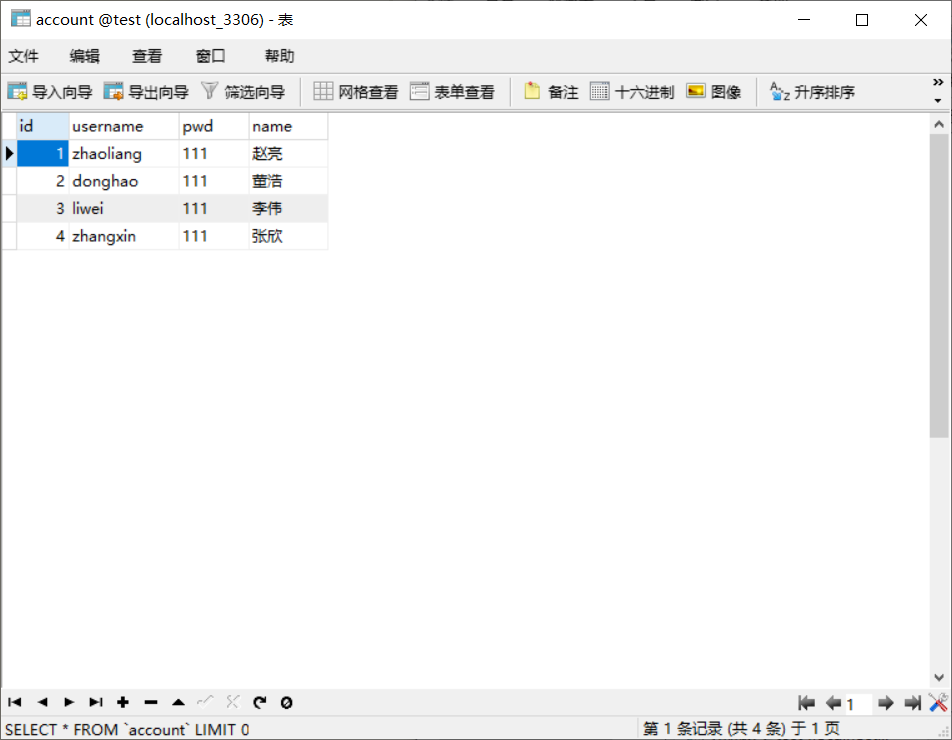
id int(10),

username varchar(10),

pwd varchar(10),

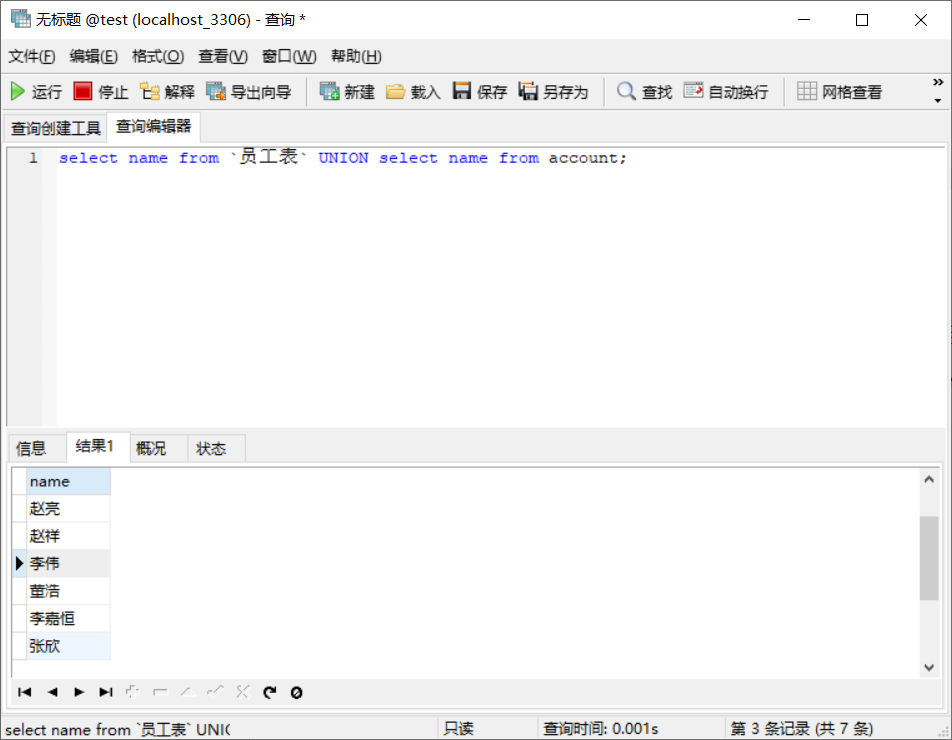
name varchar(10)

);



UNION 计算两个结果的并集，自动去掉结果集中的重复行，从员工表和account表中查出所有的人名

select name from `员工表` UNION select name from account;



UNION计算两个结果的并集，不去掉重复行，从员工表和account表中查出所有的人名

select name from `员工表` UNION ALL select name from account;

