

sql语句顺序

(1)from (3) join (2) on (4) where (5)group by(开始使用select中的别名，后面的语句中都可以使用) (6) avg,sum.... (7)having (8) select (9) distinct (10) order by

Tips:

```
\password 设置密码。  
\q 退出。  
\h 查看SQL命令的解释，比如\h select  
\l 列出所有数据库。  
\c [database_name] 连接其他数据库。  
\d 列出当前数据库的所有表格。  
\d [table_name] 列出某一张表格的结构。  
\du 列出所有用户。  
\e 打开文本编辑器。  
\conninfo 列出当前数据库和连接的信息。  
\? 查看psql命令列表
```

可视化 打开pgAdmin

创建表格

```
![create database SPJ; \c spj; create table S ( Sno char(2) unique, Sname char(6), Status char(2),  
City char(4), primary key(Sno) ); create table P ( Pno char(2) unique, Pname char(6), color char(2),  
weight int, primary key(Pno) );
```

```
create table J ( Jno char(2) unique, Jname char(8), CITY char(4), primary key(Jno) );
```

```
create table SPJ ( Sno char(2), Pno char(2), Jno char(2), QTY int, primary key(Sno,Pno,Jno), foreign  
key(Sno) references S(Sno), foreign key(Pno) references P(Pno), foreign key(Jno) references J(Jno) );
```

```
insert into S(Sno,Sname,Status,City) values ('S1','精益','20','天津'), ('S2','盛锡','10','北京'), ('S3','东方  
红','30','北京'), ('S4','丰泰盛','20','天津'), ('S5','为民','30','上海');
```

```
insert into P(Pno,Pname,color,weight) values ('P1','螺母','红',12), ('P2','螺栓','绿',17), ('P3','螺丝  
刀','蓝',14), ('P4','螺丝刀','红',14), ('P5','凸轮','蓝',40), ('P6','齿轮','红',30);
```

```
insert into J(Jno,Jname,CITY) values ('J1','三建','北京'), ('J2','一汽','长春'), ('J3','弹簧厂','天津'), ('J4','造船  
厂','天津'), ('J5','机车厂','唐山'), ('J6','无线电厂','常州'), ('J7','半导体厂','南京');
```

```
insert into SPJ(Sno,Pno,Jno,QTY) values ('S1','P1','J1',200), ('S1','P1','J3',100), ('S1','P1','J4',700),  
('S1','P2','J2',100), ('S2','P3','J1',400), ('S2','P3','J2',200), ('S2','P3','J4',500), ('S2','P3','J5',400),  
('S2','P5','J1',400), ('S2','P5','J2',100), ('S3','P1','J1',200), ('S3','P3','J1',200), ('S4','P5','J1',100),  
('S4','P6','J3',300), ('S4','P6','J4',200), ('S5','P2','J4',100), ('S5','P3','J1',200), ('S5','P6','J2',200),  
('S5','P6','J4',500);J(C:\Users\chenn\AppData\Roaming\Typora\typora-user-  
images\1571221773004.png)
```

第一个实验

\l 查看现有数据库

\c spj 进入数据库

命令行连接:

```
psql -h localhost -p 5432 -U postgres spj
```

\d 列出当前数据库的所有表格。 \d [table_name] 列出某一张表格的结构。

表格概况

S: 供应商

P: 零件表

J: 工程项目

```
spj=# \d p
           数据表 "public.p"
   栏位   | 类型      | 校对规则 | 可空的 | 预设
-----+-----+-----+-----+-----
 pno      | character(2) |          | not null |
 pname    | character(6) |          |          |
 color    | character(2) |          |          |
 weight   | integer      |          |          |
索引:
 "p_pkey" PRIMARY KEY, btree (pno)
由引用:
 TABLE "spj" CONSTRAINT "spj_pno_fkey" FOREIGN KEY (pno) REFERENCES p(pno)
```

```
spj=# select * from s;
 sno |  sname  | status | city
-----+-----+-----+-----
  S1 | 精益   |    20  | 天津
  S2 | 盛锡   |    10  | 北京
  S3 | 东方红 |    30  | 北京
  S4 | 丰泰盛 |    20  | 天津
  S5 | 为民   |    30  | 上海
(5 行记录)
```

```
spj=# select * from p;
 pno |  pname  | color | weight
-----+-----+-----+-----
  P1 | 螺母   | 红    |    12
  P2 | 螺栓   | 绿    |    17
  P3 | 螺丝刀 | 蓝    |    14
  P4 | 螺丝刀 | 红    |    14
  P5 | 凸轮   | 蓝    |    40
  P6 | 齿轮   | 红    |    30
(6 行记录)
```

```
spj=# select * from j;
```

jno	jname	city
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

(7 行记录)

```
spj=# select * from spj;
```

sno	pno	jno	qty
S1	P1	J1	200
S1	P1	J3	100
S1	P1	J4	700
S1	P2	J2	100
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J4	500
S2	P3	J5	400
S2	P5	J1	400
S2	P5	J2	100
S3	P1	J1	200
S3	P3	J1	200
S4	P5	J1	100
S4	P6	J3	300
S4	P6	J4	200
S5	P2	J4	100
S5	P3	J1	200
S5	P6	J2	200
S5	P6	J4	500

(19 行记录)

(1) 找出所有供应商的姓名和所在城市

```
spj=# select sname,city from s;
```

sname	city
精益	天津
盛锡	北京
东方红	北京
丰泰盛	天津
为民	上海

(5 行记录)

(2) 找出所有零件的名称、颜色、重量

```
spj=# select pname,color,weight from p;
```

pname	color	weight
螺母	红	12
螺栓	绿	17
螺丝刀	蓝	14
螺丝刀	红	14
凸轮	蓝	40
齿轮	红	30

(6 行记录)

(3)找出使用供应商S1所供应零件的工程号码。

```
spj=# select status from s where sno='S1';
```

status
20

(1 行记录)

```
spj=# select status from s x where x.sno='S1';
```

status
20

(1 行记录)

(4)找出工程项目J2使用的各种零件的名称及其数量。

```
spj=# select pname,qty from spj,p,j where spj.pno=p.pno and spj.jno=j.jno and j.jno='J2';
```

pname	qty
螺栓	100
螺丝刀	200
凸轮	100
齿轮	200

(4 行记录)

(5)找出上海厂商供应的所有零件号码。

内连接

内连接 (INNER JOIN) 根据连接谓词结合两个表 (table1 和 table2) 的列值来创建一个新的结果表。查询会把 table1 中的每一行与 table2 中的每一行进行比较, 找到所有满足连接谓词的行的匹配对。

当满足连接谓词时, A 和 B 行的每个匹配对的列值会合并成一个结果行。

内连接 (INNER JOIN) 是最常见的连接类型, 是默认的连接类型。

INNER 关键字是可选的。

下面是内连接 (INNER JOIN) 的语法:

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.common_field = table2.common_field;
```

基于上面的表，我们可以写一个内连接，如下所示：

```
runoobdb=# SELECT EMP_ID, NAME, DEPT FROM COMPANY INNER JOIN DEPARTMENT ON  
COMPANY.ID = DEPARTMENT.EMP_ID;
```

emp_id	name	dept
1	Paul	IT Billing
2	Allen	Engineering
7	James	Finance

(3 rows)

```
spj=# select p.pno from spj,s,p where s.city='上海' and spj.pno=p.pno and spj.sno=s.sno;  
pno
```

```
-----  
P2  
P3  
P6  
P6  
(4 行记录)
```

(5)找出上海厂商供应的所有零件号码。

```
SELECT PNO FROM SPJ,S WHERE S.SNO=SPJ.SNO AND CITY='上海'
```

```
spj=# SELECT DISTINCT pno FROM spj INNER JOIN s ON spj.sno=s.sno WHERE city='上海';  
pno
```

```
-----  
P2  
P3  
P6  
(3 行记录)
```

(6)找出使用上海产的零件的工程名称。

```
spj=# select jname from spj,j,s where s.city='上海' and spj.jno=j.jno and spj.sno=s.sno;  
jname
```

```
-----  
造船厂  
三建  
一汽  
造船厂  
(4 行记录)
```

```
spj=# select distinct jname from spj,j,s where s.city='上海' and spj.jno=j.jno and spj.sno=s.sno;  
jname
```

```
-----  
三建  
一汽  
造船厂  
(3 行记录)
```

※(7)找出没有使用天津产的零件的工程号码。

```
spj=# select j.jno,s.city from j,spj,s where s.city!='天津' and spj.jno=j.jno and spj.sno=s.sno;
 jno      city
-----
 J1      北京
 J2      北京
 J4      北京
 J5      北京
 J1      北京
 J2      北京
 J1      北京
 J1      北京
 J4      上海
 J1      上海
 J2      上海
 J4      上海
(12 行记录)

spj=# select distinct j.jno from j,spj,s where s.city!='天津' and spj.jno=j.jno and spj.sno=s.sno;
 jno
-----
 J1
 J2
 J4
 J5
(4 行记录)
```

(having 再把里面有天津产的去掉了)

```
spj=# select j.jno,s.city from j,spj,s where s.city!='天津' and spj.jno=j.jno and spj.sno=s.sno and j.jno not in (select
 j.jno from j,spj,s where spj.jno=j.jno and spj.sno=s.sno and s.city='天津');
 jno      city
-----
 J5      北京
(1 行记录)
```

注意: SELECT DISP JNO FROM SPJ WHERE JNO NOT IN (SELECT DIST JNO FROM SPJ,S WHERE S.SNO=SPJ.SNO AND S.CITY='天津') 适用于JNO是唯一或不唯一的情况.

注意: SELECT DIST JNO FROM SPJ,S WHERE S.SNO=SPJ.SNO AND S.CITY<>'天津'适用于JNO是唯一的情况

(8)把全部红色零件的颜色改成蓝色。

```
spj=# update p set color='蓝' where color='红';
UPDATE 3
spj=#
```

(9)由S5供给J4的零件P6改为由S3供应。

```
spj=# update spj set sno='S3' where sno='S5' and pno='P6' and jno='J4';
UPDATE 1
```

(10)从供应商关系中删除供应商号是S2的记录，并从供应情况关系中删除相应的记录。

```
spj=# delete from spj where sno='S2';
DELETE 6
spj=# delete from s where sno='S2';
DELETE 1
```

(11) (11) 请将 (S2,J6,P4,200) 插入供应情况关系。

```
spj=# insert into spj values('S2','J6','P4','200');
错误: 插入或更新表 "spj" 违反外键约束 "spj_sno_fkey"
描述: 键值对 (sno)=(S2) 没有在表 "s" 中出现.
spj=#
```

第二个实验

\l 查看现有数据库

\i G:/database/Database-master/wk2spj.sql

\c wk2spj 进入没修改过的spj（因为第一个实验修改过一次）

表格概况

```
wk2spj=# select * from s
wk2spj=# ;
```

sno	sname	status	city
S1	精益	20	天津
S2	盛锡	10	北京
S3	东方红	30	北京
S4	丰泰盛	20	天津
S5	为民	30	上海

(5 行记录)

```
wk2spj=# select * from p;
```

pno	pname	color	weight
P1	螺母	红	12
P2	螺栓	绿	17
P3	螺丝刀	蓝	14
P4	螺丝刀	红	14
P5	凸轮	蓝	40
P6	齿轮	红	30

(6 行记录)

```
wk2spj=# select * from j;
```

jno	jname	city
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

(7 行记录)

```

wk2spj=# select * from spj;
 sno | pno | jno | qty
-----+-----+-----+-----
 S1  | P1  | J1  | 200
 S1  | P1  | J3  | 100
 S1  | P1  | J4  | 700
 S1  | P2  | J2  | 100
 S2  | P3  | J1  | 400
 S2  | P3  | J2  | 200
 S2  | P3  | J4  | 500
 S2  | P3  | J5  | 400
 S2  | P5  | J1  | 400
 S2  | P5  | J2  | 100
 S3  | P1  | J1  | 200
 S3  | P3  | J1  | 200
 S4  | P5  | J1  | 100
 S4  | P6  | J3  | 300
 S4  | P6  | J4  | 200
 S5  | P2  | J4  | 100
 S5  | P3  | J1  | 200
 S5  | P6  | J2  | 200
 S5  | P6  | J4  | 500
(19 行记录)

```

(C:\Users\chenn\AppData\Roaming\Typora\typora-user-images\1571277800968.png)

掌握SELECT/SELECT DISTINCT/WHERE/AND/OR/ORDER BY/INSERT INTO/UPDATE/DELETE等操作

5、查询操作

1) 在零件表的视图找出weight < 20 的零件名字(PNAME)

```

wk2spj=# select pname from p where weight<20;
      pname
-----
 螺母
 螺栓
 螺丝刀
 螺丝刀
(4 行记录)

```

2) 查询供应商表中城市为北京的供应商姓名(SNAME)

```

wk2spj=# select sname from s where city='北京';
      sname
-----
 盛锡
 东方红
(2 行记录)

```

※3) 在零件表中查询平均重量在15以上的零件名字和零件代码(PNO)


```

wk2spj=# select pname,pno from p A where (select avg(B.weight) from p B where B.pname=A.pname)>15;
  pname | pno
-----+----
  螺栓   | P2
  凸轮   | P5
  齿轮   | P6
(3 行记录)

```

```

wk2spj=# select pname,pno from p group by(pname,pno) having avg(weight)>15;
  pname | pno
-----+----
  凸轮   | P5
  螺栓   | P2
  齿轮   | P6
(3 行记录)

```

having测试

```

wk2spj=# select pname,avg(weight) from p group by pname having avg(weight)>15;
  pname | avg
-----+-----
  螺栓   | 17.0000000000000000
  凸轮   | 40.0000000000000000
  齿轮   | 30.0000000000000000
(3 行记录)

```

```

wk2spj=# select pname,avg(weight) from p group by pname having avg(weight)<15;
  pname | avg
-----+-----
  螺丝刀 | 14.0000000000000000
  螺母   | 12.0000000000000000
(2 行记录)

```

但若加pno就聚集不起来

```

wk2spj=# select pname,pno from p group by pname having avg(weight)>15;
错误:  字段 "p.pno" 必须出现在 GROUP BY 子句中或者在聚合函数中使用
第1行select pname,pno from p group by pname having avg(weight)>15...

```

因为数量不一致

所以如果用这样的 其实结果是不对的

```

wk2spj=# select pname,pno,avg(weight) from p group by (pname,pno) having avg(weight)<15;
  pname | pno | avg
-----+----+-----
  螺母   | P1  | 12.0000000000000000
  螺丝刀 | P4  | 14.0000000000000000
  螺丝刀 | P3  | 14.0000000000000000
(3 行记录)

```

这样的实质由于pno都不一致导致并没有聚集起来

所以如果只要pname 可以这么写

```

wk2spj=# select pname,avg(weight) from p group by pname having avg(weight)<15;
  pname | avg
-----+-----
  螺丝刀 | 14.0000000000000000
  螺母   | 12.0000000000000000
(2 行记录)

```

```
wk2spj=# select pname,avg(weight) from p group by pname having avg(weight)>15;
```

pname	avg
螺栓	17.0000000000000000
凸轮	40.0000000000000000
齿轮	30.0000000000000000

(3 行记录)

由

```
wk2spj=# select pname,pno from p group by(pname,pno) having avg(weight)<15;
```

pname	pno
螺母	P1
螺丝刀	P4
螺丝刀	P3

(3 行记录)

```
wk2spj=# select pname,pno from p A where (select avg(B.weight) from p B where B.pname=A.pname)<15;
```

pname	pno
螺母	P1
螺丝刀	P3
螺丝刀	P4

(3 行记录)

觉得这方法一定有问题 将数据集的一个14改成17

```
wk2spj=# update p set weight=17 where pno='P4';
UPDATE 1
wk2spj=# select * from p;
```

pno	pname	color	weight
P1	螺母	红	12
P2	螺栓	绿	17
P3	螺丝刀	蓝	14
P5	凸轮	蓝	40
P6	齿轮	红	30
P4	螺丝刀	红	17

(6 行记录)

然乎按理两个螺丝刀都不被算进去

```
wk2spj=# select pname,pno from p group by(pname,pno) having avg(weight)<15;
```

pname	pno
螺母	P1
螺丝刀	P3

(2 行记录)

```
wk2spj=# select pname,pno from p A where (select avg(B.weight) from p B where B.pname=A.pname)<15;
```

pname	pno
螺母	P1

(1 行记录)

可以看到虽然用group by 但是那个方法并没有聚集

而用相关子查询才ok 所以这题必须用相关子查询（其实也是聚集并且更鲁棒）

```
wk2spj=# select pname,pno from p A where (select avg(B.weight) from p B where B.pname=A.pname)>15;
```

pname	pno
螺栓	P2
螺丝刀	P3
凸轮	P5
齿轮	P6
螺丝刀	P4

(5 行记录)

就是指先子查询先筛一遍结果再给父查询

4) 查询全体供应商的姓名 (SNAME) 和状态(STATUS)

```
spj=# SELECT sname, status FROM s;
```

sname	status
精益	20
东方红	30
丰泰盛	20
为民	30

(4 行记录)

5) 查询所有weight在13到20岁 (含13和20) 的零件代码 (PNO)、零件名 (PNAME) 和颜色(COLOR)

```
spj=# SELECT pno, pname, color FROM p WHERE weight between 13 and 20;
```

pno	pname	color
P2	螺栓	绿
P3	螺丝刀	蓝
P4	螺丝刀	蓝

(3 行记录)

6) 查询所有“螺”开头的零件代码 (PNO) 和零件名 (PNAME)

```
spj=# SELECT pno, pname FROM p WHERE pname like '螺%';
```

pno	pname
P2	螺栓
P3	螺丝刀
P1	螺母
P4	螺丝刀

(4 行记录)

7) 查询所有零件的平均重量

```
wk2spj=# select pname, avg(weight) from p group by pname;
```

pname	avg
螺丝刀	15.5000000000000000
螺栓	17.0000000000000000
螺母	12.0000000000000000
凸轮	40.0000000000000000
齿轮	30.0000000000000000

(5 行记录)

8) 查询同在“天津”的工程项目名 (JNAME)

```
spj=# SELECT pno, pname FROM p WHERE pname like '螺%';
```

pno	pname
P2	螺栓
P3	螺丝刀
P1	螺母
P4	螺丝刀

(4 行记录)

9) 查询在“精益”供应商下的零件，且质量小于15的零件详细信息

```
wk2spj=# select pno,pname,color,weight from p A where(select avg(B.weight) from p B where B.pname=A.pname)<15;
```

pno	pname	color	weight
P1	螺母	红	12

(1 行记录)

```
wk2spj=# select A.pno,A.pname,A.color,A.weight from p A,spj,s where A.pno=spj.pno and spj.sno=s.sno and s.sname='天津' and (select avg(B.weight) from p B where B.pname=A.pname)<15;
```

pno	pname	color	weight
-----	-------	-------	--------

(0 行记录)

↑如果平均质量就这样

↓标准

```
spj=# SELECT DISTINCT pname, p.pno, weight, color
spj-# FROM s,p,spj
spj-# WHERE s.sno = spj.sno
spj-# AND p.pno= spj.pno
spj-# AND s.sname = '精益',
spj-# AND p.weight < 15;
```

pname	pno	weight	color
螺母	P1	12	蓝

(1 行记录)

订正:

1) 在零件表的视图中找出weight < 20 的零件名字(PNAME)

```
postgres=# \c wk2spj
您现在已经连接到数据库 "wk2spj",用户 "postgres".
wk2spj=# create view pview as select * from p;
CREATE VIEW
wk2spj=# select * from pview;
```

pno	pname	color	weight
P1	螺母	红	12
P2	螺栓	绿	17
P3	螺丝刀	蓝	14
P5	凸轮	蓝	40
P6	齿轮	红	30
P4	螺丝刀	红	17

(6 行记录)

实验三

练习带连接的子查询操作

\c wk2spj

6、复杂子查询操作

表格概况

```
wk2spj=# select * from s
```

```
wk2spj=# ;
```

sno	sname	status	city
S1	精益	20	天津
S2	盛锡	10	北京
S3	东方红	30	北京
S4	丰泰盛	20	天津
S5	为民	30	上海

(5 行记录)

```
wk2spj=# select * from p;
```

pno	pname	color	weight
P1	螺母	红	12
P2	螺栓	绿	17
P3	螺丝刀	蓝	14
P4	螺丝刀	红	14
P5	凸轮	蓝	40
P6	齿轮	红	30

(6 行记录)

```
wk2spj=# select * from j;
```

jno	jname	city
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

(7 行记录)

```
wk2spj=# select * from spj;
 sno | pno | jno | qty
-----+-----+-----+-----
 S1  | P1  | J1  | 200
 S1  | P1  | J3  | 100
 S1  | P1  | J4  | 700
 S1  | P2  | J2  | 100
 S2  | P3  | J1  | 400
 S2  | P3  | J2  | 200
 S2  | P3  | J4  | 500
 S2  | P3  | J5  | 400
 S2  | P5  | J1  | 400
 S2  | P5  | J2  | 100
 S3  | P1  | J1  | 200
 S3  | P3  | J1  | 200
 S4  | P5  | J1  | 100
 S4  | P6  | J3  | 300
 S4  | P6  | J4  | 200
 S5  | P2  | J4  | 100
 S5  | P3  | J1  | 200
 S5  | P6  | J2  | 200
 S5  | P6  | J4  | 500
(19 行记录)
```

(C:\Users\chenn\AppData\Roaming\Typora\typora-user-images\1571277800968.png)

1) 在零件表中找出weight排名前三的零件名字(PNAME)，按降序输出

```
wk2spj=# select * from p order by weight DESC;
 pno | pname | color | weight
-----+-----+-----+-----
 P5  | 凸轮 | 蓝    | 40
 P6  | 齿轮 | 红    | 30
 P2  | 螺栓 | 绿    | 17
 P4  | 螺丝刀 | 红    | 17
 P3  | 螺丝刀 | 蓝    | 14
 P1  | 螺母 | 红    | 12
(6 行记录)
```

(ASC DESC)

```
wk2spj=# select * from p order by weight DESC LIMIT 3 OFFSET 0;
 pno | pname | color | weight
-----+-----+-----+-----
 P5  | 凸轮 | 蓝    | 40
 P6  | 齿轮 | 红    | 30
 P2  | 螺栓 | 绿    | 17
(3 行记录)
```

※2) 查询至少使用了供应商S1所供应的全部零件的城是(CITY)

错误理解与做法

```

wk2spj=# select j.city from spj,s,j where spj.sno=s.sno and spj.jno=j.jno and s.sno='S1';
city
-----
北京
长春
天津
天津
(4 行记录)

```

只用了S1所提供的全部零件（错）

需要S1产的pnoP1和P2没有被别的零件商产过

```

wk2spj=# select j.city from spj,j where (spj.jno=j.jno and spj.sno='S1' and spj.pno not in (select DISTINCT pno from spj
where sno!='S1'))
wk2spj=# ;
city
-----
(0 行记录)

```

(这份思路是找S1有的所有Pno即P1和P2并不在别的S非1的有的Pno（P1到P6）里面 也就是下面这种 但是这样理解题意是错的)

```

wk2spj=# select j.city from spj,j where spj.jno=j.jno and EXISTS
wk2spj=# (
wk2spj=# SELECT pno from spj where sno='S1'
wk2spj=# EXCEPT
wk2spj=# SELECT pno from spj where sno<>'S1'
wk2spj=# );
city
-----
(0 行记录)

```

正确的用相关子查询的做法

理解方式：

2) 查询至少使用了~~供应商S1所供应的全部零件~~的城是(CITY)
不可单指P1和P2

S1所供应的全部零件 这不能单指

sno	pno	jno	qty
S1	P1	J1	200
S1	P1	J3	100
S1	P1	J4	700
S1	P2	J2	100

这边P1和P2 而指 S1产的P1和S1产的P2

然后这里的J1,J2,J3,J4必须同时造过S1产的P1和S1产的P2，并且一个城市比如天津

```

wk2spj=# select * from j;

```

jno	jname	city
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

(7 行记录)

的J3和J4算一个，如果J3和J4分别产过P1和P2也要算进这个城市，当然事实还是不存在

子问题1 在子表中操作

修改视图会导致原表一起修改!!

```

wk2spj=# create view zibiao as (select pno,jno from spj where sno='S1')
wk2spj=# ;
CREATE VIEW

```

```

wk2spj=# select * from zibiao
wk2spj=# ;

```

pno	jno
P1	J1
P1	J3
P1	J4
P2	J2

(4 行记录)


```

wk2spj=# update zibiao set pno='P2' where jno='J4';
UPDATE 1
wk2spj=# update zibiao set jno='J1' where jno='J2';
UPDATE 1
wk2spj=# zibiao
wk2spj=# ;
错误: 语法错误 在 "zibiao" 或附近的
第1行zibiao

```

```

wk2spj=# select * from zibiao;

```

pno	jno
P1	J1
P1	J3
P2	J4
P2	J1

(4 行记录)

这样按理要输出1

法1: 假设知道P1和P2

```

wk2spj=# select jno from zibiao z1 where EXISTS(select * from zibiao z2 where z2.pno='P1' and z2.jno=z1.jno) and EXISTS
select * from zibiao z2 where z2.pno='P2' and z2.jno=z1.jno);
 jno
----
 J1
 J1
(2 行记录)

```

这方法不是很好 得自己遍历pno

法2: (select count(*) from (select distinct pno from zibiao) z)为pno中不同的项的个数 (即pno总数) 比如这边是P1和P2两个

(select count(distinct z2.*) from zibiao z2 where z2.jno=z1.jno group by z2.jno)聚集jno去找jno对应pno的个数 若为总数则匹配上

```

wk2spj=# select count(pno) from zibiao;
 count
-----
      4
(1 行记录)

```

备注:

```

wk2spj=# select count(*) from zibiao;
 count
-----
      4
(1 行记录)

```

(count distinct *是找所有不相同的记录)

```

wk2spj=# select z1.jno from zibiao z1 where (select count(distinct z2.*) from zibiao z2 where z2.jno=z1.jno group
jno)=(select count(*) from (select distinct pno from zibiao) z);
 jno
----
 J1
 J1
(2 行记录)

```

↑这张表是真聚集!!!!

的确最后输出了J1

子问题2

将

```
wk2spj=# select * from j;
```

jno	jname	city
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

(7 行记录)

以city合并jno

理论上应该生成天津（对应J3J4）和北京（J1）

1

```
wk2spj=# select distinct zibiao.pno from j,zibiao where city='天津' and j.jno=zibiao.jno;
pno
-----
P1
P2
(2 行记录)
```

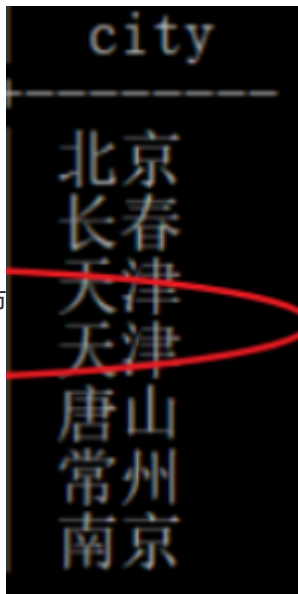
天津对应的JNO对应的每个pno

也就是

```
wk2spj=# select zibiao.pno from j,zibiao where j.jno in (select jno from j where city='天津') and zibiao.jno=j.jno;
pno
-----
P1
P2
(2 行记录)
```

2 让城市一个个像for循环一样输出出来 用IN （相关子查询的灵魂）

```
select distinct jc.city from j jc where jc.city in (select distinct city from j)
```



把后面的某个城市

比如天津 换成jc.city 这就是相关子查询的灵魂！！

最终答案

逻辑： 第一层 让city排队一个个丢给where的某个元素

第二层 对这个元素的所有jno找出所有对应的pno（去重） 并计算这个东西的个数

第二层还有一个写法是（这样是会用distinct）而按下图中第二层那样用in就不会

```
wk2spj=# select distinct zibiao.pno from j,zibiao where city='天津' and j.jno=zibiao.jno;
pno
-----
P1
P2
(2 行记录)
```

第三层 对zibiao来说有几个pno(去重) 若对应的某city所有jno覆盖到了这些pno 则的二层与第三层相等 这样的city会被输出

```
wk2spj=# select jc.city from j jc where jc.city in (select distinct city from j) and
wk2spj=# (select count(*) from (select zibiao.pno from j j1,zibiao where j1.jno in (select j2.jno from j j2 where city=j
c.city) and zibiao.jno=j1.jno) z)=
wk2spj=# (select count(*) from (select distinct pno from zibiao) z2);
city
-----
北京
天津
天津
(3 行记录)
```

```
wk2spj=# select jc.city from j jc where jc.city in (select distinct city from j) and (select count(*) from (select zibia
o.pno from j j1,zibiao where j1.jno in (select j2.jno from j j2 where city=jc.city) and zibiao.jno=j1.jno) z)=(select co
unt(*) from (select distinct pno from zibiao) z2);
city
-----
北京
天津
天津
(3 行记录)
```

```
wk2spj=# select distinct jc.city from j jc where jc.city in (select distinct city from j) and (select count(*) from (sel
ect zibiao.pno from j j1,zibiao where j1.jno in (select j2.jno from j j2 where city=jc.city) and zibiao.jno=j1.jno) z)=(
select count(*) from (select distinct pno from zibiao) z2);
city
-----
北京
天津
(2 行记录)
```

相关子查询灵魂

例如，以下查询会为每个客户返回其订单ID最大的订单。

SQL查询代码如下：



```
-- 相关子查询
SELECT custid,orderid,orderdate,empid
FROM Sales.Orders AS orders1
WHERE orderid=
    (SELECT MAX(orders2.orderid)
     FROM Sales.Orders AS orders2
     WHERE orders2.custid= orders1.custid);-- 这里引用了外部查询的custid字段
```

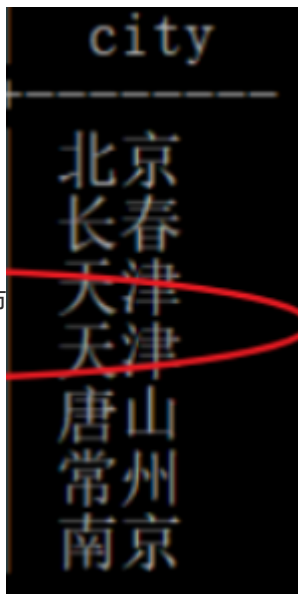
EXISTS 作为筛选操作好用！（元素个数是否等于0）

```
wk2spj=# select j.city from spj,j where spj.jno=j.jno and EXISTS
wk2spj=# (
wk2spj=# SELECT pno from spj where sno='S1'
wk2spj=# EXCEPT
wk2spj=# SELECT pno from spj where sno<>'S1'
wk2spj=# );
city
-----
(0 行记录)
```

但是真正的灵魂。。。

```
select distinct jc.city from j jc where jc.city in (select distinct city from j)
```

把后面的某个城市



比如天津 换成jc.city 这就是相关子查询的灵魂！！

第二种理解或者使用方式：

```
SELECT Sno,Cno
FROM Sc x
WHERE Grade>=(SELECT AVG(Grade)
FROM SC y
WHERE y.Sno=x.Sno)
```

把x.Sno即原来的Sno结果在where里遍历一遍 或者说y.Sno=x.Sno就是y.Sno in x.Sno 把结果一股脑给x.Sno 然后就类似having y.Sno in Sno了 就能AVG聚集了

3) 查询出供应商代码（SNO）为S1的，生产零件的全部颜色（COLOR）

```

wk2spj=# select color from spj,p where spj.pno=p.pno and spj.sno='S1';
color
-----
红
红
红
绿
(4 行记录)

```

4) 查询所有WEIGHT > 20的零件名字(PNAME),零件代码(PNO),供应商代码(SNO), 供应商姓名(SNAME)

```

wk2spj=# select DISTINCT s.sno, s.sname, p.pno, p.pname from s, p, spj where s.sno=spj.sno and p.pno=spj.pno and p.weight>20;

```

sno	sname	pno	pname
S2	盛锡	P5	凸轮
S4	丰泰盛	P5	凸轮
S5	为民	P6	齿轮
S4	丰泰盛	P6	齿轮

(4 行记录)

5) 查询供应工程J1零件为红色的供应商号码(SNO)

```

wk2spj=# select sno from p, spj where p.pno=spj.pno and p.color='红';
sno
-----
S1
S1
S1
S3
S4
S4
S5
S5
(8 行记录)

```