

SQL语言 (1)

- SQL语言的基本特征
- 样板模式与数据库
- SQL数据定义功能
- 索引的建立和删除
- SQL的查询功能

Structured Query Language



SQL标准的发展

- 70年代初 Codd提出了关系演算语言ALPHA，但一直未实现。
- 美国IBM公司研制了一个面向域的数据查询语言QBE (Query By Example)
- 美国加利福尼亚大学研制的面向元组的数据语言QUEL
- SQL (Structured Query Language) 是介于关系代数和关系演算之间的语言，Boyce和Chamberlin于1974年提出的SQUARE基础上发展而成的，目前已成功在大多数DBMS中实现。

SQL标准的发展

- 1986年10月美国国家标准局（ANSI）批准了SQL作为关系数据库语言的美国标准。
- 自SQL成为国际标准语言以后，各个数据库厂家纷纷推出各自的SQL软件或与SQL的接口软件。这就使大多数数据库均用SQL作为共同的数据存取语言 and 标准接口，使不同数据库系统之间的互操作有了共同的基础。
- SQL成为国际标准，对数据库以外的领域也产生了很大影响，有不少软件产品将SQL语言的查询功能与图形功能、软件工程工具、软件开发工具、人工智能程序结合起来。

SQL标准的基本概念

➤ SQL集数据定义、数据查询、数据更新和数据控制于一体，既可作为独立语言由终端用户以联机交互方式使用，也可作为数据子语言嵌入主语言中使用。SQL已经成为目前最主要的数据库操纵和控制语言，比较流程的程序设计工具基本上都支持嵌入式SQL语言。



SQL语言的基本特征

- 一体化的特点
- 两种使用方式，统一的语法格式
- 高度非过程化
- 语言简洁、易学易用
- SQL语言也支持关系数据库三级模式体系结构
 - 外模式：视图+一些基本表
 - 模式(概念模式)：基本表
 - 内模式：存储文件(逻辑结构)



SQL 语言主要动词含义

- | SQL功能 | 动词 |
|------------|----------------------|
| 数据查询(表) | SELECT |
| 数据定义(表) | CREATE,DROP,ALTER |
| 数据操作(表中内容) | INSERT,UPDATE,DELETE |
| 数据控制(授权收回) | GRANT,REVOKE |



样板模式

- ❑ Sailors(sid: integer, sname: string, rating: integer, age: integer)
- ❑ Boats(bid: integer, bname: string, color: string)
- ❑ Reserves(sid: integer, bid: integer, day: date)



样板数据库

Boats

Bid	bname	color
101.	Interlake	blue
102.	Interlake	red
103.	Clipper	green
104.	Marine	red

Sailors

Sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	40
85	Art	3	25
95	Bob	3	63

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

SQL数据定义功能

□ 创建表(模式)、索引(内模式)和视图(外模式)

一般格式如下:

```
CREATE TABLE <表名> (<列名> <数据类型> [列级完整性约束条件] [, <列名> <数据类型> [列级完整性约束条件]].....[, <表级完整性约束条件>] ;
```

() 必须有的内容, [] 可有可无, 自定义的内容



创建表举例

□ 创建表Sailors

Create table Sailors

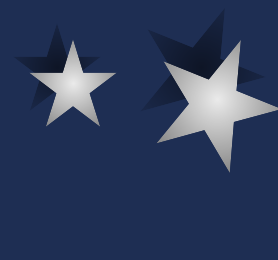
```
( sid int,  
  sname char(20),  
  rating int,  
  age int,  
  primary key ( sid ) );
```

表级约束

Create table Sailors

```
( sid int not null,  
  sname char(20),  
  rating int,  
  age int) ;
```

列级约束



SQL数据定义功能

□ 创建表Boats

Create table Boats

(bid int,
 bname char(20),
 color char(10),
 primary key (bid));

表级约束



SQL数据定义功能

□ 创建表Reverses

Create table Reverses

```
( sid int,  
  bid char(10),  
  day datetime,  
  primary key ( sid, bid ),  
  foreign key(sid) references Sailors,  
  foreign key(bid) references Boats );
```

表级约束

□ 数据类型 (p89)



基本表的修改

□ 修改

ALTER TABLE <表名>

[ADD <新列名> <数据类型>[完整性约束]];

[DROP <完整性约束>;

[MODEFY <列名> <数据类型>;

以上三者是or的关系

例： Alter table Sailors
add sex char(3);

增加在最后一列



删除

□ 删除

一般格式：

DROP TABLE <表名>

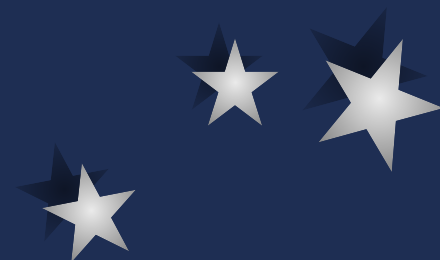
例：Drop table Sailors;

注：删除是把一个基本表的定义，连同表上的所有记录、索引以及由此基本表导出的所有视图都被删除，并释放相应的存储空间。



索引

- ❑ 索引可以加快查找
- ❑ 索引的代价（时间、空间）
- ❑ 索引的维护和使用（系统）
- ❑ 索引建立策略
 - ✱ 如果一个（组）属性经常在查询条件中出现
 - ✱ 如果一个属性经常作为min或max等集函数的参数
 - ✱ 在连接属性上



索引

❑ 唯一索引：unique

每个索引值只对应一个唯一的数据记录

❑ 聚簇索引：cluster

索引项的顺序和表中数据记录的物理顺序一致

学号	学生
95001	95002 王 18 CS
95002	95004 李 17 IS
95003	95001 张 17 MA
95004	95003 赵 18 CS



学号	学生
95001	95001 张 17 MA
95002	95002 王 18 CS
95003	95003 赵 18 CS
95004	95004 李 17 IS

索引的建立

- 在表Sailors的属性sname上建立一个索引

```
create index xname on Sailors( sname );
```

- 在表Boats的属性bname上建立一个索引

```
create index xname on Boats( bname );
```

- 在表Reverse的属性(bid,sid)上建立一个索引(组合索引)

```
create index xbid sid on  
Reverse( bid asc, sid desc );
```

- asc为升序,desc为降序,缺省为升序



索引的建立

一般格式：

CREATE [UNIQUE] [CLUSTER] INDEX

<索引名> ON <表名> (<列名> [<次序>] [, <列名>
[<次序>]]...);

□ 在表Sailors的属性sid上建立一个**唯一**索引

create **unique** index in_u_sid on Sailors(sid);

□ 在表Boats的属性bid上建立一个**聚簇**索引

create **cluster** index in_c_bid on Sailors(sid);



索引的删除

- 索引的删除: Drop index <索引名>

Drop index xsname;

- 索引可以**加快**查找，但是不是索引越多越好？



SQL的查询功能

□ SQL查询语句的基本结构

SELECT [ALL|DISTINCT] 目标列

FROM 基本表 (视图) 范围变量名,....

[WHERE 条件表达式]

[Group by 列名1 [having 分组表达式]]

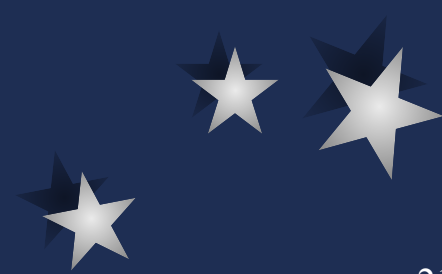
[Order By 列名2 asc | desc];

查询条件

结果输出条件

样板模式

- ❑ Students (Sno, Sname, Ssex, Sage, Sdept)
- ❑ Courses (Cno, Cname, Cpno, Credit)
- ❑ SC (Sno, Cno, Grade)
- ❑ 用SQL创建以上三个表



单表查询

- ❑ 查询内容仅涉及一个表

- ❑ 例子：查询全体学生的学号和姓名

```
select Sno, Sname  
from Students;
```

顺序无关性

- ❑ 例子：查询全体学生的基本情况

```
select *  
from Students;
```

- ❑ 例子：查询全体学生的姓名及出生年份

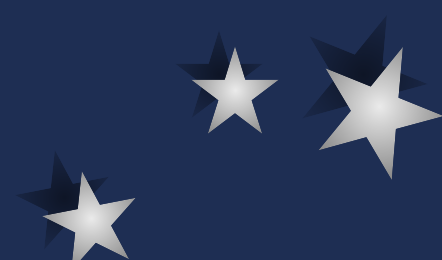
```
select Sname, 2001-Sage  
from Students;
```

单表查询

- 例子：查询全体学生的姓名、出生年份和所在系，要求用小写字母表示所有系名

```
select Sname, 'Year of birth:', 2001-Sage,  
       islower(Sdept)  
from Students;
```

Sname	'Year of birth: '	2001-Sage	islower(Sdept)
李勇	Year of birth:	1976	cs
刘辰	Year of birth:	1977	is
王敏	Year of birth:	1978	ma
张立	Year of birth:	1978	is



单表查询

中间空格

- 用户可以通过指定别名来改变查询结果的列标题

```
select Sname NAME, 'Year of birth' BIRTH,  
       2001-Sage BIRTHDAY,  
       islower(Sdept) DEPARTMENT  
from   Students;
```

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of birth:	1976	cs
刘辰	Year of birth:	1977	is
王敏	Year of birth:	1978	ma
张立	Year of birth:	1978	is



单表查询

❑ 消除取值重复的行

- 查询选修了课程的学生学号

```
select Sno  
from SC;
```

- 消除重复行

```
select distinct Sno  
from SC;
```

Sno

95001

95001

95001

95002

95002

Sno

95001

95002



单表查询

出现在where后面的运算符或者SQL定义的保留字

□ 查询满足条件的元组

常用查询条件

查询条件

谓词

比较

=, >, <, >=, <=, !=, <>, !>, !<

not + 上述比较运算符

确定范围

between... and..., not between... and...

确定集合

in, not in

字符匹配

like, not like

空值

is null, is not null

多重条件

and, or



单表查询——一般查询

□ 一般查询

- 查询年龄在20岁以下的学生姓名及其年龄

```
select Sname, Sage  
from Sduents  
where Sage < 20;
```

- 查询年龄在20~23岁之间的学生姓名、所在

系, 年龄

```
select Sname, Sdept, Sage  
from Students  
where Sage between 20 and 23;
```



单表查询 – ‘in’, ‘%’查询

- 查询信息系、数学系和计算机系学生的姓名和性别

```
select Sname, Ssex  
from Students  
where Sdept in ('IS', 'MA', 'CS');
```

- 查询所有姓刘的学生的姓名、学号和性别

```
select Sname, Sno, Ssex  
from Students  
where Sname like '刘%';
```



单表查询—‘_’条件

- 查询姓欧阳且全名为三个汉字的学生姓名

```
select Sname
```

```
from Students
```

```
where Sname like '欧阳__';
```

- 查询DB_Design课程号和学分

```
select Cno, Ccredit
```

```
from Course
```

```
where Cname like 'DB__Design';
```

?



单表查询 – ‘_’, ‘is’ 条件

```
select Cno, Ccredit  
from Course  
where Cname like 'DB\_Design' escape '\';
```

➤ 查询缺考学生的学号和课号

```
select Sno, Cno  
from SC  
where Grade is null;
```

这里is 可否用=代替?



单表查询－多条件查询

- 查询计算机系年龄在20岁以下的学生的姓名

```
select Sname
```

```
from Students
```

```
where Sdept='CS' and Sage<20;
```

- 查询信息系、数学系和计算机系学生的姓名和性别

```
select Sname, Ssex
```

```
from Students
```

```
where Sdept='CS' or Sdept='IS' or  
Sdept='MA'
```



单表查询 — 查询结果排序

□ 对查询结果排序

- 例子：查询选修了3号课程的学生学号和成绩，要求查询结果按成绩降序排列

```
select Sno, Grade  
from SC  
where Cno='3'  
order by grade desc;
```



单表查询 — 查询结果排序

- 例子：查询全体学生的情况，查询结果按系号升序排列，同一系的学生按年龄降序排列

```
select *
```

```
from SC
```

```
order by Sdept, Sage desc;
```

- 使用集函数

- 例子：查询学生总数

```
select count(*)
```

```
from Students;
```



单表查询—集函数

- 例子: 查询选修了课程的学生人数

```
select count( distinct Sno)
from SC;
```

- 求选修1号课程的学生们的平均成绩

```
select avg(Grade)
from SC
where Cno='1';
```



单表查询 — 集函数

➤ SQL提供的主要集函数 使用函数对结果进行处理

count([distinct|all] *)

统计元组个数

count([distinct|all] <列名>)

统计一列中值的个数

sum([distinct|all] <列名>)

计算一列值的总和

avg([distinct|all] <列名>)

计算一列值的平均值

max ([distinct|all] <列名>)

求一列值的最大值

min ([distinct|all] <列名>)

求一列值的最小值



单表查询 — 对查询结果分组

- 对查询结果分组: 将查询结果按一列或多列值分组, 目的是将**集函数**作用到组上(例如: 小计)

- 例子: 求每门课的课号及其选课人数

```
select Cno, count(Sno)
from SC
group by Cno;
```

- 例子: 求选课人数超过10人的课程号及其人数

```
select Cno, count(Sno)
from SC
group by Cno
having count(Sno)>10;
```

组内条件

多表连接查询

❑ 等值与非等值连接查询

➤ 查询每个学生及其选修课的情况

```
select Students.*, SC.*  
from Students, SC  
where Students.Sno=SC.sno;
```

Students.Sno	Sname	...	SC.Sno	Cno	...
95001	李勇	...	95001	1	...

...

➤ 自然连接



多表连接查询

```
select Students.*, Cno, Grade
from Students, SC
where Students.Sno=SC.sno;
```

为两个表起别名
C1,C2, 可以先
使用, 后起名

❑ 自身连接 (一个表与自身或不同的两个表之间)

➤ 查询每门课的课号及其间接先修课课号

Courses (Cno, Cname, Cjno, Credit)

Courses (Cno, Cname, Cjno, Credit)

```
select C1.Cno, C2.Cjno
from Course C1, Course C2
where C1.Cjno=C2.Cno;
```

多表连接查询

□ **外连接**: 在一般连接中,只有满足条件的元组才可以作为结果输出, 外连接则不同(左、右外连接)

➤ 查询每个学生及其选修课的情况, 若某个学生没有选课, 则输出其基本情况。

```
select Students.*, Cno, Grade  
from Students, SC
```

```
where Students.Sno=SC.sno(*);
```

万能行

Students.Sno	Sname	...	Cno	...
--------------	-------	-----	-----	-----

95001	李勇	...	1	...
-------	----	-----	---	-----

...

95003	王敏	...
-------	----	-----

多表连接查询

❑ 复合条件连接

- 查询选修2号课程且成绩在90分以上的所有学生

```
select Students.*
```

```
from Students, SC
```

```
where Students.Sno=SC.Sno and  
       Cno='2' and Grade>90;
```

- 例子：查询每个学生的学号、姓名、选修的课程名和成绩

```
select Students.Sno, Sname, Cname, Grade
```

```
from Students, Course, SC
```

```
where Students.Sno=SC.Sno and  
       SC.Cno=Course.Cno;
```


嵌套查询

❑ 嵌套查询

- * 查询块
- * 一个查询块可以放在另一个查询块的where子句或having短语的条件中（例子 例子）
- * 父查询和子查询
- * 子查询中不可以使用order by 子句，order by只能对最终结果排序
- * 多个简单查询构造一个复杂的查询
- * 相关子查询(例子)和不相关子查询(例子)



不相关子查询

□ 带有in的子查询：子查询结果为多值

* 查询与刘晨在一个系学习的学生
select *

from Students

where Sdept in (select Sdept
from Students

where Sname='刘晨');



* 不相关子查询求解方法由里向外



不相关子查询

- * 查询选修了课程名为信息系统的学生学号和姓名

```
select Sno, Sname
```

```
from Students
```

```
where Sno in (select Sno
```

```
from SC
```

```
where Cno in (select Cno
```

```
from Courses
```

```
where Cname='信息系统'));
```

- * 有些嵌套查询可以用连接查询代替，有些不可

不相关子查询

```
select S1.*
```

```
  from Students S1, Students S2
```

```
 where S1.Sdept=S2.Sdept and
```

```
       S2.Sname='刘晨');
```

- 带有比较运算符的子查询：子查询结果为单值

- 例子：查询与刘晨在一个系学习的学生

```
select *
```

```
  from Students
```

```
 where Sdept = (select Sdept
```

```
                from Students
```

```
                where Sname='刘晨');
```

不相关子查询

□ 带有any或all的子查询：必须同比较运算符同用

* θ any θ all

* 查询其他系比信息系某一学生年龄小的学生姓名和年龄

```
select Sname, Sage
from Students
where Sage <any (select Sage
                  from Students
                  where Sdept='IS')
and Sdept!='IS' ;
```



不相关子查询

- ✳ 用集函数实现上述查询(效率比any和all高)

```
select Sname, Sage
```

```
from Students
```

```
where Sage < (select max(Sage)
```

```
from Students
```

```
where Sdept='IS')
```

```
and Sdept!='IS' ;
```

- ✳ 查询其他系比信息系所有学生年龄都小的学生姓名和年龄(思考题)

- ✳ Any和all和集函数的对应关系p₁₁₁

相关子查询

❑ 带有exists的子查询：子查询结果为true或false

✱ 查询所有选修了1号课程的学生姓名

```
select Sname
```

```
from Students
```

```
where exists (select *
```

```
from SC
```

```
where Sno=Students.Sno and
```

```
Cno='1');
```



✱ 相关子查询求解方法由外向里(样板数据库)

✱ 查询没有选修1号课程的学生姓名(思考题)

相关子查询

```
select Sno
  from SC X
 where not exists (select *
                   from SC Y
                   where Sno='95002' and not exists
                     (select *
                      from SC Z
                      where Sno=X.Sno and
                            Cno=Y.Cno));
```



集合查询

- ❑ 多个SQL的查询结果可以进行集和操作
- ❑ 集和操作主要有: union、intersect和minus
- * 例子: 查询计算机系的学生以及年龄不大于19岁

的学生
select *

from Students
where Sdept='CS'

union

select *
from Students
where Sage<=19;

集合查询

- ✱ 例子：查询计算机系的学生与年龄不大于19岁的学生的交集

select *

from Students

where Sdept='CS'

intersect

select *

from Students

where Sage<=19;

集合查询

✱ 例子：查询计算机系的学生与年龄不大于19岁的学生的差集
select *

from Students
where Sdept='CS'

minus

select *
from Students
where Sage<=19;

习题

□ 样板模式

- Sailors(sid: integer, sname: string, rating: integer, age: integer)
- Boats(bid: integer, bname: string, color: string)
- Reserves(sid: integer, bid: integer, day: date)

习题

□ 查询租用过103号船的船员姓名

```
SELECT S.sname
FROM   Sailors S, Reserves R
WHERE  S.sid=R.sid AND bid=103
```

```
SELECT sname
FROM   Sailors, Reserves
WHERE  Sailors.sid=Reserves.sid
      AND bid=103
```

习题

- 查找租用过船只的船员编号

```
SELECT S.sid  
FROM Sailors S, Reserves R  
WHERE S.sid=R.sid
```

- 是否需要在Select子句中添加DISTINCT?
- 如果将Select子句中的S.sid换成S.sname,是否需要添加DISTINCT?



习题

□ 查找rating>7且年龄>25的水手编号

```
Select S.sid  
from   Sailors S  
where  S.rating>7 and S.age>25
```

□ 使用Intersect来代替and运算

```
Select S.sid from Sailors S where s.rating>7  
intersect  
select S2.sid from Sailors S2 where S2.age>25
```

习题

□ 查找租用过红船和绿船的水手名字

□ 写法1

```
Select S.sname  
from Sailors S, Reverses R, Boats B  
where S.sid=R.sid and R.bid=B.bid and  
       B.color="red" and B.color="green"
```



□ 写法2

```
Select S.sname  
from Sailors S, Reverses R1, Boats B1, Reverses R2, Boats B2  
where S.sid=R1.sid and R1.bid=B1.bid and S.sid=R2.sid and  
       R2.bid=B2.bid and  
       B1.color="red" and B2.color="green"
```



❑ 查找租用过红船或绿船的水手编号

```
SELECT S.sid  
FROM Sailors S, Boats B, Reserves R  
WHERE S.sid=R.sid AND R.bid=B.bid  
AND (B.color='red' OR B.color='green')
```

❑ 可以使用Union来代替or运算

```
SELECT S.sid  
FROM Sailors S, Boats B, Reserves R  
WHERE S.sid=R.sid AND  
R.bid=B.bid AND B.color='red'  
UNION  
SELECT S.sid  
FROM Sailors S, Boats B, Reserves R  
WHERE S.sid=R.sid AND  
R.bid=B.bid AND B.color='green')
```

习题（嵌套查询）

- 查找租用过103号船只的水手的名字


```
SELECT S.sname  
FROM Sailors S  
WHERE S.sid IN (SELECT R.sid  
                FROM Reserves R  
                WHERE R.bid=103)
```



嵌套查询

□ 查找租用过103号船只的水手的名字

```
SELECT S.sname  
FROM Sailors S  
WHERE EXISTS (SELECT *  
               FROM Reserves R  
               WHERE R.bid=103 AND S.sid=R.sid)
```



嵌套查询

□ 查找级别比一些叫Harotio高的水手

```
SELECT *  
FROM Sailors S  
WHERE S.rating > ANY (SELECT S2.rating  
                       FROM Sailors S2  
                       WHERE S2.sname='Horatio')
```



使用In查询来改写Intersect查询

□ 查找租用过红船和绿船的水手编号

□ Intersect查询

```
Select S.sid  
from Sailors S, Reverses, R Boats B  
where S.sid=R.bid and R.bid=B.bid and B.color="red"  
intersect  
select S2.sid  
from Sailors S2, Reverses R2, Boats B2  
where S2.sid=R2.bid and R2.bid=B2.bid  
and B2.color="green"
```

使用In查询来改写Intersect查询

□ In查询

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'
      AND S.sid IN (SELECT S2.sid
                     FROM Sailors S2, Boats B2, Reserves R2
                     WHERE S2.sid=R2.sid AND R2.bid=B2.bid
                        AND B2.color='green')
```

使用In查询来改写Intersect查询

□ 查找rating>7且年龄>25的水手编号

➤ Intersect查询

```
Select S.sid  
from Sailors S  
where s.rating>7  
intersect  
select S2.sid  
from Sailors S2  
where S2.age>25
```

使用In查询来改写Intersect查询

□ 查找rating>7且年龄>25的水手编号

➤ In查询

```
Select S.sid  
from Sailors S  
where s.rating>7 and  
       S.sid in( select S2.sid  
                  from Sailors S2  
                  where S2.age>25 )
```


查找最年长的水手的年龄和名字

```
SELECT S.sname, MAX (S.age)
FROM Sailors S
```

```
SELECT S.sname, S.age
FROM Sailors S
WHERE S.age =
      (SELECT MAX (S2.age)
       FROM Sailors S2)
```

```
SELECT S.sname, S.age
FROM Sailors S
WHERE (SELECT MAX (S2.age)
      FROM Sailors S2)
      = S.age ?
```

在18岁以上水手中,对于每个rating级别中最少有两个水手以上的组中最年轻水手的年龄

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

在18岁以上水手中,对于每个rating级别中最少有两个水手以上的组中最年轻水手的年龄

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
71	zorba	10	16.0
64	horatio	7	35.0
29	brutus	1	33.0
58	rusty	10	35.0

rating	age
1	33.0
7	45.0
7	35.0
8	55.5
10	35.0

rating	
7	35.0

Answer relation

Sehrke

25

查找每条红色船只被租用的次数


```
SELECT B.bid, COUNT (*)  
FROM Sailors S, Boats B, Reserves R  
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color='red'  
GROUP BY B.bid
```

- ❑ 在三个关系连接之上的一个分组操作
- ❑ 如果去掉颜色选择条件,查询结果如何?
- ❑ 如果去掉Sailors和与S.sid相关的条件,情况会怎么样?



在18岁以上水手中,对于每个rating级别中最少有两个水手以上的组中最年轻水手的年龄(子查询)

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age > 18
GROUP BY S.rating
HAVING 1 < (SELECT COUNT (*)
            FROM Sailors S2
            WHERE S.rating=S2.rating)
```



- Having子句中也可以包含子查询
- Having子句可以为 **Count(*) > 1**



本节内容小节

- SQL语言的基本特征
- 样板模式与数据库
- SQL数据定义功能
- 索引的建立和删除
- SQL的查询功能

- SQL数据操纵功能
- 嵌入式SQL
- 约束
- 触发器

Structured Query Language



结 束

