

第三章 关系数据库标准语言SQL

(续1)



3.4 数据查询

3.4.1 单表查询

3.4.2 连接查询

3.4.3 嵌套查询

3.4.4 集合查询

3.4.5 **Select**语句的一般形式



嵌套查询的应用场景

- 一个查询块是对关系集合进行搜索找出满足资格的元组。对目标关系中成员 t 的判断可能会出现情况：

(1) 该成员 t 是否属于某个select结果集合 R ;

$t \in R$ 是否成立? (属于运算)

(2) 该成员是否比某个select集中所有成员或至少一个成员大或小;

$t > R$ 中所有或某个成员是否成立? (比较运算)

(3) 该成员是否能使得集合逻辑式成立;

t 是否能使得逻辑命题 L 成立? 其中: L 是一个通过 t 求出的集合逻辑式。 $L(R(t))$ (逻辑运算)

➤图为学生-课程数据库中的student关系、Course关系、SC关系

Course:

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

SC:

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

Student:

Sno	Sname	Ssex	Sage	Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS



带有IN谓词的子查询

[例]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 Student
WHERE Sno IN
    (SELECT Sno
     FROM SC
     WHERE Cno IN
        (SELECT Cno
         FROM Course
         WHERE Cname
            = '信息系统'
        )
    );
```

③ 最后在Student关系中
取出Sno和Sname

② 然后在SC关系中找到选
修了3号课程的学生学号

① 首先在Course关系中
找出“信息系统”的课
程号，为3号



嵌套查询求解方法

- 不相关子查询

子查询的查询条件不依赖于父查询

- 由里向外逐层处理。即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件。



嵌套查询求解方法

- 相关子查询

子查询的查询条件依赖于父查询

- 首先取外层查询中表的第一个元组，根据它与内层查询相关的属性值处理内层查询，若WHERE子句返回值为真，则取此元组放入结果表
- 然后再取外层表的下一个元组
- 重复这一过程，直至外层表全部检查完为止



情形二、带有比较运算符的嵌套子查询

[例] 找出每个超过他选修课程平均成绩的学生学号及课程号。

分析：

- 1) 用一个select块构造一个能够查询某个元组t的平均成绩的子函数
- 2) 从SC中搜索出满足其成绩大于该元组平均成绩的选课记录

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

相关子查询



带有比较运算符的子查询（续）

- 可能的执行过程：

1. 从外层查询中取出SC的一个元组x，将元组x的Sno值（200215121）传送给内层查询。


```
SELECT AVG(Grade)
FROM SC y
WHERE y.Sno='200215121';
```

2. 执行内层查询，得到值88（近似值），用该值代替内层查询，得到外层查询：

```
SELECT Sno, Cno
FROM SC x
WHERE Grade >=88;
```

3. 执行这个查询，得到 (200215121, 1) (200215121, 3)
4. 外层查询取出下一个元组重复做上述1至3步骤，直到外层的SC元组全部处理完毕。结果为：

(200215121, 1) (200215121, 3) (200215122, 2)



➤图为学生-课程数据库中的student关系、Course关系、SC关系

Course:

Cno	Cname	Cpno	Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

SC:

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

Student:

Sno	Sname	Ssex	Sage	Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS



带有比较运算符的子查询问题分析

问题：在嵌套查询情形二中，若需要判断关系中元组t与一个集合的“任意一个”或“所有”是否满足某个关系式，该如何解决？

解决：SQL中给出了两个特殊聚集函数ANY(SOME)、ALL，能够求出一个集合的“任意一个”或“所有”元组

谓词语法：

- ANY (R) : R的任意一个值
- ALL (R) : R所有值

谓词语义：

与关系运算符配合使用

> any(R), 表示“至少比R...的某一个大”；

> all(R), 表示“比R...所有大”；




带有ANY或ALL谓词的子查询示例

[例] 查询其他系中比计算机科学**某一学生年龄**小的学生姓名和年龄

分析：1) 用一个select找出计算机专业所有学生年龄集合R;
2) 对R做any运算, any(R)为R中任意一个;
3) 从学生表中找出满足“非计算机专业”且其年龄小于any(R)的元组

```
SELECT Sname, Sage
FROM Student
WHERE Sage < ANY (SELECT Sage
                   FROM Student
                   WHERE Sdept= 'CS')
AND Sdept <> 'CS' ;
```



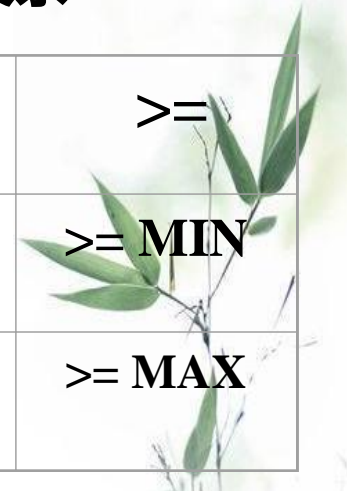
带有ANY或ALL谓词的子查询示例

- 上例可以用聚集函数实现

```
SELECT Sname, Sage
FROM Student
WHERE Sage < (SELECT MAX(Sage)
              FROM Student
              WHERE Sdept= 'CS ')
AND Sdept <> 'CS' ;
```

表 ANY、ALL谓词与聚集函数、IN谓词的等价转换关系

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>= MIN
ALL	--	NOT IN	<MIN	<= MIN	>MAX	>= MAX



带有ANY或ALL谓词的子查询示例

[例] 查询其他系中比计算机科学系所有学生年龄都小的学生姓名及年龄。

方法一：用ALL谓词

```
SELECT Sname, Sage
FROM Student
WHERE Sage < ALL
      (SELECT Sage
       FROM Student
       WHERE Sdept= ' CS ')
AND Sdept <> ' CS ';
```



带有ANY或ALL谓词的子查询示例

方法二：用聚集函数

```
SELECT Sname, Sage
FROM Student
WHERE Sage <
      (SELECT MIN(Sage)
       FROM Student
       WHERE Sdept= ' CS ')
AND Sdept <>' CS ';
```



带有**ANY**或**ALL**谓词的子查询（续）

- 用集函数实现子查询通常比直接用**ANY**或**ALL**查询效率要高，因为前者通常能够减少比较次数
- 原因：集函数首先把需要比较的集合计算出来（通常情况下是变小的），缩小了和父查询比较的次数



情形三、带有逻辑表达式的嵌套子查询

- **EXISTS**: 本质上是一个返回值为“真”或“假”的集函数, 用于判断一个集合是否为空。
 - 若内层查询结果非空, 则外层的WHERE子句返回真值
 - 若内层查询结果为空, 则外层的WHERE子句返回假值
- **NOT EXISTS**: 用于判断一个集合是否不为空。
 - 若内层查询结果非空, 则外层的WHERE子句返回假值
 - 若内层查询结果为空, 则外层的WHERE子句返回真值



带有EXISTS谓词的子查询示例

[例] 查询所有选修了2号课程的学生姓名。

分析：

- 1) 对SC使用一个select块编写一个对参数Student.Sno，求其选2号课的选课记录集合R；
- 2) 再对R使用集合逻辑函数EXISTS，用于判断Student.Sno是否选过2号课
- 3) 对Student中每个学生元组t，选择出能够让逻辑函数EXISTS(R)成立的元组

```
SELECT Sname  
FROM Student  
WHERE EXISTS  
    (SELECT *  
        FROM SC  
        WHERE Sno=Student.Sno AND Cno= ' 2 ');
```



带有EXISTS谓词的子查询

■ 带EXISTS谓词查询的执行过程

外	Sno	Sname	Ssex	Sage	Sdept	内	Sno	Cno	Grade
→	0215121	李勇	男	20	CS	→	0215121	1	92
	0215122	刘晨	女	19	IS	→	0215121	2	85
	0215123	王敏	女	18	MA		0215126	2	88
	0215125	张立	男	18	IS		0215122	2	90
							0215122	3	80

➤ 用带EXISTS谓词的嵌套查询

SELECT Sname FROM Student

WHERE EXISTS

(SELECT * FROM SC

WHERE Sno=Student.Sno AND Cno= '2')

Sname
李勇

带有EXISTS谓词的子查询

■ 带EXISTS谓词查询的执行过程：

外	Sno	Sname	Ssex	Sage	Sdept	内	Sno	Cno	Grade
	0215121	李勇	男	20	CS		0215121	1	92
	0215122	刘晨	女	19	IS		0215121	2	85
	0215123	王敏	女	18	MA		0215126	2	88
	0215125	张立	男	18	IS		0215122	2	90
							0215122	3	80

➤ 用带EXISTS谓词的嵌套查询

SELECT Sname FROM Student

WHERE EXISTS

(SELECT * FROM SC

WHERE Sno=Student.Sno AND Cno= '2')

Sname
李勇
刘晨


带有EXISTS谓词的子查询示例

[例] 查询与“刘晨”在同一个系学习的学生学号和姓名。

- 1) 用EXISTS函数判断任一个学生t，其院系与刘晨院系是否相同
- 2) 从学生表中，搜索让上述逻辑式为真的元组

```
SELECT Sno, Sname  
FROM Student S1  
WHERE EXISTS
```

```
( SELECT *  
  FROM Student S2  
  WHERE S2.Sdept = S1.Sdept AND  
        S2.Sname = '刘晨' );
```



查询其他系中比计算机科学某一学生年龄小的学生姓名和年龄

```
SELECT Sname , Sage  
FROM Student S1  
WHERE EXISTS (SELECT * FROM Student S2  
               WHERE Sdept = 'CS' AND S1.Sage < S2.Sage)  
               AND Sdept <> 'CS' ;
```

注：S1是其他系学生元组变量，S2是计算机系元组变量。



带有EXISTS谓词的子查询(续)

[例] 查询没有选修1号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
  (SELECT *
   FROM SC
   WHERE Sno = Student.Sno AND Cno='1');
```

NOT EXISTS: 用于判断一个集合是否不为空。

- 若内层查询结果非空，则外层的WHERE子句返回假值
- 若内层查询结果为空，则外层的WHERE子句返回真值

外

Sno	Sname	Ssex	Sage	Sdept
0215121	李勇	男	20	CS
0215122	刘晨	女	19	IS
0215123	王敏	女	18	MA
0215125	张立	男	18	IS

内

Sno	Cno	Grade
0215121	1	92
0215121	2	85
0215126	2	88
0215122	2	90
0215122	3	80

带有EXISTS谓词的子查询(续)

[例] 查询没有选修1号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
  (SELECT *
   FROM SC
   WHERE Sno = Student.Sno AND Cno='1');
```

NOT EXISTS: 用于判断一个集合是否不为空。

- 若内层查询结果非空，则外层的WHERE子句返回假值
- 若内层查询结果为空，则外层的WHERE子句返回真值

Sno	Sname	Ssex	Sage	Sdept
0215121	李勇	男	20	CS
0215122	刘晨	女	19	IS
0215123	王敏	女	18	MA
0215125	张立	男	18	IS

Sno	Cno	Grade
0215121	1	92
0215121	2	85
0215126	2	88
0215122	2	90
0215122	3	80

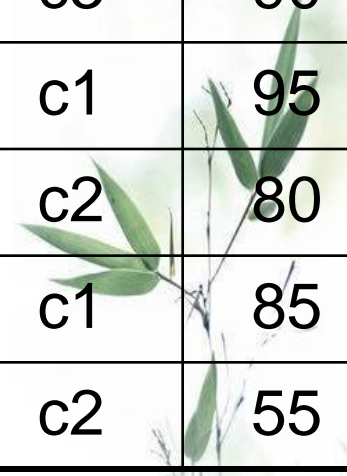
查询所有课程成绩均及格的学生学号和姓名。

S

sno	sn	age	sex
s1	丁岩	19	M
s2	王爽	17	F
s3	李红	18	F
s4	赵立	21	M

SC

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	83
s2	c2	56
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	85
s4	c2	55



```
select sno, sn from s
where not exist
( select *
  from sc
  where s.sno=sc.sno
    and g<60);
```

S

sno	sn	age	sex
s1	丁岩	19	M
s2	王爽	17	F
s3	李红	18	F
s4	赵立	21	M

查询所有课程成绩均及格的学生学号和姓名。

SC

sno	cno	G
s1	c1	79
s1	c3	85
s2	c1	83
s2	c2	56
s2	c3	90
s3	c1	95
s3	c2	80
s4	c1	85
s4	c2	55

带有EXISTS谓词的子查询(续)

- 不同形式的查询间的替换
 - 一些带EXISTS或NOT EXISTS谓词的子查询不能被其他形式的子查询等价替换
 - 所有带IN谓词、比较运算符、ANY和ALL谓词的子查询都能用带EXISTS谓词的子查询等价替换



用EXISTS/NOT EXISTS实现全称量词(难点)

SQL语言中没有全称量词 \forall (For all)

可以把带有全称量词的谓词转换为等价的带有存在量词的谓词:

$$(\forall x)P \equiv \neg (\exists x(\neg P))$$

即对于每个x 都满足P成立, 可以改写为: 不存在一个x使得P不成立



方法1:

用EXISTS/NOT EXISTS实现全称量词示例

[例] 查询选修了全部课程的学生姓名。

【分析】 “选修了全部课程” \Leftrightarrow “没有一门课他没有选”

SELECT Sname

FROM Student

WHERE NOT EXISTS //不存在学生t没选任一课程c

(SELECT * FROM Course

WHERE NOT EXISTS //学生t 选修了一门课程c， 为

(SELECT * FROM SC “假”，即：t没选课程c

WHERE Sno= Student.Sno //学生t 选修

AND Cno= Course.Cno) 了一门课程c

) ;

[例] 查询选修了全部课程的学生姓名。

- 方法2：用集函数

```
SELECT Sname  
FROM Student  
WHERE Sno IN  
( SELECT Sno  
  FROM SC  
   GROUP BY Sno  
   HAVING COUNT(*) =  
     (SELECT COUNT(*)  
      FROM Course));
```



用EXISTS/NOT EXISTS实现逻辑蕴含示例

[例] 查询至少选修了2号学生选修的全部课程的学生的学号。

【分析】

- 用逻辑蕴涵表达：查询学号为x的学生，对所有的课程y，只要2号学生选修了课程y，则x也选修了y。
- 形式化表示：
用P表示谓词 “2号学生选修了课程y”
用q表示谓词 “学生x选修了课程y”
则上述查询为: $(\forall y) p \rightarrow q$
- 利用谓词演算，做等价变换：
$$(\forall y)p \rightarrow q \equiv \neg \exists y(p \wedge \neg q)$$
- 变换后语义：不存在这样的课程y，2号学生选修了y，而学生x没有选。



- 题目要求：查询至少选修了2号学生选修的全部课程的学生学号，用NOT EXISTS谓词表示：

SELECT DISTINCT Sno

FROM SC X

WHERE NOT EXISTS //不存在一门课2号学生选了，学生t 没选

(SELECT * FROM SC Y

WHERE Sno = '2' AND

NOT EXISTS //学生t 没选2号学生选的一门课c

(SELECT * FROM SC Z

WHERE Z.Sno= X.Sno AND //学生t 选修了2号
Z.Cno= Y.Cno)) ; 学生选的一门课c



3.4 数据查询

3.4.1 单表查询

3.4.2 连接查询

3.4.3 嵌套查询

3.4.4 集合查询

3.4.5 基于派生表的查询



3.4.4 集合查询

- 集合操作的种类
 - 并操作UNION
 - 交操作INTERSECT
 - 差操作EXCEPT
- 参加集合操作的各查询结果的列数必须相同；对应项的数据类型也必须相同



集合查询（续）

并操作

[例] 查询计算机科学系的学生及年龄不大于19岁的学生。

方法一：

```
SELECT *  
FROM Student  
WHERE Sdept= 'CS'  
UNION  
SELECT *  
FROM Student  
WHERE Sage<=19;
```

- UNION: 将多个查询结果合并起来时，系统自动去掉重复元组。
- UNION ALL: 将多个查询结果合并起来时，保留重复元组。



集合查询（续）

方法二：

```
SELECT DISTINCT *  
FROM Student  
WHERE Sdept= 'CS' OR Sage<=19;
```



集合查询（续）

[例] 查询选修了课程1或者选修了课程2的学生。

```
SELECT Sno  
FROM SC  
WHERE Cno=' 1 '  
UNION  
SELECT Sno  
FROM SC  
WHERE Cno=' 2 ';
```



集合查询（续）

交操作

[例] 查询计算机科学系的学生与年龄不大于19岁的学生的交集

```
SELECT *  
FROM Student  
WHERE Sdept='CS'  
INTERSECT  
SELECT *  
FROM Student  
WHERE Sage<=19
```



集合查询（续）

- [例] 实际上就是查询计算机科学系中年龄不大于19岁的学生

```
SELECT *  
FROM Student  
WHERE Sdept= 'CS' AND  Sage<=19;
```



集合查询（续）

[例] 查询选修课程1的学生集合与选修课程2的学生集合的交集

```
SELECT Sno  
FROM SC  
WHERE Cno='1 '  
INTERSECT  
SELECT Sno  
FROM SC  
WHERE Cno='2 ';
```



集合查询（续）

[例]实际上是查询既选修了课程1又选修了课程2
的学生

```
SELECT Sno  
FROM SC  
WHERE Cno=' 1 ' AND Sno IN  
      (SELECT Sno  
       FROM SC  
       WHERE Cno=' 2 ');
```



集合查询（续）

差操作

[例] 查询计算机科学系的学生与年龄不大于19岁的学生的差集。

```
SELECT *  
FROM Student  
WHERE Sdept='CS'  
EXCEPT  
SELECT *  
FROM Student  
WHERE Sage <=19;
```



集合查询（续）

[例]实际上是查询计算机科学系中年龄大于19岁的学生

```
SELECT *  
FROM Student  
WHERE Sdept= 'CS' AND  Sage>19;
```



差操作（续）

[例] 查询学生姓名与教师姓名的差集

实际上是查询学校中未与教师同名的学生姓名

```
SELECT DISTINCT Sname
```

```
FROM Student
```

```
WHERE Sname NOT IN
```

```
  (SELECT Tname
```

```
   FROM Teacher);
```



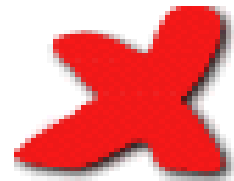
对集合操作结果的排序

- **ORDER BY**子句只能用于对最终查询结果排序，不能对中间结果排序
- 对集合操作结果排序时，**ORDER BY**子句中用数字指定排序属性



对集合操作结果的排序

```
SELECT * FROM Student  
WHERE Sdept= 'CS' ORDER BY Sno  
UNION  
SELECT * FROM Student  
WHERE Sage<=19 ORDER BY Sno;
```



```
SELECT * FROM Student  
WHERE Sdept= 'CS'  
UNION  
SELECT * FROM Student  
WHERE Sage<=19  
ORDER BY Sno;
```



基于派生表的查询

- 子查询还可以出现在from子句中。
- 派生表是在外部查询的**FROM**子句中定义的。派生表的存在范围为定义它的外部查询，只要外部查询一结束，派生表就不存在了。定义派生表的查询语句要写在一对圆括号内，后面跟着**AS**子句和派生表的名称。
- 例：找出每个学生超过他自己选修课程平均成绩的课程号。

Select sno,cno

From sc, (select sno avg(grade) from sc group by sno)

as avg_sc (avg_sno,avg_grade)

where sc.sno=avg_sc.sno and sc.grade>=avg_sc.avg_grade



SC:

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

avg_sc

avg_sno	avg_grade
200215121	88
200215122	85

Select sno,cno

From sc, (select sno avg(grade) from sc group by sno)

as avg_sc (avg_sno,avg_grade)

where sc.sno=avg_sc.sno and sc.grade>=avg_sc.avg_grade



- 如果子查询中没有聚集函数，派生表可以不指定属性列，子查询**SELECT**子句后面的列名为其缺省属性。
- [例3.60]查询所有选修了1号课程的学生姓名，可以用如下查询完成：

```
SELECT Sname  
FROM Student,  
(SELECT Sno FROM SC WHERE Cno='1 ') AS SC1  
WHERE Student.Sno = SC1.Sno;
```

