数据库系统概论

An Introduction to Database System

第三章 关系数据库标准语言SQL

数据查询

（连接查询）

3.4.2 连接查询

- 不像关系代数中“连接”是用一个特殊符号来表达的，在SQL中“连接”是用“连接条件”来表达的。
- 连接条件或连接谓词：用来连接两个表的条件

  一般格式：[<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>

- 连接字段：连接谓词中的列名称
  - 连接条件中的各连接字段类型必须是可比的，但名字不必相同
- 语义：SELECT $A_1, \dots, A_n$
  FROM $R_1, \dots, R_n$ ⇒ $\Pi_{A1, \dots, An}(\sigma_F(R_1 \times \dots \times R_n))$
  WHERE F ;

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

1. 等值与非等值连接查询

❖等值连接：连接运算符为 “=”

[例 3.49] 查询每个学生及其选修课程的情况

```
SELECT  Student.*, SC.*
FROM    Student, SC
WHERE   Student.Sno = SC.Sno;
```

等值与非等值连接查询（续）

查询结果：

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
201215121	李勇	男	20	CS	201215121	1	92
201215121	李勇	男	20	CS	201215121	2	85
201215121	李勇	男	20	CS	201215121	3	88
201215122	刘晨	女	19	CS	201215122	2	90
201215122	刘晨	女	19	CS	201215122	3	80

等值与非等值连接查询（续）

- 自然连接
- 采用在SELECT中去掉重复字段的方式实施

[例 3.50] 对[例 3.49]用自然连接完成。

```
SELECT  Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade
FROM    Student,SC
WHERE   Student.Sno = SC.Sno;
```

等值与非等值连接查询（续）

[例 3.51]查询选修2号课程且成绩在70分以上的所有学生的学号和姓名。

```
SELECT Student.Sno, Sname
FROM   Student, SC
WHERE  Student.Sno=SC.Sno  AND
       SC.Cno=' 2 ' AND SC.Grade>70;
```

连接谓词

选择谓词

一条SQL语句可以同时完成选择和连接查询，这时WHERE子句是由连接谓词和选择谓词组成的复合条件。

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

2. 自身连接

- 自身连接：一个表与其自己进行连接，是一种特殊的连接
- 需要给表起别名以示区别
- 由于所有属性名都是同名属性，因此必须使用别名前缀

[例 3.52]查询每一门课的直接先修课的名称

```
SELECT  FIRST.Cname ，  SECOND.Cname
FROM  Course  FIRST, Course  SECOND
WHERE FIRST.Cpno = SECOND.Cno;
```

FROM子句中，可以为表或视图取一别名，这别名只在本句中有效

自身连接（续）

FIRST表(Course表)SECOND表

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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	数据处理导论		2
7	C语言	6	4

查询结果：

First.Cname	Second.Cname
数据库	数据结构
信息系统	数据库
操作系统	数据处理导论
数据结构	C语言
C语言	数据处理导论

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

3. 外连接

- 外连接与普通连接的区别
  - 普通连接操作只输出满足连接条件的元组
  - 外连接操作以指定表为连接主体，将主体表中不满足连接条件的元组一并输出
  - 左外连接
    - 列出左边关系中所有的元组
  - 右外连接
    - 列出右边关系中所有的元组

select <目标列表达式> [into 表名]
from 表名1 [inner | right | left | full] [outer] join 表名2 on 条件
inner join： 内连接，显示符合on指定连接条件的记录

left [outer] join：左外连接，连接结果中包括左表(表名1)中的所有行，而不仅仅是连接列所匹配的行。如果表名1的某行在表名2中没有匹配行，则在该行新增加的属性上填空值。

right (outer) join 右外连接，返回右表（表名2）的所有行。如果右表的某行在左表中没有匹配行，则在新增加的属性上填空值。

full (outer) join 完全外连接，返回左表和右表中的所有行。当某行在另一个表中没有匹配行时,则另一个表新增加的属性上填空值

外连接（续）

[例 3. 53] 改写[例 3.49]

```
SELECT Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade
FROM  Student  LEFT JOIN SC ON  (Student.Sno=SC.Sno);
```

执行结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
201215121	李勇	男	20	CS	1	92
201215121	李勇	男	20	CS	2	85
201215121	李勇	男	20	CS	3	88
201215122	刘晨	女	19	CS	2	90
201215122	刘晨	女	19	CS	3	80
201215123	王敏	女	18	MA	NULL	NULL
201215125	张立	男	19	IS	NULL	NULL

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

4. 多表连接

❖ 多表连接：两个以上的表进行连接

[例3.54]查询每个学生的学号、姓名、选修的课程名及成绩

```
SELECT Student.Sno, Sname, Cname, Grade
FROM   Student, SC, Course    /*多表连接*/
WHERE Student.Sno = SC.Sno
      AND SC.Cno = Course.Cno;
```

应用示例

C（CNO，CNAME，Credit，CreditHours，CPNO，TNO）

S（SNO，SNAME，AGE，SEX，NativePlace）

T（TNO，TNAME，TITLE，SEX）

SC（SNO，CNO，Grade）

1.统计每门课程的学生选修人数，要求显示课程号、课程名和学生人数

– Select C.Cno, Cname, COUNT(Sno) as 学生人数
  From C, SC
  Where C.Cno = SC.Cno
  Group By C.Cno, Cname ;

	Cno	Cname	学生人数
1	C1	Math	2
2	C2	English	3
3	C3	PM	2
4	C4	DB	4

CNO	CNAME	Credit	CreditHours	CPNO	TNO
C1	Math	3	48	NULL	T1
C2	English	4	64	NULL	T2
C3	PM	2	32	C2	T2
C4	DB	3.5	56	C1	T1

SNO	CNO	Grade
S1	C2	80
S1	C3	70
S1	C4	85
S2	C1	60
S2	C2	75
S2	C3	90
S2	C4	NULL
S3	C1	85
S3	C4	80
S4	C2	85
S4	C4	75

应用示例（续）

2.按教师号统计每位教师每门课程的学生选修人数，要求：

- 1）仅显示选修人数在3人（>=3）以上的信息
- 2）显示TNO、CNO和选修人数
- 3）显示时，查询结果按选修人数降序排列，人数相同按TNO升序、 CNO降序排列

C（CNO，CNAME，Credit，CreditHours，CPNO，TNO）

S（SNO，SNAME，AGE，SEX，NativePlace）

T（TNO，TNAME，TITLE，SEX）

SC（SNO，CNO，Grade）

应用示例（续）

C（CNO，CNAME，Credit，CreditHours，CPNO，TNO）

S（SNO，SNAME，AGE，SEX，NativePlace）

T（TNO，TNAME，TITLE，SEX）

SC（SNO，CNO，Grade）

SNO	CNO	Grade
S1	C2	80
S1	C3	70
S1	C4	85
S2	C1	60
S2	C2	75
S2	C3	90
S2	C4	NULL
S3	C1	85
S3	C4	80
S4	C2	85
S4	C4	75

- Select Tno, C.Cno, COUNT(Sno) as 选修人数
  From C, SC Where C.Cno = SC.Cno
  Group By Tno, C.Cno
  Having COUNT(*)>=3
  Order By 3 DESC, Tno, C.Cno DESC

	Tno	Cno	选修人数
1	T1	C4	4
2	T2	C2	3
3	T1	C1	2
4	T2	C3	2

	Tno	Cno	选修人数
1	T1	C4	4
2	T2	C2	3

CNO	CNAME	Credit	CreditHours	CPNO	TNO
C1	Math	3	48	NULL	T1
C2	English	4	64	NULL	T2
C3	PM	2	32	C2	T2
C4	DB	3.5	56	C1	T1

重点回顾

- ［GROUP BY <列名序列>［HAVING <组条件表达式>］］
  - 作用
    - 1）数据按GROUP BY子句列名序列中的列值进行分组
    - 2）组内数据按SELECT子句中的聚集函数进行计算
    - 3）提取满足HAVING子句的条件表达式值的分组
  - 注意
    - HAVING子句支持聚集函数
    - WHERE子句不支持聚集函数
    - SELECT子句只能取聚集函数或GROUP BY子句指的列

{列名表 1} ⊂ {列名表 2}

```
Select {列名表 1}，聚集函数
… …
Group By {列名表 2}
… …   ;
```

重点回顾（续）

- **GROUP BY**子句分组：

  细化聚集函数的作用对象

  - 未对查询结果分组，聚集函数将作用于整个查询结果
  - 对查询结果分组后，聚集函数将分别作用于每个组
  - 作用对象是查询的中间结果表
  - 按指定的一列或多列值分组，值相等的为一组

重点回顾（续）

- HAVING短语与WHERE子句的区别：
  - 作用对象不同
  - WHERE子句作用于基表或视图，从中选择满足条件的元组
  - HAVING短语作用于组，从中选择满足条件的组。

重点回顾（续）

- ［ORDER BY <列名｜列序号>［ASC｜DESC］
  ［｛,<列名｜列序号>［ASC｜DESC］｝］
  - 对查询结果按子句中指定列的值排序，如果ORDER BY后有多个列名
    - 先按第一列名值排序
    - 再对于第一列值相同的行，按第二列名值排序
    - 依次类推… …
  - 列序号是在SELECT子句中出现的序号（选的列是聚集函数或表达式时）
  - ASC表示升序，DESC表示降序，缺省时表示升序

数据查询

（嵌套查询）

嵌套查询

- 嵌套查询概述
  - 一个SELECT-FROM-WHERE语句称为一个查询块
  - 将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中的查询称为嵌套查询

```
SELECT Sname                          /*外层查询/父查询*/
FROM Student
WHERE Sno IN
            ( SELECT Sno              /*内层查询/子查询*/
              FROM SC
              WHERE Cno= ‘ 2 ‘);
```

嵌套查询（续）

- 上层的查询块称为外层查询或父查询
- 下层查询块称为内层查询或子查询
- SQL语言允许多层嵌套查询
- 子查询的限制
  - 不能使用ORDER BY子句

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（续）

- 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

3.4.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

1. 带有IN谓词的子查询

[例 3.55] 查询与“刘晨”在同一个系学习的学生。

此查询要求可以分步来完成

① 确定“刘晨”所在系名

```
SELECT  Sdept
FROM    Student
WHERE  Sname= ' 刘晨 ';
```

结果为： CS

带有IN谓词的子查询（续）

② 查找所有在CS系学习的学生。

```
SELECT   Sno, Sname, Sdept
FROM     Student
WHERE    Sdept= ' CS ';
```

结果为：

Sno	Sname	Sdept
201215121	李勇	CS
201215122	刘晨	CS

带有IN谓词的子查询（续）

将第一步查询嵌入到第二步查询的条件中

```
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept  IN
          (SELECT Sdept
           FROM Student
           WHERE Sname= ' 刘晨 ');
```

此查询为不相关子查询。

带有IN谓词的子查询（续）

用自身连接完成[例 3.55]查询要求

```
SELECT  S1.Sno, S1.Sname,S1.Sdept
FROM    Student S1,Student S2
WHERE  S1.Sdept = S2.Sdept  AND
       S2.Sname = '刘晨';
```

带有IN谓词的子查询（续）

[例 3.56]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关系中找出选修了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

带有IN谓词的子查询（续）

用连接查询实现[例 3.56]：

```
SELECT Student.Sno,Sname
FROM    Student,SC,Course
WHERE Student.Sno = SC.Sno  AND
            SC.Cno = Course.Cno AND
            Course.Cname='信息系统';
```

3.4.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

2. 带有比较运算符的子查询

- 当能确切知道内层查询返回单值时，可用比较运算符（>，<，=，>=，<=，!=或< >）。

在[例 3.55]中，

```
SELECT Sno,Sname,Sdept
FROM   Student
WHERE Sdept  =  /*由于一个学生只可能在一个系学习，用=代替*/
      (SELECT Sdept
       FROM   Student
       WHERE Sname= '刘晨');
```

带有比较运算符的子查询（续）

[例 3.57]找出每个学生超过他选修课程平均成绩的课程号。

```
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
  - 从外层查询中取出SC的一个元组x，将元组x的Sno值（201215121）传送给内层查询。

```
SELECT AVG(Grade)
FROM SC y
WHERE y.Sno='201215121‘;
```

带有比较运算符的子查询（续）

- 可能的执行过程（续）
  - 执行内层查询，得到值88.33（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 >=88.33;
```

  - 执行该查询，得到结果：

（201215121,1）

带有比较运算符的子查询（续）

- 然后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:

（201215121,1）

（201215122,2）

3.4.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY或ALL谓词的子查询

4.带有EXISTS谓词的子查询

带有ANY（SOME）或ALL谓词的子查询（续）

使用ANY或ALL谓词时必须同时使用比较运算

语义为：

> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值

带有ANY（SOME）或ALL谓词的子查询 （续）

[例 3.58] 查询非计算机科学系中比计算机科学系任意一个学生年龄小的学生姓名和年龄

```
SELECT Sname,Sage
FROM   Student
WHERE Sage < ANY (SELECT  Sage
                  FROM    Student
                  WHERE Sdept= ' CS ')
 AND Sdept <> 'CS ' ;        /*父查询块中的条件 */
```

带有ANY（SOME）或ALL谓词的子查询 （续）

结果：

Sname	Sage
王敏	18
张立	19

执行过程：

（1）首先处理子查询，找出CS系中所有学生的年龄，构成一个集合（20,19）

（2）处理父查询，找所有不是CS系且年龄小于20 或 19的学生

带有ANY（SOME）或ALL谓词的子查询（续）

- 用聚集函数实现[例 3.58]

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

带有ANY（SOME）或ALL谓词的子查询（续）

[例 3.59] 查询非计算机科学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（SOME）或ALL谓词的子查询（续）

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（SOME）或ALL谓词的子查询（续）

表3.7 ANY（或SOME），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

3.4.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

带有EXISTS谓词的子查询

❖ EXISTS谓词

- 存在量词∃
- 带有EXISTS谓词的子查询不返回任何数据，只产生逻辑真值“true”或逻辑假值“false”。
  - 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
- 由EXISTS引出的子查询，其目标列表达式通常都用 * ，因为带EXISTS的子查询只返回真值或假值，给出列名无实际意义。

带有EXISTS谓词的子查询（续）

- **NOT EXISTS谓词**
  - 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（续）

[例 3.60]查询所有选修了1号课程的学生姓名。

思路分析：

- 本查询涉及Student和SC关系
- 在Student中依次取每个元组的Sno值，用此值去检查SC表
- 若SC中存在这样的元组，其Sno值等于此Student.Sno值，并且其Cno= ‘1’，则取此Student.Sname送入结果表

带有EXISTS谓词的子查询（续）

```
SELECT Sname
    FROM Student
    WHERE EXISTS
            (SELECT *
             FROM SC
             WHERE Sno=Student.Sno AND Cno= ' 1 ');
```

其他方法

1）连接查询

```
Select Sname
    From Student, SC
    Where Student.Sno=SC.Sno and Cno='1';
```

2）IN嵌套查询

```
Select Sno, Sname From Student Where Sno in
        ( Select Sno From SC Where Cno='1' ) ;
Select Sno, Sname From Student Where '1' in
        ( Select Cno From SC Where Sno=Student.Sno ) ;
```

带有EXISTS谓词的子查询（续）

[例 3.61] 查询没有选修1号课程的学生姓名。

$$\Pi_{SNAME}(Student) - \Pi_{SNAME}(\sigma_{CNO='1'}(Student \bowtie SC))$$

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

不能用连接查询完成

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
  - SQL语言中没有全称量词∀（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：

    （∀x）P ≡ ¬ （∃ x（¬ P））

带有EXISTS谓词的子查询（续）

[例 3.62] 查询选修了全部课程的学生姓名。

$$\Pi_{SNAME}(\text{Student} \bowtie (\Pi_{SNO,\ CNO}(\text{SC}) \div \Pi_{CNO}(\text{Course})))$$

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

数据查询

（集合查询）

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）

[例 3.64] 查询“计算机科学系的学生”及“年龄不大于19岁的学生”。

```
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）

[例 3.65] 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）

[例3.66] 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 3.66] 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 3.67]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 '
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='2 ';
```

集合查询（续）

[例3.67]查询既选修了课程1又选修了课程2的学生。

也可以表示为：

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

❖ 也可表示为自身连接，同学们自己思考完成

集合查询（续）

[例3.67]查询既选修了课程1又选修了课程2的学生。

自身连接 $\Pi_1(\sigma_{1=4 \wedge 2='1' \wedge 5='2'}(SC \times SC))$

```
Select X.Sno
    From SC as X, SC as Y
    Where X.Sno=Y.Sno
            and X.Cno='1'
             and Y.Cno='2' ;
```

集合查询（续）

SC

学号	课程号	成绩
98410	C4	85
98410	C2	90
98410	C2	85
98410	C4	90

SC

学号	课程号	成绩
98410	C4	85
98410	C2	90
98410	C2	85
98410	C4	90

SC×SC

学号	课程号	成绩	学号	课程号	成绩
98410	C4	85	98410	C4	85
~~98410~~	~~C4~~	~~85~~	~~98410~~	~~C2~~	~~90~~
98410	C4	85	98410	C2	85
98410	C4	85	98410	C4	90
…	…	…	…	…	…

思考：至少选了2门课的学生学号？至少有两个人选的课程号？

集合查询（续）

[例 3.68] 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3.68]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;
```

3.4.5 基于派生表的查询

- 子查询不仅可以出现在WHERE子句中，还可以出现在FROM子句中，这时子查询生成的临时派生表（Derived Table）成为主查询的查询对象

[例3.57]找出每个学生超过他自己选修课程平均成绩的课程号，可改写为：

```
SELECT Sno, Cno
FROM SC, (SELECT Sno avg_sno, Avg(Grade)  avg_grade
                FROM SC
                   GROUP BY Sno)
                AS   Avg_sc
WHERE SC.Sno = Avg_sc.avg_sno
  and SC.Grade >=Avg_sc.avg_grade;
```

基于派生表的查询（续）

- 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;
```

小结： SELECT语句的一般格式

SELECT [ALL|DISTINCT]

<目标列表达式> [别名] [,<目标列表达式> [别名]] …

FROM <表名或视图名> [别名]

[,<表名或视图名> [别名]] …

|(<SELECT语句>)[AS]<别名>

[WHERE <条件表达式>]

[GROUP BY <列名1>[HAVING<条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]];

1. 目标列表达式的可选格式

❖ 目标列表达式格式

（1）*

（2）<表名>.*

（3）COUNT([DISTINCT|ALL]*)

（4）[<表名>.]<属性列名表达式>[,<表名>.]<属性列名表达式>…

其中<属性列名表达式>可以是由属性列、作用于属性列的聚集函数和常量的任意算术运算（+，-，*，/）组成的运算公式

2. 聚集函数的一般格式

{ COUNT
SUM
AVG
MAX
MIN } （[DISTINCT|ALL]<列名>）

3. WHERE子句的条件表达式的可选格式

（1）

<属性列名>θ { <属性列名> | <常量> | [ANY|ALL]（SELECT语句） }

（2）

<属性列名>[NOT] BETWEEN { <属性列名> | <常量> | （SELECT语句） } AND { <属性列名> | <常量> | （SELECT语句） }

WHERE子句的条件表达式的可选格式（续）

（3）

<属性列名> [NOT] IN { （ <值1>[,<值2>]… | （SELECT语句） }

（4） <属性列名> [NOT] LIKE <匹配串>

（5） <属性列名> IS [NOT] NULL

（6）[NOT] EXISTS（SELECT语句）

WHERE子句的条件表达式的可选格式（续）

（7）

<条件表达式> { AND / OR } <条件表达式> { AND / OR } <条件表达> …