第三章 关系数据库标准语言

SQL (续1）

3.3 查　询

- 3.3.1 概述
- 3.3.2 单表查询
- 3.3.3 连接查询
- 3.3.4 嵌套查询
- 3.3.5 集合查询
- 3.3.6 小结

3.3.3 连接查询

同时涉及多个表的查询称为连接查询

用来连接两个表的条件称为连接条件或连接谓词

一般格式：

- [<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>

  比较运算符：=、>、<、>=、<=、!=

- [<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2>

  AND [<表名2>.]<列名3>

连接查询（续）

- **连接字段**
  - 连接谓词中的列名称为连接字段
  - 连接条件中的各连接字段类型必须是可比的，但不必是相同的

连接操作的执行过程

嵌套循环法(NESTED-LOOP)

- 首先在表1中找到第一个元组，然后从头开始扫描表2，逐一查找满足条件的连接起来，形成结果表中一个元组。
- 表2全部查找完后，再找表1中第二个元组，然后再从头开始扫描表2，逐一查找满足连接条件的元组，找到后就将表1中的第二个元组与该元组拼接起来，形成结果表中一个元组。
- 重复上述操作，直到表1中的全部元组都处理完毕

连接查询（续）

SQL中连接查询的主要类型

- 广义笛卡尔积
- 等值连接(含自然连接)
- 非等值连接查询
- 自身连接查询
- 外连接查询
- 复合条件连接查询

一、广义笛卡尔积

- 不带连接谓词的连接
- 很少使用

例：

```
SELECT  Student.* ，SC.*
FROM    Student, SC；
```

等值连接

- 连接运算符为 = 的连接操作
  - [<表名1>.]<列名1> = [<表名2>.]<列名2>
  - 任何子句中引用表1和表2中同名属性时，都必须加表名前缀。引用唯一属性名时可以加也可以省略表名前缀。

等值连接

假设Student表、SC表分别有下列数据：

Student表

Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

等值连接

SC表

Sno	Cno	Grade
95001	1	92
95001	2	85
95001 95002 95002	3 2 3	88 90 80

二、等值与非等值连接查询

等值连接

[例32] 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；
```

等值连接

结果表

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
95001	李勇	男	20	CS	95001	1	92
95001	李勇	男	20	CS	95001	2	85
95001	李勇	男	20	CS	95001	3	88
95002	刘晨	女	19	IS	95002	2	90
95002	刘晨	女	19	IS	95002	3	80

自然连接

- 等值连接的一种特殊情况，把目标列中重复的属性列去掉。

[例33] 对[例32]用自然连接完成。

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

非等值连接查询

连接运算符 不是 = 的连接

[<表名1>.]<列名1><比较运算符>[<表名2>.]<列名2>

比较运算符：>、<、>=、<=、!=

[<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2>

AND [<表名2>.]<列名3>

三、自身连接

- 一个表与其自己进行连接，称为表的自身连接
- 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

自身连接（续）

[例34]　查询每一门课的间接先修课（即先修课的先修课）

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

自身连接（续）

FIRST表（Course表）

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

自身连接（续）

SECOND表（Course表）

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

自身连接（续）

查询结果

cno	cpno
1	7
3	5
5	6

四、外连接（Outer Join）

- **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

外连接（续）

- 在表名后面加外连接操作符(*)或(+)指定非主体表
- 非主体表有一“万能”的虚行，该行全部由空值组成
- 虚行可以和主体表中所有不满足连接条件的元组进行连接
- 由于虚行各列全部是空值，因此与虚行连接的结果中，来自非主体表的属性值全部是空值

外连接（续）

- **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

外连接（续）

[例 33] 查询每个学生及其选修课程的情况包括没有选修课程的学生----用外连接操作

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

外连接（续）

结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
95001	李勇	男	20	CS	1	92
95001	李勇	男	20	CS	2	85
95001	李勇	男	20	CS	3	88
95002	刘晨	女	19	IS	2	90
95002	刘晨	女	19	IS	3	80
95003	王敏	女	18	MA		
95004	张立	男	19	IS		

五、复合条件连接

WHERE子句中含多个连接条件时，称为复合条件连接

[例35]查询选修2号课程且成绩在90分以上的所有学生的学号、姓名

```
SELECT Student.Sno, student.Sname
FROM   Student, SC
WHERE Student.Sno = SC.Sno AND    /* 连接谓词*/
          SC.Cno= ' 2 ' AND   /* 其他限定条件 */
          SC.Grade > 90;        /* 其他限定条件 */
```

多表连接

[例36] 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
WHERE Student.Sno = SC.Sno
          and SC.Cno = Course.Cno；
```

结果：

Student.Sno	Sname	Cname	Grade
95001	李勇	数据库	92
95001	李勇	数学	85
95001	李勇	信息系统	88
95002	刘晨	数学	90
95002	刘晨	信息系统	80

练习：

- **Student(S#,Sname,Sage,Ssex)**
- **--S# 学生编号,Sname 学生姓名,Sage 出生年月,Ssex 学生性别**

  Course(C#,Cname,T#)
- **--C# --课程编号,Cname 课程名称,T# 教师编号**

  Teacher(T#,Tname)
- **--T# 教师编号,Tname 教师姓名**

  SC(S#,C#,score)
- **--S# 学生编号,C# 课程编号,score 分**

完成下列查询：

- 1、查询姓“李” 老师的数量 。
- 2、查询学过“张三”老师授课的同学的信息 。
- 3、检索“01”课程分数小于60，按分数降序排列的学生信息。
- 4、查询“01”课程比“02”课程成绩高的学生的信息及课程分数。
- 5、查询平均成绩大于等于60分的同学的学生编号和学生姓名和平均成绩。
- 6、查询所有同学的学生编号、学生姓名、选课总数、所有课程的总成绩。

1、查询姓“李” 老师的数量 。

```
select count(Tname)  姓"李"老师的数量
from Teacher
 where Tname like '李%'
```

2、查询学过“张三”老师授课的同学的信息 。

```
select distinct Student.*
from Student, SC, Course, Teacher
where Student.S# = SC.S# and SC.C# = Course.C
# and Course.T# = Teacher.T# and Teacher.Tnam
e = '张三'
```

3、检索“01”课程分数小于60，按分数降序排列的学生信息。

```
select student.* , sc.C# , sc.score
from student, sc
where student.S# = SC.S# and sc.score < 60 and sc.C# = '01'
order by sc.score desc
```

4、查询“01”课程比“02”课程成绩高的学生的信息及课程分数。

```
select  Student.* , b.score, c.score
from Student, SC b , SC c
where  Student.S# = b.S# and  Student.S# = c.S#
and b.C# = '01' and c.C# = '02' and b.score > c.sco
re
```

5、查询平均成绩大于等于60分的同学的学生编号、学生姓名和平均成绩。

```
select  Student.S# , Student.Sname,
 avg(SC.score)  avg_score
from Student, SC
where  Student.S# =  SC.S#
group by SC.S#
having avg(score) >= 60
```

6、查询所有同学的学生编号、学生姓名、选课总数、所有课程的总成绩。

```
select  Student.S#, Student.Sname, count(SC.C#)
选课总数, sum(score)  所有课程的总成绩
from Student, SC
where  Student.S# = SC.S#
group by  SC.S#
```

3.3 查　询

- 3.3.1 概述
- 3.3.2 单表查询
- 3.3.3 连接查询
- 3.3.4 嵌套查询
- 3.3.5 集合查询
- 3.3.6 小结

3.3.4 嵌套查询

- 嵌套查询概述
- 嵌套查询分类
- 嵌套查询求解方法
- 引出子查询的谓词

嵌套查询(续)

- **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

嵌套查询(续)

```
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)

- 子查询的限制
  - **不能使用ORDER BY子句**
- 层层嵌套方式反映了 **SQL**语言的结构化
- 有些嵌套查询可以用连接运算替代

嵌套查询分类

- 不相关子查询

  子查询的查询条件不依赖于父查询

- 相关子查询

  子查询的查询条件依赖于父查询

嵌套查询求解方法

- **不相关子查询**

  是由里向外逐层处理。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

- 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。

引出子查询的谓词

- 带有IN谓词的子查询
- 带有比较运算符的子查询
- 带有ANY或ALL谓词的子查询
- 带有EXISTS谓词的子查询

一、带有IN谓词的子查询

[例37] 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：

Sdept
IS

带有IN谓词的子查询（续）

② 查找所有在IS系学习的学生。

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

结果为：

Sno	Sname	Sdept
95001	刘晨	IS
95004	张立	IS

构造嵌套查询

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。DBMS求解该查询时也是分步去做的。

带有IN谓词的子查询（续）

用自身连接完成本查询要求

```
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）

父查询和子查询中的表均可以定义别名

```
SELECT Sno，Sname，Sdept
FROM   Student  S1
WHERE  S1.Sdept  IN
      (SELECT Sdept
       FROM    Student  S2
       WHERE S2.Sname=‘刘晨’);
```

带有IN谓词的子查询（续）

[例38]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，结果为3号

带有IN谓词的子查询（续）

结果：

Sno	Sname
95001	**李勇**
95002	**刘晨**

带有IN谓词的子查询（续）

- 用连接查询

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

二、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< >）。

- 与ANY或ALL谓词配合使用

带有比较运算符的子查询（续）

例：假设一个学生只可能在一个系学习，并且必须属于一个系，则在[例37]可以用 = 代替IN ：

```
SELECT Sno，Sname，Sdept
FROM    Student
WHERE Sdept   =
      （SELECT Sdept
      FROM    Student
      WHERE Sname= ‘ 刘晨 ’ ）；
```

带有比较运算符的子查询（续）

子查询一定要跟在比较符之后

错误的例子：

```
SELECT  Sno，Sname，Sdept
FROM    Student
WHERE ( SELECT Sdept
        FROM Student
        WHERE Sname= ‘ 刘晨 ’ )；
```

三、带有ANY或ALL谓词的子查询

谓词语义

- **ANY**：某一个值
- **ALL**：所有值

带有ANY或ALL谓词的子查询（续）

需要配合使用比较运算符

> 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
<= ANY	小于等于子查询结果中的某个值
<= ALL	小于等于子查询结果中的所有值
= ANY	等于子查询结果中的某个值
=ALL	等于子查询结果中的所有值（通常没有实际意义）
!=（或<>）ANY	不等于子查询结果中的某个值
!=（或<>）ALL	不等于子查询结果中的任何一个值

带有ANY或ALL谓词的子查询（续）

[例39] 查询其他系中比信息系其中某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= ' IS ')
    AND Sdept <> ' IS ' ;
        /* 注意这是父查询块中的条件 */
```

带有ANY或ALL谓词的子查询（续）

结果

Sname	Sage
王敏	18

执行过程

1.DBMS执行此查询时，首先处理子查询，找出IS系中所有学生的年龄，构成一个集合(19，18)

2. 处理父查询，找所有不是IS系且年龄小于19 或 18的学生

带有ANY或ALL谓词的子查询（续）

- **ANY和ALL谓词有时可以用集函数实现**
  - **ANY与ALL与集函数的对应关系**

	=	<>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（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

带有ANY或ALL谓词的子查询（续）

[例39]：用集函数实现[例39]

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

带有ANY或ALL谓词的子查询（续）

[例40] 查询其他系中比信息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= ' IS ')
      AND Sdept <> ' IS ';
```

查询结果为空表。

带有ANY或ALL谓词的子查询（续）

方法二：用集函数

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

四、带有EXISTS谓词的子查询

1. EXISTS谓词
2. NOT EXISTS谓词
3. 不同形式的查询间的替换
4. 相关子查询的效率
5. 用EXISTS/NOT EXISTS实现全称量词
6. 用EXISTS/NOT EXISTS实现逻辑蕴函

带有EXISTS谓词的子查询(续）

- 1. 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，则返回真值
    - 若内层查询结果为空，则返回假值
  - 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
- 2. NOT EXISTS谓词

带有EXISTS谓词的子查询(续）

[例41] 查询所有选修了1号课程的学生姓名。

- 用嵌套查询

```
SELECT Sname
FROM Student
WHERE EXISTS
  (SELECT *
   FROM SC                    /*相关子查询*/
   WHERE Sno=Student.Sno AND
Cno= '1');
```

带有EXISTS谓词的子查询(续）

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关系。
- 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送入结果关系。

带有EXISTS谓词的子查询(续）

- 用连接运算

```
SELECT Sname
FROM Student, SC
WHERE Student.Sno=SC.Sno AND
      SC.Cno= '1';
```

带有EXISTS谓词的子查询(续）

[例42] 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');
```

此例用连接运算难于实现

带有EXISTS谓词的子查询(续）

3. 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。

带有EXISTS谓词的子查询(续）

例：[例37]查询与“刘晨”在同一个系学习的学生

- 可以用带EXISTS谓词的子查询替换：

```
SELECT Sno, Sname, Sdept
FROM Student S1
WHERE EXISTS
     ( SELECT *
       FROM Student S2
       WHERE S2.Sdept = S1.Sdept AND
             S2.Sname = ‘刘晨 ’);
```

带有EXISTS谓词的子查询(续）

5. 用EXISTS/NOT EXISTS实现全称量词(难点)

- 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：

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

带有EXISTS谓词的子查询(续）

[例43] 查询选修了全部课程的学生姓名。

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

带有EXISTS谓词的子查询(续）

6. 用EXISTS/NOT EXISTS实现逻辑蕴函(难点)

- **SQL**语言中没有蕴函**(Implication)**逻辑运算
- 可以利用谓词演算将逻辑蕴函谓词等价转换为：

$$p \rightarrow q \equiv \neg p \vee q$$

带有EXISTS谓词的子查询(续）

［例44］ 查询至少选修了学生95002选修的全部课程的学生号码。

解题思路：

- 用逻辑蕴函表达：查询学号为x的学生，对所有的课程y，只要95002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 “学生95002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：$(\forall y)\ p \rightarrow q$

带有EXISTS谓词的子查询(续）

- 等价变换：

$$(\forall y)p \rightarrow q \equiv \neg(\exists y(\neg(p \rightarrow q)))$$
$$\equiv \neg(\exists y(\neg(\neg p \vee q)))$$
$$\equiv \neg \exists y(p \wedge \neg q)$$

- 变换后语义：不存在这样的课程y，学生95002选修了y，而学生x没有选。

带有EXISTS谓词的子查询(续）

- 用NOT EXISTS谓词表示：

```
SELECT DISTINCT Sno
FROM SC SCX
WHERE NOT EXISTS
        (SELECT *
         FROM SC SCY
         WHERE SCY.Sno = ' 95002 '  AND
             NOT EXISTS
                 (SELECT *
                  FROM SC SCZ
                  WHERE SCZ.Sno=SCX.Sno AND
                                SCZ.Cno=SCY.Cno));
```

练习

- Student(S#,Sname,Sage,Ssex)
- --S# 学生编号,Sname 学生姓名,Sage 出生年月,Ssex 学生性别

  Course(C#,Cname,T#)
- --C# --课程编号,Cname 课程名称,T# 教师编号

  Teacher(T#,Tname)
- --T# 教师编号,Tname 教师姓名

  SC(S#,C#,score)
- --S# 学生编号,C# 课程编号,score 分

练习：

- 1、查询没学过“张三”老师讲授的任一门课程的学生姓名 。
- 2、查询至少有一门课与学号为“01”的同学所学相同的同学的信息 。
- 3、查询没有学全所有课程的同学的信息 。
- 4、查询学过编号为“01”但是没有学过编号为“02”的课程的同学的信息。

- 1、查询没学过“张三”老师讲授的任一门课程的学生姓名 。
- select student.Sname
- from student
- where student.S# not in
  (select distinct sc.S# from sc , course , teacher
- where sc.C# = course.C# and course.T# = teacher.T# and teacher.tname = '张三')

- 2、查询至少有一门课与学号为“01”的同学所学相同的同学的信息 。
- select Student.*
- from Student, SC
- where Student.S# = SC.S# and SC.C# in
- (select C# from SC
- where S# = '01') and Student.S# <> '01'

- 3、查询没有学全所有课程的同学的信息 。
- select Student.*
- from Student, SC
- where Student.S# = SC.S#
- group by SC.S#
- having count(C#) < (select count(C#)
- from Course)

- 4、查询学过编号为“01”但是没有学过编号为“02”的课程的同学的信息。
- select Student.* from Student, SC
- where Student.S# = SC.S# and SC.C# = '01'

  and Student.S# not in (Select SC_2.S#

  from SC SC_2

  where SC_2.C# = '02')

3.3 查　询

3.3.1 概述

3.3.2 单表查询

3.3.3 连接查询

3.3.4 嵌套查询

3.3.5 集合查询

3.3.6 小结

3.3.5 集合查询

标准SQL直接支持的集合操作种类

并操作(UNION)

一般商用数据库支持的集合操作种类

并操作(UNION)

交操作(INTERSECT)

差操作(MINUS)

1. 并操作

形式

<查询块>

UNION

<查询块>

- 参加**UNION**操作的各结果表的列数必须相同；对应项的数据类型也必须相同

并操作（续）

[例45] 查询计算机科学系的学生以及年龄不大于19岁的学生。

方法一：

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

并操作（续）

方法二：

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

并操作（续）

[例46] 查询选修了课程1或者选修了课程2的学生。

方法一：

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

并操作（续）

方法二：

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

并操作（续）

[例47] 设数据库中有一教师表Teacher(Tno, Tname,...)。查询学校中所有师生的姓名。

```
SELECT Sname
 FROM   Student
 UNION
 SELECT  Tname
 FROM   Teacher;
```

2. 交操作

标准SQL中没有提供集合交操作，但可用其他方法间接实现。

2. 交操作

[例48] 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

本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；
```

交操作（续）

[例49] 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

本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 ');
```

交操作（续）

[例50] 查询学生姓名与教师姓名的交集

本例实际上是查询学校中与教师同名的学生姓名

```
SELECT DISTINCT Sname
FROM Student
WHERE Sname IN
    (SELECT Tname
    FROM Teacher);
```

3. 差操作

标准SQL中没有提供集合差操作，但可用其他方法间接实现。

3. 差操作

[例51] 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。

本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52] 查询学生姓名与教师姓名的差集

本例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);
```

4. 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
- 任何情况下，ORDER BY子句只能出现在最后
- 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（续）

[例53] 错误写法

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

对集合操作结果的排序（续）

正确写法

```
SELECT *
FROM Student
WHERE Sdept= 'CS'
UNION
SELECT *
FROM Student
WHERE Sage<=19
ORDER BY 1;
```

3.3.6 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]] …

[WHERE <条件表达式>]

[GROUP BY <列名1>[，<列名2>] ...

[HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]

[，<列名2’> [ASC|DESC]] …];

目标列表达式

- **目标列表达式格式**

  (1) [<表名>.] *

  (2) [<表名>.]<属性列名表达式>[，[<表名>.]<属性列名表达式>] …

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

集函数格式

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

条件表达式格式

（1）

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

条件表达式格式

（2）

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

条件表达式格式

（3）

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

条件表达式格式

(4) <属性列名> [NOT] LIKE <匹配串>

(5) <属性列名> IS [NOT] NULL

(6) [NOT] EXISTS (SELECT语句)

条件表达式格式

（7）

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

思考题：

建3张表：学生表，选课表，课程表

查询：

- （1）查询选修了’JAVA’课程的学生姓名，年龄
- （2）查询选修了所有课程的学生姓名和年龄
- （3）查询没有选修所有课程的学生姓名和年龄