



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 (续1)

福州大学软件学院

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

3.4.2 连接查询



- ❖ 连接查询：同时涉及多个表的查询
- ❖ 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>**

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>**

3.4.2 连接查询



- ❖ 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，逐一查找满足连接件的元组，找到后就将表1中的第一个元组与该元组拼接起来，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

排序合并法(SORT-MERGE)



常用于=连接

- 首先按连接属性对表1和表2排序
- 对表1的第一个元组，从头开始扫描表2，顺序查找满足连接条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组。当遇到表2中第一条大于表1连接字段值的元组时，对表2的查询不再继续

排序合并法（续）



- 找到表1的第二条元组，然后从刚才的中断点处继续顺序扫描表2，查找满足连接条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组。直接遇到表2中大于表1连接字段值的元组时，对表2的查询不再继续
- 重复上述操作，直到表1或表2中的全部元组都处理完毕为止

索引连接(INDEX-JOIN)



- 对表2按连接字段建立索引
- 对表1中的每个元组，依次根据其连接字段值查询表2的索引，从中找到满足条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组

连接查询（续）



- 一、等值与非等值连接查询
- 二、自身连接
- 三、外连接
- 四、复合条件连接

一、等值与非等值连接查询



❖ 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。

Student表



| 学 号 Sno | 姓 名 Sname | 性 别 Ssex | 年 龄 Sage | 所 在 系 Sdept |
|------------|--------------|-------------|-------------|----------------|
| 200215121 | 李勇 | 男 | 20 | CS |
| 200215122 | 刘晨 | 女 | 19 | CS |
| 200215123 | 王敏 | 女 | 18 | MA |
| 200515125 | 张立 | 男 | 19 | IS |

Course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 | | 2 |
| 7 | PASCAL语言 | 6 | 4 |

SC表



| 学号 Sno | 课程号 Cno | 成绩 Grade |
|-----------|------------|-------------|
| 200215121 | 1 | 92 |
| 200215121 | 2 | 85 |
| 200215121 | 3 | 88 |
| 200215122 | 2 | 90 |
| 200215122 | 3 | 80 |

一、等值与非等值连接查询



[例33] 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 |
|-------------|-------|------|------|-------|-----------|-----|-------|
| 200215121 | 李勇 | 男 | 20 | CS | 200215121 | 1 | 92 |
| 200215121 | 李勇 | 男 | 20 | CS | 200215121 | 2 | 85 |
| 200215121 | 李勇 | 男 | 20 | CS | 200215121 | 3 | 88 |
| 200215122 | 刘晨 | 女 | 19 | CS | 200215122 | 2 | 90 |
| 200215122 | 刘晨 | 女 | 19 | CS | 200215122 | 3 | 80 |

等值与非等值连接查询（续）



❖ 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。
 - $\langle \text{表名1} \rangle . \langle \text{列名1} \rangle = \langle \text{表名2} \rangle . \langle \text{列名2} \rangle$
- SELECT语句不能直接实现自然连接

等值与非等值连接查询（续）



❖ 自然连接：

[例33] 查询每个学生及其选修课程的情况

[例34] 对[例33]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）



❖ 非等值连接

- 连接运算符不为 = 的连接操作

[<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）



- 一、等值与非等值连接查询
- 二、自身连接
- 三、外连接
- 四、复合条件连接

二、自身连接



- ❖ 自身连接：一个表与其自己进行连接
- ❖ 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

二、自身连接



[例35]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 | Pcno |
|-----|------|
| 1 | 7 |
| 3 | 5 |
| 5 | 6 |

连接查询（续）



- 一、等值与非等值连接查询
- 二、自身连接
- 三、外连接
- 四、复合条件连接

三、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

三、外连接



[例33] 查询每个学生及其选修课程的情况

[例 36] 改写[例33]

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

/*也可以使用USING来去掉结果中的重复值:

```
FROM Student LEFT OUT JOIN SC USING (Sno);*/
```

外连接（续）



执行结果：

| Student.Sno | Sname | Ssex | Sage | Sdept | Cno | Grade |
|-------------|-------|------|------|-------|------|-------|
| 200215121 | 李勇 | 男 | 20 | CS | 1 | 92 |
| 200215121 | 李勇 | 男 | 20 | CS | 2 | 85 |
| 200215121 | 李勇 | 男 | 20 | CS | 3 | 88 |
| 200215122 | 刘晨 | 女 | 19 | CS | 2 | 90 |
| 200215122 | 刘晨 | 女 | 19 | CS | 3 | 80 |
| 200215123 | 王敏 | 女 | 18 | MA | NULL | NULL |
| 200215125 | 张立 | 男 | 19 | IS | NULL | NULL |

外连接（续）



❖ 左外连接

- 列出左边关系（如本例**Student**）中所有的元组

❖ 右外连接

- 列出右边关系中所有的元组

连接查询（续）



- 一、等值与非等值连接查询
- 二、自身连接
- 三、外连接
- 四、复合条件连接

四、复合条件连接



❖ 复合条件连接：WHERE子句中含多个连接条件

[例37]查询选修2号课程且成绩在90分以上的所有学生

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

复合条件连接（续）



[例38]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;
```


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(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)



例:

```
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



- 一、 帶有IN谓词的子查询
- 二、 帶有比较运算符的子查询
- 三、 帶有ANY（**SOME**）或**ALL**谓词的子查询
- 四、 帶有**EXISTS**谓词的子查询

一、带有IN谓词的子查询



[例39] 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 |
|-----------|-------|-------|
| 200215121 | 李勇 | CS |
| 200215122 | 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。

DBMS求解该查询时也是分步去做的。

带有IN谓词的子查询（续）



[例39] 查询与“刘晨”在同一个系学习的学生。

用自身连接完成[例39]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EXISTS谓词的子查询(续)

带有IN谓词的子查询（续）



[例40] 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关系中找到
“信息系统”的课程号，为3号

带有IN谓词的子查询（续）



一种可能的结果：

| Sno | Sname |
|-------|-------|
| ----- | ----- |
| 95001 | 李勇 |
| 95002 | 刘晨 |

带有IN谓词的子查询（续）



[例40]查询选修了课程名为“信息系统”的学生学号和姓名

用连接查询实现[例40]

```
SELECT 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



- 一、带有**IN**谓词的子查询
- 二、带有比较运算符的子查询
- 三、带有**ANY** (**SOME**) 或**ALL**谓词的子查询
- 四、带有**EXISTS**谓词的子查询

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39] 查询与“刘晨”在同一个系学习的学生。

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39]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 ' )
= Sdept;
```

带有比较运算符的子查询（续）



[例41] 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:

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;
```

带有比较运算符的子查询（续）



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)

3.4.3 嵌套查询



- 一、带有**IN**谓词的子查询
- 二、带有比较运算符的子查询
- 三、带有**ANY (SOME)** 或**ALL**谓词的子查询
- 四、带有**EXISTS**谓词的子查询

三、带有ANY（SOME）或ALL谓词的子查询



谓词语义

- ANY: 任意一个值
- ALL: 所有值

带有ANY (SOME) 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 (SOME) 或ALL谓词的子查询 (续)



[例42] 查询其他系中比计算机科学系某一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. RDBMS 执行此查询时, 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)



[例42] 查询其他系中比计算机科学系某一学生年龄小的学生姓名和年龄

用聚集函数实现[例42]

```
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43] 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 (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3.5 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 |

带有ANY (SOME) 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

3.4.3 嵌套查询



- 一、 帶有**IN**谓词的子查询
- 二、 帶有比较运算符的子查询
- 三、 帶有**ANY** (**SOME**) 或**ALL**谓词的子查询
- 四、 帶有**EXISTS**谓词的子查询

带有EXISTS谓词的子查询(续)



❖ 1. EXISTS谓词

- 存在量词 \exists
- 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

带有EXISTS谓词的子查询(续)



❖ 1. EXISTS谓词

- 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)



❖ 2. 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44]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)



[例44]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
```

```
      WHERE Sno=Student.Sno AND Cno= ' 1 ' );
```

带有EXISTS谓词的子查询(续)



[例44]查询所有选修了1号课程的学生姓名。

- 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)



❖ 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45] 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)



例：[例39]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' );
```

一、带有IN谓词的子查询

带有EXISTS谓词的子查询(续)



❖ 相关子查询的效率

- 由于带EXISTS量词的相关子查询只关心内层查询是否有返回值，并不需要查具体值，因此其效率并不一定低于其他形式的查询。

带有EXISTS谓词的子查询(续)



❖ 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:

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

带有EXISTS谓词的子查询(续)



[例46] 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)



用EXISTS/NOT EXISTS实现逻辑蕴涵(难点)

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

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

带有EXISTS谓词的子查询(续)



[例47]查询至少选修了学生200215122选修的全部课程的学生号码。

解题思路：

- 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200215122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200215122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:

$$\begin{aligned}(\forall y)p \rightarrow q &\equiv \neg (\exists y (\neg(p \rightarrow q))) \\ &\equiv \neg (\exists y (\neg(\neg p \vee q))) \\ &\equiv \neg \exists y(p \wedge \neg q)\end{aligned}$$

- 变换后语义: 不存在这样的课程 y , 学生200215122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 = ' 200215122 ' AND
            NOT EXISTS
          (SELECT *
           FROM   SC SCZ
           WHERE SCZ.Sno=SCX.Sno AND
                SCZ.Cno=SCY.Cno));
```


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

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

1. 并操作



❖ 形式

<查询块>

UNION

<查询块>

1. 并操作



[例48] 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

1. 并操作



[例48] 查询计算机科学系的学生及年龄不大于19岁的学生。

方法二：

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

1. 并操作



[例49] 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 ';
```

1. 并操作



[例49] 查询选修了课程1或者选修了课程2的学生。

方法二：

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

1. 并操作



[例49_a] 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



[例50] 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
```

2. 交操作



❖ [例50] 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;
```

2. 交操作



[例51] 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 ';
```

2. 交操作



[例51]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 ');
```

2. 交操作



[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



[例52] 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;
```

3. 差操作



[例52] 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;
```

3. 差操作



[例52_a] 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;
```



对集合操作结果的排序（续）



[例54]

```
SELECT Sname, Sage, Sdept  
FROM Student
```

UNION ALL

```
SELECT H_Sname, H_Sage, H_Sdept  
FROM History_Student
```

ORDER BY 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

3.4.5 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>

[HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]

3.4.5 SELECT语句的一般格式



❖ 目标列表表达式

- 目标列表表达式格式

(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。

3.4.5 SELECT语句的一般格式



- 集函数格式

$\left\{ \begin{array}{l} \text{COUNT} \\ \text{SUM} \\ \text{AVG} \\ \text{MAX} \\ \text{MIN} \end{array} \right\} ([\text{DISTINCT}|\text{ALL}] \text{ <列名>})$

COUNT ([DISTINCT|ALL] *)

3.4.5 SELECT语句的一般格式



❖ 条件表达式格式

(1)

$$\langle \text{属性列名} \rangle \theta \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ [\text{ANY}|\text{ALL}] (\text{SELECT语句}) \end{array} \right\}$$

3.4.5 SELECT语句的一般格式



(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)
}

3.4.5 SELECT语句的一般格式



(3)
<属性列名> [NOT] IN {
 (<值1>[, <值2>] ...)
 (SELECT语句)}

3.4.5 SELECT语句的一般格式



- (4) <属性列名> [NOT] LIKE <匹配串>
- (5) <属性列名> IS [NOT] NULL
- (6) [NOT] EXISTS (SELECT语句)

3.4.5 SELECT语句的一般格式



(7)

$$\left\{ \begin{array}{c} \text{AND} \\ \text{OR} \end{array} \right\} \left\{ \begin{array}{c} \text{AND} \\ \text{OR} \end{array} \right\} \left\{ \begin{array}{c} \text{AND} \\ \text{OR} \end{array} \right\} \dots$$

<条件表达式> { <条件表达式> { <条件表达式> { ...

下课了。。。



追求



休息一会儿。。。



www.hesee.com