

数据库原理

软件工程系 刘慰

liuwei@nbu.edu.cn

连接查询

- 连接查询：

- 同时涉及到两个或两个以上的表的查询。

- 连接查询的分类：

- 等值连接

- 内连接（自然连接）

- 外连接（左外连接、右外连接与全外连接）

- 自身连接

- 复合条件连接

等值连接

□ 查询多张表：

- **SELECT ***

- **FROM Student, SC**

- 得到的结果是 Student 与 SC 这两个关系的笛卡尔积：
 $Student \times SC$

□ 连接运算的含义：

- 从两个关系的笛卡尔积中选出符合条件的元组。

- $Student \bowtie_{SC.Sno = Student.Sno} SC$



- $\sigma_{SC.Sno=Student.Sno}(Student \times SC)$



如何用 SQL
来表示？

等值连接

□ $Student \bowtie SC$ 对应的 SQL 代码：
 $SC.Sno = Student.Sno$

□ **SELECT ***

FROM Student, SC

WHERE Student.Sno = SC.Sno;

□ $\sigma_{SC.Sno=Student.Sno}(Student \times SC)$

□ 注意：

- 连接条件中各属性的值应该是可比的（同一数据类型）。
- 用于比较的字段名可以相同，也可以不同。

自然连接

- 用 SQL 来表示 $Student \bowtie SC$
- **SELECT** Student.Sno, Sname, Cno, Grade
FROM Student, SC
WHERE Student.Sno = SC.Sno;

为什么Sno前要加表名，
而其他属性前不用加？

内连接 (INNER JOIN)

▣ 格式：

▣ **SELECT** 列1, 列2, , 列n
FROM 表1 **[INNER] JOIN** 表2
ON 条件表达式;

内连接 (INNER JOIN)

▣ 用内连接查询每个学生的学号、姓名及其选修的课程的课号与成绩。

▣ **SELECT** Student.Sno, Sname, Cno, Grade
FROM Student **INNER JOIN** SC
ON Student.Sno = SC.Sno;

▣ 等价于：

▣ **SELECT** Student.Sno, Sname, Cno, Grade
FROM Student, SC
WHERE Student.Sno = SC.Sno;

内连接 (INNER JOIN)

▫ 用内连接查询每个学生的学号、姓名及其选修的课程的课号、课名与成绩。

```
SELECT Student.Sno, Sname, Course.Cno,  
        Cname, Grade  
FROM Student  
INNER JOIN SC ON Student.Sno = SC.Sno  
INNER JOIN Course ON SC.Cno =  
                    Course.Cno;
```

等价于： $Student \bowtie SC \bowtie Course$

内连接 (INNER JOIN)

□ 也等价于：

```
□ SELECT Student.Sno, Sname, Course.Cno,  
        Cname, Grade  
FROM Student, SC, Course  
WHERE Student.Sno = SC.Sno  
        AND SC.Cno = Course.Cno;
```

□ 从 Student、Course 与 SC 三张表的笛卡尔积中根据连接条件选择元组。

内连接（ INNER JOIN ）

▫ 用内连接查询选修了“数据结构”这门课程的学生学号、姓名及成绩。

```
SELECT Student.Sno, Sname, Grade
FROM Student
INNER JOIN SC ON Student.Sno = SC.Sno
INNER JOIN Course ON SC.Cno =
                Course.Cno
WHERE Cname = '数据结构'
```

外连接 (OUTER JOIN)

▣ 格式：

▣ **SELECT** 列1, 列2, , 列n
FROM 表1 [**LEFT|RIGHT|FULL**] **JOIN** 表2
ON 条件表达式;

左连接与右连接

▣ 查询每个学生的学号、姓名、选修的课程的课号与成绩，结果应包括尚未选课的学生。

```
▣ SELECT Student.Sno, Sname, Cno, Grade  
FROM Student LEFT JOIN SC  
ON Student.Sno = SC.Sno;
```

▣ 查询每门课程的课号、课名及选修了这门课的学生学号与成绩，结果应包括无任何学生选修的课程。

```
▣ SELECT SC.Cno, Cname, Sno, Grade  
FROM SC RIGHT JOIN Course  
ON SC.Cno = Course.Cno;
```

自身连接

- 一个表与其自身进行连接称为自身连接。
- 查询每一门课程的课名及其先修课的课名。
 - **SELECT** C1.Cname, C2.Cname
FROM Course C1 **INNER JOIN** Course C2
ON C1.Cpno = C2.Cno
- 等价于
 - **SELECT** C1.Cname, C2.Cname
FROM Course C1, Course C2
WHERE C1.Cpno = C2.Cno

自身连接

- 查询与刘晨在同一个系的学生的学号、姓名与所在的系。
 - **SELECT** S2.Sno, S2.name, S2.Sdept
FROM Student S1 **INNER JOIN** Student S2
ON S1.Sdept = S2.Sdept
WHERE S1.Sname = '刘晨'
AND S2.Sname <> '刘晨'

自身连接

- 查询每一门课程的课名及其先修课的课名。
 - **SELECT C1.Cname, C2.Cname**
FROM Course C1 INNER JOIN Course C2
ON C1.Cpno = C2.Cno;

多表的自身连接

▣ 查询每一门课程的课名及其先修课的先修课的课名。

```
▣ SELECT C1.Cname, C3.Cname  
   FROM Course C1 INNER JOIN Course C2  
   ON C1.Cpno = C2.Cno  
   INNER JOIN Course C3  
   ON C2.Cpno = C3.Cno
```


复合条件连接

▣ 查询选修了1号课程，且成绩在90分以上（含）的学生的学号与姓名。

```
▣ SELECT Student.Sno, Sname  
FROM Student, SC  
WHERE Student.Sno = SC.Sno  
AND SC.Cno = '1'  
AND SC.Grade >= 90;
```

嵌套查询

- 一个 **SELECT.....FROM.....WHERE** 结构是一个查询块。
 - 将一个查询块嵌套在一个 **WHERE** 或 **HAVING** 子句的条件中，称为嵌套查询。
- 格式：
 - **SELECT** 目标列表表达式1
FROM 表1
WHERE 列1 **IN**
 (**SELECT** 目标列表表达式2
 FROM 表2
 WHERE 条件)

带有 IN 谓词的子查询

▣ 查询与“刘晨”在同一个系学习的学生。

▣ 步骤：

▣ STEP1：查询“刘晨”所在的系。

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

▣ STEP2：查询在 X 系中学习的学生。

```
• SELECT Sdept  
  FROM Student  
 WHERE Sdept = X
```



结果用 X
来表示

带有 IN 谓词的子查询

▣ 步骤（续）

▣ STEP3：用子查询替换 X

- **SELECT** Sdept
FROM Student
WHERE Sdept =

(**SELECT** Sdept
FROM Student
WHERE Sname = '刘晨')

X

带有 IN 谓词的子查询

□ 查询既选修了1号课程又选修了2号课程的学生学号。

```
SELECT Sno  
FROM SC  
WHERE Cno = 1 AND Cno = 2;
```

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

□ 那么应该怎么办？

对吗？

带有 IN 谓词的子查询

□ 思路：

□ STEP1：选出所有选修了1号课程的学生学号。

```
• SELECT Sno  
  FROM SC  
 WHERE Cno = 1;
```

R

□ STEP2：在集合 *R* 中，选出所有选修了2号课程的学生学号。

```
• SELECT Sno  
  FROM SC  
 WHERE Sno IN R AND Cno = 2
```

为什么用 IN
而不是 = ?

带有 IN 谓词的子查询

▣ 思路（续）：

▣ STEP3：将 R 用相应的 SQL 语句替代。

- **SELECT Sno**

- FROM SC**

- WHERE Sno IN**

- (SELECT Sno**

- FROM SC**

- WHERE Cno = 1)**

- AND Cno = 2**


带有 IN 谓词的子查询

▣ 查询所有选修了1号课程的学生姓名。

▣ 步骤：

▣ STEP1：在 *SC* 中得到所有选修了1号课程的学生学号。

```
• SELECT Sno  
  FROM SC  
 WHERE Cno = 1;
```



R

▣ STEP2：在 *Student* 中，得到所有学号在 *R* 中的学生姓名。

```
• SELECT Sname  
  FROM Student  
 WHERE Sno IN R;
```


带有 IN 谓词的子查询

▣ 步骤（续）：

▣ STEP3：将 R 用相应的 SQL 语句替代。

- **SELECT** Sname
FROM Student
WHERE Sno **IN**
 (**SELECT** Sno
 FROM SC
 WHERE Cno = 1);

带有 IN 谓词的子查询

□ 等价于：

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

□ 或等价于：

```
□ SELECT Sname  
   FROM Student INNER JOIN SC  
         ON Student.Sno = SC.Sno  
         AND SC.Cno = 1;
```

带有 IN 谓词的子查询

▣ 查询选修了“数据结构”这门课的学生的学号。

▣ **SELECT Sno**

FROM SC

WHERE Cno IN

(SELECT Cno

FROM Course

WHERE Cname = ‘数据结构’);

带有 IN 谓词的子查询

▣ 查询选修了“数据结构”这门课的学生的姓名。

```
SELECT Sname
FROM Student
WHERE Sno IN
    (SELECT Sno
     FROM SC
     WHERE Cno IN
         (SELECT Cno
          FROM Course
          WHERE Cname = '数据结构'));
```

相关子查询

- 若子查询的查询条件不依赖于父查询，则这类子查询称为不相关子查询。
 - 不相关子查询可以脱离于外层查询独立地执行
- 若子查询的查询条件依赖于父查询，称为相关子查询。

相关子查询

▣ 查询每个学生超过他/她选修的所有课程的平均成绩的课程的课程号。

```
▣ SELECT Cno  
FROM SC  
WHERE Grade >= ?
```

相关子查询

▣ 查询每个学生超过他/她选修的所有课程的平均成绩的课程的课程号。

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

相关子查询

▣ 查询。

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


带有 ALL 谓词的子查询

▫ 查询所有非“数学”系中的，且年龄比“数学”系中所有学生年龄都要大的学生的学号、姓名与年龄。

▫ 步骤：

▫ STEP1：得到“数学”系所有学生的年龄的集合。

```
• SELECT Sage  
  FROM Student  
 WHERE Sdept = '数学';
```

R

带有 ALL 谓词的子查询

▣ 步骤（续）：

- ▣ STEP2：在 *Student* 中选取全体非数学系的，且其年龄比 *R* 中的所有学生的年龄都要大的学生的学号、姓名与年龄。

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

带有 ALL 谓词的子查询

- ALL 谓词：

- 表示子查询结果中的**所有值**。

- ALL 谓词与比较运算符的结合：

- = ALL：等于子查询结果中的所有值（在子查询结果只有一个值的时候才有意义）。

- < ALL：小于子查询结果中的所有值。

- > ALL：大于子查询结果中的所有值。

- <> ALL：不等于子查询结果中的所有（即每一个）值。

带有 ANY 谓词的子查询

▫ 查询所有非“数学”系中的，且年龄比“数学”系中某一个学生的年龄小的学生的姓名与学号。

```
SELECT Sno, Sname, Sage
FROM Student
WHERE Sage < ANY
  (SELECT Sage
   FROM Student
   WHERE Sdept = '数学');
```

带有 ANY 谓词的子查询

- ANY 谓词：

- 表示子查询结果中的某一个值。

- ANY 谓词与比较运算符的结合：

- = ANY：等于子查询结果中的某一个值。

- < ANY：小于子查询结果中的某一个值。

- > ANY：大于子查询结果中的某一个值。

- <> ANY：不等于子查询结果中的某一个值。

带有 ANY 或 ALL 谓词的子查询

- ▣ 带 ANY 或 ALL 谓词的子查询也可以用聚集函数来实现：
- ▣ 查询所有非“数学”系中的，且年龄比“数学”系中所有学生年龄都要大的学生的学号、姓名与年龄。
- ▣ 步骤：
 - ▣ STEP1：得到“数学”系学生的最大年龄。
 - `SELECT MAX(Sage)`
`FROM Student`
`WHERE Sdept = ‘数学’;`

带有 ANY 或 ALL 谓词的子查询

▣ 步骤（续）：

- ▣ STEP2：得到所有非“数学”系，且年龄比这个最大年龄还要大的学生的学号、姓名与年龄。

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

带有 ANY 或 ALL 谓词的子查询

▫ 查询所有非“数学”系中的，且年龄比“数学”系中某一个学生的年龄小的学生的姓名与学号。

```
SELECT Sno, Sname, Sage
FROM Student
WHERE Sage <
    (SELECT MAX(Sage)
     FROM Student
     WHERE Sdept = '数学');
```


带有 ANY 或 ALL 谓词的子查询

▣ ANY , ALL 谓词与聚集函数、IN 谓词的等价转换关系：


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

推荐用 SOME 谓词替代 ANY 谓词

□ 由于 ANY 谓词在语义上的二义性，现在推荐使用 SOME 谓词。
这两者是完全等效的。

□ 对于 SQL 语句：

- `SELECT Sname FROM Student
WHERE Sage < ANY (SELECT Sage FROM Student
WHERE Sdept = 'CS')`
- `SELECT Sname FROM Student
WHERE Sage < SOME (SELECT Sage FROM Student
WHERE Sdept = 'CS')`
- `SELECT Sname FROM Student
WHERE Sage < ALL (SELECT Sage FROM Student
WHERE Sdept = 'CS')`



使用 ANY
和使用
SOME 哪
个语义更
清楚？

习题

- ▣ 找出所有学生中年龄最小的那个学生的姓名。
 - ▣ 使用聚集函数
 - ▣ 不使用聚集函数
- ▣ 找出 所有课程都不及格的学生的姓名
 - ▣ 使用聚集函数
 - ▣ 不使用聚集函数
- ▣ 找出 “李勇” 成绩最低的那门课的课号。
 - ▣ 使用聚集函数
 - ▣ 不使用聚集函数