

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(3)





3.4 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 语句的一般格式**
-



3.4.2 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: =、>、<、>=、<=、!=

- 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 (交叉连接)

- 又称非限制连接，它将两个表不加任何约束地组合在一起，也就是将第一个表中的所有记录分别与第二个表的所有记录组成新的记录。（进行广义笛卡尔乘积）
 - 不带连接谓词的连接
 - 很少使用
 - 例： **SELECT Student.* , SC.***
 - **FROM Student, SC**
-



二、等值与非等值连接查询

等值连接、自然连接、非等值连接

[例] 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

- 连接运算符为 $=$ 的连接操作
 - [\langle 表名 1 \rangle .] \langle 列名 1 \rangle $=$ [\langle 表名 2 \rangle .] \langle 列名 2 \rangle
 - 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。
-



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]

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

等价于内连接

```
Select S.Sno , Sname , Ssex , Sage ,  
       Sdept , Cno , Grade  
From s Inner join sc on s.sno=sc.sno
```



非等值连接查询

连接运算符 不是 $=$ 的连接操作

$[< \text{表名 } 1>.]< \text{列名 } 1> < \text{比较运算符} > [< \text{表名 } 2>.]< \text{列名 } 2>$

比较运算符: $>$ 、 $<$ 、 $>=$ 、 $<=$ 、 \neq



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] 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

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



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





连接种类 (P59)

- 左外连接

列出左边关系中所有的元组，
并在右表的相应列中填上 **NULL** 值。

- 右外连接

列出右边关系中所有的元组，
并在左表的相应列中填上 **NULL** 值。



外连接（续）

[例] 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** sno,sname,ssex,sage,
 - **sdept,cno,grade**
 - **from s left join sc**
 - **on s.sno=sc.sno ;**
-



外连接（续）

结果：

S.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	NULL	NULL
95004	张立	男		19	IS	NULL	NULL



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] 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 S.Sno, S.Sname  
FROM   S, SC  
WHERE  S.Sno = SC.Sno  
       AND  SC.Cno= ' 2 '  
       AND  SC.Grade > 90 ;
```



多表连接

[例] 查询每个学生的学号、姓名、选修的课程名及成绩。

```
SELECT S.Sno , Sname , Cname , Grade  
FROM S , SC , C  
WHERE S.Sno = SC.Sno  
and SC.Cno = C.Cno ;
```

结果:

S.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



3.4 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** 语句的一般格式
-



3.4.3 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
FROM Student
WHERE Sno IN**

外层查询 / 父查询

(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



例 查询与刘晨在一个系学习的学生

- 解法 1
- **select sno,sname,sdept**
- **from s**
- **where sdept in**
- **(select sdept**
- **from s**
- **where sname = '刘晨')**
- 当子查询的结果唯一时， **in** 可以用 **=** 替代。



不相关子查询



例 解法 2

- 查询与刘晨在一个系学习的学生
 - **select s1.sno,s1.sname,s1.sdept**
 - **from s s1,s s2**
 - **where s1.sdept = s2.sdept**
 - **and s2.sname = ' 刘晨 '**
-



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，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。

- 相关子查询

子查询的查询条件依赖于父查询



-
- **[例]** 查询所有选修了 1 号课程的学生姓名。
 - 用嵌套查询
 - **SELECT Sname**
 - **FROM S**
 - **WHERE EXISTS**
 - **(SELECT * /* 相关子查询 */**
 - **FROM SC**
 - **WHERE Sno=S.Sno AND Cno= '1') ;**
-



引出子查询的谓词

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

```
SELECT Sno , Sname , Sdept
FROM S
WHERE Sdept IN
(SELECT Sdept
FROM S
WHERE 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 S S1

WHERE S1.Sdept IN

(SELECT Sdept

FROM S S2

WHERE S2.Sname= ' 刘晨 ');



[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 S  
WHERE Sno IN  
    (SELECT Sno  
        FROM SC  
        WHERE Cno IN  
            (SELECT Cno  
                FROM C  
                WHERE Cname= ‘ 信息系统’ ));
```

③ 最后在 **S** 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 **C** 关系中找到“信
息系统”的课程号，结果为 3 号



带有 IN 谓词的子查询（续）

– 用连接查询

SELECT Sno , Sname

FROM S , SC , C

WHERE S.Sno = SC.Sno AND

SC.Cno = Course.Cno AND

C.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，则可以用 **=** 代替 **IN**：

SELECT Sno , Sname , Sdept

FROM S

WHERE Sdept =

(SELECT Sdept

FROM S

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 S
WHERE ( SELECT Sdept
        FROM S
        WHERE Sname= ' 刘晨 ' )
      = Sdept ;
```



三、带有 **ANY** 或 **ALL** 谓词的子查询

谓词语义

- **ANY** : 任意一个值
- **ALL** : 所有值



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 41] 查询其他系中比信息系任意一个（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
-



帶有 **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，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
- **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



帶有 ANY 或 ALL 谓词的子查询（续）

[例] 查询其他系中比信息系任意一个（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 <
      (SELECT MAX(Sage)
       FROM Student
       WHERE Sdept= ' IS ')
AND Sdept <> ' IS ';
```

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



帶有 ANY 或 ALL 谓词的子查询（续）

[例] 查询其他系中比计算机系**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 ';
```

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** 实现逻辑蕴涵
-



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，其目标列表表达式通常都用 * ，带 EXISTS 的子查询只返回真值或假值
 - 2. NOT EXISTS 谓词
-



带有 EXISTS 谓词的子查询 (续)

[例] 查询所有选修了 1 号课程的学生姓名。

- 用嵌套查询

```
SELECT Sname
```

```
FROM S
```

```
WHERE EXISTS
```

```
(SELECT * /* 相关子查询 */
```

```
FROM SC
```

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

- 用连接运算

```
SELECT Sname
```

```
FROM S, SC
```

```
WHERE S.Sno=SC.Sno AND SC.Cno= '1' ;
```



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 S
```

```
WHERE NOT EXISTS
```

```
    ( SELECT *
```

```
      FROM SC
```

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



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：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 ' ) ;
```

Page111 中例 46 ， 例 47 （ exists 的复杂应用）



3.4 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 语句的一般格式
-



3.4.4 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 1 >

UNION

< 查询块 2 >

- 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】 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 ;
```



并操作（续）

方法二：

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



并操作（续）

[例] 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] 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

- 形式

< 查询块 1>

INTERSECT

< 查询块 2>

- 参加 **INTERSECT** 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
-



2. 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

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

复合条件查询

SELECT *

FROM Student

WHERE Sdept= 'CS' AND Sage<=19 ;



交操作（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

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

复合条件查询

(SELECT Sno

FROM SC

WHERE Cno=' 2 ') ;



交操作（续）

[例] 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);

嵌套查询



3. 差操作 (except)

标准 **SQL** 中没有提供集合差操作，但可用

其他方法间接实现。（商用数据库支持）

Select * from student where Sdept='CS'

EXCEPT

Select * from student where Sage<=19



3. 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。 $A-B=\{ \dots \}$

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】 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** 子句中可以用数字指定排序属性
-



对集合操作结果的排序（续）

【例】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]

< 目标列表表达式 > [别名] [, < 目标列表表达式 > [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 1' >] ...

[**HAVING** < 条件表达式 >]

[**ORDER BY** < 列名 2 > [ASC|DESC]

[, < 列名 2' > [ASC|DESC]] ...] ;



目标列表表达式

- 目标列表表达式格式

(1) [\langle 表名 \rangle .] *

(2) [\langle 表名 \rangle .] \langle 属性列名表达式 \rangle [, [\langle 表名 \rangle .] \langle 属性列名表达式 \rangle] ...

\langle 属性列名表达式 \rangle : 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
MAX
MIN

([DISTINCT|ALL] < 列名 >)

COUNT ([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) }**



(2)

列名 >

< 属性列名 > [NOT] BETWEEN { < 属性列名 >
< 常量 > } AND { < 属性
< 常量 > }

(SELECT

语句)

语句)

语



条件表达式格式

(3)

1>[, < 值 2>] ...)

< 属性列名 > [NOT] IN

(SELECT 语句)

(< 值)



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

