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

刘 淇
Email: qiliuql@ustc.edu.cn

课程主页：
http://staff.ustc.edu.cn/~qiliuql/DB2020HF.html

第三章 关系数据库标准语言SQL

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

3.5 数据更新

3.6 视图

3.7 小结

数据查询

- 语句格式

SELECT [ALL | DISTINCT] <目标列表达式>

[，<目标列表达式>] ...

FROM <表名或视图名>[， <表名或视图名>] ...

[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 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

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 **ORDER BY**子句
  - 四、 聚集函数
  - 五、 **GROUP BY**子句

一、选择表中的若干列

- 查询指定列

[例1] 查询全体学生的学号与姓名。

```
SELECT Sno，Sname
FROM Student;
```

```
1 • SELECT Sno, Sname
2       FROM Student;
```

Result Grid | Filter Rows: | Edit:

Sno	Sname
200215126	张成民
200215129	陈春

[例2] 查询全体学生的姓名、学号、所在系。

```
SELECT Sname，Sno，Sdept
FROM Student;
```

类似于关系代数中的哪一个操作?

2. 查询全部列

- 选出所有属性列：
  - 在**SELECT**关键字后面列出所有列名
  - 将<目标列表达式>指定为 *

[例3] 查询全体学生的详细记录。

```
SELECT Sno，Sname，Ssex，Sage，Sdept
FROM Student;
```

或

```
SELECT *
FROM Student;
```

3. 查询经过计算的值

- **SELECT子句的<目标列表达式>可以为：**
  - 算术表达式
  - 字符串常量
  - 函数
  - 列别名

查询经过计算的值（续）

[例4] 查全体学生的姓名及其出生年份。

```
SELECT Sname，2004-Sage   /*假定当年的年份为2004年*/
FROM Student;
```

输出结果：

Sname	2004-Sage
李勇	1984
刘晨	1985
王敏	1986
张立	1985

查询经过计算的值（续）

[例5] 查询全体学生的姓名、出生年份和所有系，要求用小写字母表示所有系名

```
SELECT Sname，  'Year of Birth: '，2004-Sage，
               LOWER(Sdept)
FROM Student;
```

输出结果：

Sname	'Year of Birth:'	2004-Sage	LOWER(Sdept)
李勇	Year of Birth:	1984	cs
刘晨	Year of Birth:	1985	is
王敏	Year of Birth:	1986	ma
张立	Year of Birth:	1985	is

查询经过计算的值（续）

- 使用列别名改变查询结果的列标题:

```
SELECT Sname NAME，'Year of Birth: ' BIRTH，
       2000-Sage BIRTHDAY，LOWER(Sdept) DEPARTMENT
FROM Student;
```

输出结果:

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1984	cs
刘晨	Year of Birth:	1985	is
王敏	Year of Birth:	1986	ma
张立	Year of Birth:	1985	is

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 **ORDER BY**子句
  - 四、 聚集函数
  - 五、 **GROUP BY**子句

二、选择表中的若干元组

- **1. 消除取值重复的行**

  如果没有指定DISTINCT关键词，则缺省为ALL

  [例6] 查询选修了课程的学生学号。

  SELECT Sno FROM SC;

  等价于：

  SELECT ALL Sno FROM SC;

  执行上面的SELECT语句后，结果为：

Sno
200215121
200215121
200215121
200215122
200215122

消除取值重复的行（续）

- 指定DISTINCT关键词，去掉表中重复的行

```
SELECT DISTINCT Sno
FROM SC;
```

执行结果：

Sno
200215121
200215122

2.查询满足条件的元组

表3.4 常用的查询条件

查 询 条 件	谓　词
比　较	=，>，<，>=，<=，!=，<>，!>，!<；NOT+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空　值	IS NULL，IS NOT NULL
多重条件（逻辑运算）	AND，OR，NOT

(1) 比较大小

[例7] 查询计算机科学系全体学生的名单。

```
SELECT Sname
FROM Student
WHERE Sdept='CS';     # WHERE Sdept="CS";
```

[例8] 查询所有年龄在20岁以下的学生姓名及其年龄。

```
SELECT Sname，Sage
FROM    Student
WHERE Sage < 20;
```

[例9] 查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT Sno
FROM  SC
WHERE Grade<60;
```

（2）确定范围

- 谓词：**BETWEEN … AND …**

  NOT BETWEEN … AND …

[例10] 查询年龄在20~23岁（包括20岁和23岁）之间的学生的姓名、系别和年龄

```
SELECT Sname，Sdept，Sage
FROM   Student
WHERE  Sage BETWEEN 20 AND 23;
```

[例11] 查询年龄不在20~23岁之间的学生姓名、系别和年龄

```
SELECT Sname，Sdept，Sage
FROM   Student
WHERE Sage NOT BETWEEN 20 AND 23;
```

(3) 确定集合

- 谓词：IN <值表>，NOT IN <值表>　　<值表>是一个集合

[例12]查询信息系（IS）、数学系（MA）和计算机科学系（CS）学生的姓名和性别。

```
SELECT Sname，Ssex
FROM  Student
WHERE Sdept IN ( 'IS'，'MA'，'CS' ); #("IS", "MA", "CS");
```

[例13]查询既不是信息系、数学系，也不是计算机科学系的学生的姓名和性别。

```
SELECT Sname，Ssex
FROM Student
WHERE Sdept NOT IN ( 'IS'，'MA'，'CS' );
```

(4)字符匹配

- 谓词： [NOT] LIKE ‘<匹配串>’ [ESCAPE ‘ <换码字符>’]

1) 匹配串为固定字符串

[例14] 查询学号为200215121的学生的详细情况。

```
SELECT *
FROM  Student
WHERE  Sno LIKE ‘200215121’;
```

等价于：

```
SELECT  *
FROM  Student
WHERE Sno = ' 200215121 ';
```

字符匹配（续）

2) 匹配串为含通配符的字符串

%：任意多个字符；_：单个字符（汉字为2个字符）

[例15] 查询所有姓刘学生的姓名、学号和性别。

```
SELECT Sname，Sno，Ssex
FROM Student
WHERE  Sname LIKE ‘刘%’;
```

[例16] 查询姓"欧阳"且全名为三个汉字的学生的姓名。

```
SELECT Sname
FROM   Student
WHERE  Sname LIKE '欧阳__';
```

字符匹配（续）

[例17] 查询名字中第2个字为"阳"字的学生的姓名和学号。

```
SELECT Sname，Sno
FROM Student
WHERE Sname LIKE '__阳%';
```

[例18] 查询所有不姓刘的学生姓名。

```
SELECT Sname，Sno，Ssex
FROM Student
WHERE Sname NOT LIKE '刘%';
```

字符匹配（续）

3) 使用换码字符ESCAPE '\'将通配符转义为普通字符

[例19] 查询DB_Design课程的课程号和学分。

```
SELECT Cno，Ccredit
FROM Course
WHERE Cname LIKE 'DB\_Design' ESCAPE '\';
```

```
WHERE Cname LIKE "DB/_Design" ESCAPE "/";
```

[例20] 查询以"DB_"开头，且倒数第3个字符为 i的课程的详细情况。

```
SELECT  *
FROM   Course
WHERE  Cname LIKE  'DB\_%i_ _' ESCAPE ' \ ';
```

ESCAPE '\' 表示“ \ ” 为换码字符

实例

SELECT Cno FROM Course WHERE Cname LIKE “DB_Design” ;

SELECT Cno FROM Course WHERE Cname LIKE “DB_Design”;

SELECT Cno FROM Course WHERE Cname LIKE "DB/_Design" escape "/";

结果分别是什么？

Cno	Cname	Cpno	Ccredit
1	DB_Design	1	NULL
2	DBEDesign	2	NULL
NULL	NULL	NULL	NULL

#right 能找到DB_Design
#right,这里_表示通配符，它能找出来DB_Design 和 DBEDesign
#right,这里_不表示通配符，它能找出来DB_Design

(5) 涉及空值的查询

- 谓词： IS NULL 或 IS NOT NULL
- “IS” 不能用 “=” 代替

[例21] 某些学生选修课程后没有参加考试，所以有选课记录，但没有考试成绩。查询缺少成绩的学生的学号和相应的课程号。

```
SELECT Sno，Cno
FROM  SC
WHERE  Grade IS NULL
```

[例22] 查所有有成绩的学生学号和课程号。

```
SELECT Sno，Cno
FROM  SC
WHERE  Grade IS NOT NULL;
```

(6) 多重条件查询

- 逻辑运算符：**AND**和 **OR**来联结多个查询条件
  - **AND的优先级高于OR**
  - 可以用括号改变优先级
- 可用来实现多种其他谓词
  - **[NOT] IN**
  - **[NOT] BETWEEN … AND …**

多重条件查询（续）

[例23] 查询计算机系年龄在20岁以下的学生姓名。

```
SELECT Sname
FROM  Student
WHERE Sdept= 'CS' AND Sage<20;
```

多重条件查询（续）

- 改写[例12]

[例12] 查询信息系（IS）、数学系（MA）和计算机科学系（CS）学生的姓名和性别。

```
SELECT Sname，Ssex
FROM Student
WHERE Sdept IN ( 'IS'，'MA'，'CS' )
```

可改写为：

```
SELECT Sname，Ssex
FROM   Student
WHERE  Sdept= ' IS ' OR Sdept= ' MA' OR Sdept= ' CS ';
```

练习

- 在Select字句中，关键字（）用于消除重复项
  - A. AS　　B. DISTINCT　　C. TOP　D. PERCENT
- 要用模糊查询从数据库中查找与某一数据相关的所有元组，可以用（）关键字。
  - A. AND　　B. OR　　C. ALL　D. Like
- 在SQL中，下列涉及空值的操作，不正确的是（　　）。
  - A．AGE IS NULL　　B．AGE IS NOT NULL
  - C．AGE = NULL　　D．NOT (AGE IS NULL)

练习

关系 R、S 如下表所示，

$R \div (\pi_{A1,A2}(\sigma_{1<3}(S)))$ 的结果为__

R关系		
A1	A2	A3
a	b	c
b	a	d
c	d	d
d	f	g

S关系		
A1	A2	A4
a	z	a
b	a	h
c	d	d
d	s	c

作业-7 关系代数与SQL(Part 1)

- 设有如下关系表R、S和T
  - R(BH, XM, XB, DWH)
  - S(DWH, DWM)
  - T(BH, XM, XB, DWH)
- 写出实现下列关系代数的SQL语句：

1） $\sigma_{DWH='100'}(R)$

2） $\prod_{XM,XB}(R)$

3） $\prod_{XM,DWH}(\sigma_{XB='女'}(R))$

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 ORDER BY子句
  - 四、 聚集函数
  - 五、 **GROUP BY子句**

三、ORDER BY子句

- **ORDER BY**子句（对结果进行排序）
  - 可以按一个或多个属性列排序
  - 升序：**ASC**；降序：**DESC**；缺省值为升序
- 当排序列含空值时
  - **ASC**：排序列为空值的元组最后显示
  - **DESC**：排序列为空值的元组最先显示

ORDER BY子句（续）

[例24] 查询选修了3号课程的学生的学号及其成绩，查询结果按分数降序排列。

```
SELECT Sno，Grade
FROM  SC
WHERE  Cno= ' 3 '
ORDER BY Grade DESC;
```

[例25] 查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列。

```
SELECT  *
FROM  Student
ORDER BY Sdept，Sage DESC;
```

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 **ORDER BY**子句
  - 四、 聚集函数
  - 五、 **GROUP BY**子句

四、聚集函数

- 聚集函数：
  - 计数

    COUNT（[DISTINCT|ALL] *）

    COUNT（[DISTINCT|ALL] <列名>）
  - 计算总和

    SUM（[DISTINCT|ALL] <列名>）
  - 计算平均值

    AVG（[DISTINCT|ALL] <列名>）
  - 最大最小值

    MAX（[DISTINCT|ALL] <列名>）

    MIN（[DISTINCT|ALL] <列名>）

聚集函数（续）

[例26] 查询学生总人数。

```
SELECT COUNT(*)
FROM  Student;
```

[例27] 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT Sno)
FROM SC;
```

[例28] 计算1号课程的学生平均成绩。

```
SELECT AVG(Grade)
FROM SC
WHERE Cno= ' 1 ';
```

聚集函数（续）

[例29] 查询选修1号课程的学生最高分数。

```
SELECT MAX(Grade)
FROM SC
WHER Cno=‘1’;
```

［例30］查询学生200215012选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM  SC，  Course
WHERE Sno='200215012' AND
SC.Cno=Course.Cno;
```

3.4.1 单表查询

- 查询仅涉及一个表：
  - 一、 选择表中的若干列
  - 二、 选择表中的若干元组
  - 三、 **ORDER BY**子句
  - 四、 聚集函数
  - 五、 **GROUP BY**子句

五、GROUP BY子句

- **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

GROUP BY子句（续）

[例31] 求各个课程号及相应的选课人数。

```
SELECT Cno，COUNT(Sno)
FROM   SC
GROUP BY Cno;
```

查询结果：

Cno	COUNT(Sno)
1	22
2	34
3	44
4	33
5	48

GROUP BY子句（续）

[例32] 查询选修了3门以上课程的学生学号。

```
SELECT Sno
FROM  SC
GROUP BY Sno
HAVING  COUNT(*) >3;
```

GROUP BY子句（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。

练习

- 若SQL语句中的Order By短语指定了多个字段，则（）
  - A.无法排序　　B.只按第一个字段排序
  - C.按自左至右的字段顺序排序　　D.按自右至左的顺序排序
- 下列哪个函数不属于聚集函数（ ）。
  - A、count()　　B、avg()　　C、min()　　D、str()
- 聚集函数的返回值是（）。
  - A、一个标量值 B、一组值 C、表达式 D、表
- 在SQL查询语句中,GROUP BY语句用于( )，如果要求分组满足指定条件，则需要使用（ ）子句来限定分组。
  - A.选择行条件 B.对查询进行排序 C.列表 D.分组查询
  - E. Ordering　　F. Having　　G. Giving　　H. Holding

练习

- 下列各运算符中（ ）不属于逻辑运算符。
  - A、&　　B、not　　C、and　　D、or
- 下列哪条语句能够从学生表中查询出姓名的第二个字是“敏”的学生的信息（ ）。
  - A、select * from 学生表 where 姓名=’_敏%’
  - B、select * from 学生表 where 姓名 like ’_敏%’
  - C、select * from 学生表 where 姓名 like ’%敏%’
  - D、select * from 学生表 where 姓名 like ’%敏’
- 查询家庭地址与“上海”有关的记录应该用( )。
  - SELECT * FROM 家庭信息 WHERE 家庭地址 LIKE ’*上海*’
  - B. SELECT * FROM家庭信息WHERE家庭地址= ’%上海%’
  - C. SELECT * FROM家庭信息WHERE家庭地址LIKE ’?上海?’
  - D. SELECT * FROM家庭信息WHERE家庭地址LIKE ’%上海%’

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>
- [<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>
- 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

- 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=

[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：

[例34]查询每个学生及其选修课程的情况----用自然连接完成。

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

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

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

[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

自身连接（续）

- 查询每门课的间接先修课。
- *在同一个关系课程中进行连接，为加以区分引入别名。*
- **SELECT FIRST.Cno，SECOND.Cpno**
  FROM Course FIRST， Course SECOND
  WHERE FIRST. Cpno=SECOND. Cno；

Cno	Cpno
1	7
3	5
5	6

FIRST

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

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

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

[例 36]查询每个学生及其选修课程的情况

```
SELECT Student.Sno，Sname，Ssex，Sage，Sdept，Cno，Grade
FROM  Student  LEFT OUT JOIN SC ON (Student.Sno=SC.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
- 全外连接
  - 列出两边关系中所有的元组
- 不符合连接条件的元组的另一个关系的属性取空值

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;
```

作业-7 关系代数与SQL(Part 2)

- 设有如下关系表R、S和T：
  - R(BH, XM, XB, DWH)
  - S(DWH, DWM)
  - T(BH, XM, XB, DWH)
- 写出实现下列关系代数的SQL语句：

4) $R \infty S$

5) $\prod_{XM,XB,DWH}(\sigma_{XB='男'}(R \infty S))$

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)

查询选修2号课程的学生姓名。

```
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 ---根据情况而定
    - 子查询是可以用order by的，并且这个orderby 会影响主查询的结果顺序
- 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、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

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IS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。

返回

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

此查询为不相关子查询。

带有IN谓词的子查询（续）

用连接查询实现[例40]—练习

[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，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：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

练习

- 嵌套查询命令中的IN，相当于（）
  - A．等号＝　B．集合运算符 ∈
    C．加号+　　D．减号－
- 下面关于SELECT嵌套语句的叙述中，错误的是（）。
  - A）首先应对子查询求值
  - B）外部查询依赖于子查询的求值结果
  - C）子查询必须被括在圆括号中
  - D）子查询的结果会被显示出来

作业-8 用SQL实现简单查询

- 学生选课数据库由以下三个关系模式组成：
  - 学生关系 S（Sno，SN，Sdept，Age）
  - 课程关系 C（Cno，CN，Teacher）
  - 学生选课关系 SC（Sno，Cno，Grade）
- 关系模式中各属性含义是：Sno 学生号，SN 学生名，Sdept 学生所在系，Age 年龄，Cno 课程号，CN 课程名，Teacher 授课教师，Grade 成绩。
- **用SQL语言实现下列的操作：**
  - （1）查询“计算机系”年龄18岁以下的学生名单；
  - （2）查询选修‘数据库’课程且分数在90到100分之间的学生的学号、姓名、成绩，要求按成绩降序排序；
  - （3）按课程号统计各门课程的选课人数、最高分、最低分、平均分；
  - （4）将所有选修“数据库”课程的成绩增加10分（假设原成绩无超过90分的）。

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

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

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');
```

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(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：[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’);
```

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
                   )
        ）;
```

1.任意一个学生A

2.不存在一个课程a

3.在选课记录表里
不存在A对a的选课记录

不存在没有被选修的课程
（双重否定）

带有EXISTS谓词的子查询(续）

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：

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

p	q	$p \rightarrow q$	$\neg p \vee q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

Why?