数据库系统概论

Introduction to Database Systems

第3章 关系数据库标准语言SQL（续1）

中国人民大学信息学院

第3章 关系数据库标准语言SQL

3.1 SQL概述

3.2 数据定义

3.3 数据查询

3.4 数据更新

3.5 空值的处理

3.6 视图

本章小结

3.3 数据查询

❖语句格式

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]]] … | (<SELECT语句>) [AS] <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]]

LIMIT <行数1>[OFFSET <行数2>];

数据查询（续）

- SELECT语句：指定要显示的属性列
- FROM子句：指定查询对象（基本表或视图）
- WHERE子句：指定查询条件
- GROUP BY子句：结果按照<列名1>的值进行分组，该属性列值相等的元组为一个组，通常在每组中作用聚集函数。
- HAVING短语：只有满足指定条件的组才予以输出
- ORDER BY子句：对查询结果表按<列名1>的值的升序或降序排序
- LIMIT子句:限制SELECT语句查询结果的数量为<行数1>行，OFFSET <行数2>，表示在计算<行数1>行前忽略<行数2>行

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5 基于派生表的查询

3.3.1 单表查询

- 查询仅涉及一个表
  1. 选择表中的若干列
  2. 选择表中的若干元组
  3. ORDER BY子句
  4. 聚集函数
  5. GROUP BY子句
  6. LIMIT子句

1.选择表中的若干列

❖（1）查询指定列

[例3.16] 查询全体学生的学号与姓名

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

[例3.17] 查询全体学生的姓名、学号、主修专业。

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

选择表中的若干列（续）

❖（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3.18] 查询全体学生的详细记录

```
SELECT  Sno,Sname,Ssex,Sbirthdate,Smajor
FROM Student;
```

或

```
SELECT  *
FROM Student;
```

查询经过计算的值（续）

❖（3）查询经过计算的值

■ SELECT子句的<目标列表达式>可以是算术表达式、字符串常量、函数等

[例3.19] 查全体学生的姓名及其年龄

kingbaseES语句为：

```
SELECT Sname, (extract(year from current_date) - extract(year from Sbirthdate)) "年龄"
FROM Student;
```

输出结果：

Sname	年龄
李勇	21
刘晨	22
王敏	20
张立	21
陈新奇	20
赵明	21
王佳佳	20

查询经过计算的值（续）

[例3.20] 查询全体学生的姓名、出生日期和主修专业

```
SELECT Sname, 'Date of Birth:',Sbirthdate, Smajor
FROM Student;
```

输出结果：

Sname	Date of Birth：	Sbirthdate	Smajor
李勇	Date of Birth：	2000-3-8	信息安全
刘晨	Date of Birth：	1999-9-1	计算机科学与技术
王敏	Date of Birth：	2001-8-1	计算机科学与技术
张立	Date of Birth：	2000-1-8	计算机科学与技术
陈新奇	Date of Birth：	2001-11-1	信息管理与信息系统
赵明	Date of Birth：	2000-6-12	数据科学与大数据技术
王佳佳	Date of Birth：	2001-12-7	数据科学与大数据技术

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

2. 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3.21] 查询选修了课程的学生学号。

```
SELECT Sno
FROM SC;
```

等价于：

```
SELECT ALL  Sno
FROM SC;
```

消除取值重复的行（续）

- 该查询结果里包含了许多重复的行。如想去掉结果表中的重复行，必须指定DISTINCT：

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

执行结果：

Sno
20180001
20180002
20180003
20180004
20180005

（2）查询满足条件的元组

表3.5 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3.22] 查询主修计算机科学与技术专业全体学生的姓名

```
SELECT Sname
FROM   Student
WHERE  Smajor='计算机科学与技术';
```

[例3.23]查询所有2000年后（包括2000年）出生的学生姓名及其性别

```
SELECT Sname, Ssex
FROM   Student
WHERE  extract(year from Sbirthdate)>=2000;
       /*函数extract(year from Sbirthdate)从出生日期中抽取出年份*/
```

[例3.24]查询考试成绩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3.25] 查询年龄在20~23岁（包括20岁和23岁）之间的学生的学生的姓名、出生年月和主修专业

```
SELECT Sname, Sbirthdate, Smajor
FROM   Student
WHERE extract(year from current_date) - extract(year from
  Sbirthdate) BETWEEN 20 AND 23;
```

[例3.26]查询年龄不在20~23岁之间的学生姓名、出生年月和主修专业

```
SELECT Sname,Sbirthdate,Smajor
FROM Student
WHERE extract(year from current_date) - extract(year from Sbirthdate)
  NOT BETWEEN 20 AND 23;
```

③ 确定集合

❖谓词：IN <值表>，NOT IN <值表>

[例3.27] 查询计算机科学与技术专业和信息安全专业学生的姓名和性别

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor IN ( '计算机科学与技术','信息安全' );
```

[例3.28] 查询既不是计算机科学与技术专业也不是信息安全专业学生的姓名和性别

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor NOT IN ( '计算机科学与技术','信息安全' );
```

④ 字符匹配

❖ 谓词： [NOT] LIKE ‘<匹配串>’ [ESCAPE ‘ <换码字符>’]
<匹配串>：一个完整的字符串或含有通配符%和 _

- % （百分号）：任意长度（长度可以为0）的字符串
  - 例如a%b表示以a开头，以b结尾的任意长度的字符串
- _ （下横线）：任意单个字符。
  - 例如a_b表示以a开头，以b结尾的长度为3的任意字符串

字符匹配（续）

- 匹配串为固定字符串

[例3.29] 查询学号为20180003的学生的详细情况

```
SELECT *
FROM  Student
WHERE  Sno LIKE '20180003';
```

等价于：

```
SELECT  *
FROM  Student
WHERE Sno = '20180003';
```

字符匹配（续）

■ 匹配串为含通配符的字符串

[例3.30]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3.31]查询2018级学生的学号和姓名

```
SELECT Sno,Sname
FROM Student
WHERE Sno LIKE '2018%';        /*学号的数据类型是字符，用字符匹配*/
```

字符匹配（续）

[例3.32] 查询课程号为81开头，最后一位是6的课程名称和课程号

```
SELECT Cname,Cno
FROM Course
WHERE Cno LIKE '81_ _6';
```

/* 注意课程关系中课程号为固定长度，占5个字符大小 */

[例3.33] 查询所有不姓刘的学生姓名、学号和性别

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

字符匹配（续）

- 使用换码字符将通配符转义为普通字符

[例3.34] 查询DB_Design课程的课程号和学分

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

[例3.35] 查询以“DB_”开头，且倒数第三个字符为 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
- 第一个_前面有换码字符＼，被转义为普通的_字符
- i后面的两个_ 的前面均没有换码字符＼，仍作为通配符

⑤ 涉及空值的查询

❖谓词： IS NULL 或 IS NOT NULL

■ “IS” 不能用 “=” 代替

[例3.36]某些学生选修课程后没有参加考试，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;      /*分数Grade是空值*/
```

[例3.37] 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来连接多个查询条件

■ AND的优先级高于OR，可以用括号改变优先级

[例3.38] 查询主修计算机科学与技术专业2000年（包括2000年）以后出生的学生学号、姓名和性别。

```
SELECT Sno,Sname,Ssex
FROM Student
WHERE Smajor='计算机科学与技术' AND extract(year from Sbirthdate)>=2000;
```

例3.27中的IN谓词实际上是多个OR运算符的缩写：

```
SELECT Sname,Ssex
FROM Student
WHERE Smajor='计算机科学与技术' OR Smajor='信息安全';
```

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

3.ORDER BY子句

❖ORDER BY子句

- 可以按一个或多个属性列排序升序（ASC）降序（DESC）排列，默认值为升序
- 对于空值，排序时显示的次序由具体系统实现决定

ORDER BY子句（续）

[例3.39]查询选修了81003号课程的学生的学号及其成绩，查询结果按分数的降序排列

```
SELECT Sno,Grade
FROM SC
WHERE Cno='81003'
ORDER BY Grade DESC;
```

[例3.40]查询全体学生选修课程情况，查询结果先按照课程号升序排列，同一课程中按成绩降序排列。

```
SELECT *
FROM SC
ORDER BY Cno,grade DESC;        /*DESC需要明确写出来*/
```

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

4. 聚集函数

❖ 聚集函数：

- 统计元组个数

  COUNT(*)

- 统计一列中值的个数

  COUNT([DISTINCT|ALL] <列名>)

- 计算一列值的总和（此列必须为数值型）

  SUM([DISTINCT|ALL] <列名>)

- 计算一列值的平均值（此列必须为数值型）

  AVG([DISTINCT|ALL] <列名>)

- 求一列中的最大值和最小值

  MAX([DISTINCT|ALL] <列名>)

  MIN([DISTINCT|ALL] <列名>)

聚集函数（续）

[例3.41] 查询学生总人数

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

[例3.42] 查询选修了课程的学生人数

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

[例3.43] 计算选修81001号课程的学生平均成绩

```
SELECT AVG(Grade)
FROM    SC
WHERE Cno= ‘ 81001 ';
```

聚集函数（续）

[例3.44] 查询选修1号课程的学生最高分数

```
SELECT MAX(Grade)
FROM SC
WHERE Cno='81001';
```

[例3.45] 查询学号为20180003学生选修课程的总学分数

```
SELECT SUM(Ccredit)
FROM  SC,Course
WHERE Sno='20180003' AND SC.Cno=Course.Cno;
```

3.3.1 单表查询

❖查询仅涉及一个表：

1.选择表中的若干列

2.选择表中的若干元组

3.ORDER BY子句

4.聚集函数

5.GROUP BY子句

6.LIMIT子句

5. GROUP BY子句

- GROUP BY子句分组：
  - 按指定的一列或多列值分组，值相等的为一组
  - 如果未对查询结果分组，聚集函数将作用于整个查询结果
  - 分组后，聚集函数将作用于每一个组

GROUP BY子句（续）

[例3.46] 求各个课程号及选修该课程的人数

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

查询结果可能为：

Cno	COUNT(Sno)
81001	42
81002	44
81003	44
81004	33
81005	48
81006	45
81007	48
81008	39

GROUP BY子句（续）

[例3.47] 查询2019年第2学期选修了10门以上课程的学生学号

```
SELECT Sno
FROM  SC
WHERE Semester='20192'          /*先求出2019年第2学期选课的所有学生*/
GROUP BY Sno                    /*用GROUP BY子句按Sno进行分组*/
HAVING COUNT(*) >10;            /* 用聚集函数COUNT对每一组计数 */
```

GROUP BY子句（续）

[例3.48]查询平均成绩大于等于90分的学生学号和平均成绩

```
SELECT Sno,AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade)>=90;
```

下面的语句是错误的：

```
SELECT Sno,AVG(Grade)
FROM SC
WHERE AVG(Grade)>=90
GROUP BY Sno;
```

GROUP BY子句（续）

- HAVING短语与WHERE子句的区别：
  - 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。
- 参见爱课程网 数据库系统概论 数据查询节动画《GROUP BY子句》

6.LIMIT子句

LIMIT子句用于限制SELECT语句查询结果的（元组）数量

LIMIT <行数1>[OFFSET <行数2>];

- 语义是忽略前<行数2>行，然后取<行数1>作为查询结果数据
- OFFSET可以省略，代表不忽略任何行
- LIMIT子句经常和ORDER BY子句一起使用

LIMIT子句（续）

[例3.49]查询选修了数据库系统概论课程的成绩排名前10名的学生学号

```
SELECT Sno
FROM SC,Course
WHERE Course.Cname='数据库系统概论'
AND SC.Cno=Course.Cno
ORDER BY GRADE DESC
LIMIT 10;                    /*取前10行数据为查询结果*/
```

LIMIT子句（续）

[例3.50]查询平均成绩排名在3-5名的学生学号和平均成绩

```
SELECT Sno,AVG(Grade)
FROM SC
GROUP BY Sno
ORDER BY AVG(Grade) DESC
LIMIT 5 OFFSET 2;      /*取5行数据，忽略前2行，之后为查询结果数据*/
```

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

3.3.5 Select语句的一般形式

3.3.2 连接查询

- 连接查询：同时涉及两个以上的表的查询
- 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，但名字不必相同

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

1. 等值与非等值连接查询

❖等值连接：连接运算符为=

[例3.51]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
                    /* 将Student与SC中同一学生的元组连接起来 */
```

等值与非等值连接查询（续）

假设Student表、SC表的数据如第2章图2.1所示，本查询的执行结果如下图所示

Student.Sno	Sname	Ssex	Sbirthdate	Smajor	SC.Sno	Cno	Grade	Semester	Teachingclass
20180001	李勇	男	2000-3-8	信息安全	20180001	81001	85	20192	81001-01
20180001	李勇	男	2000-3-8	信息安全	20810001	81002	96	20201	81002-01
20180001	李勇	男	2000-3-8	信息安全	20810001	81003	87	20202	81003-01
20180002	刘晨	女	1999-9-1	计算机科学与技术	20180002	81001	80	20192	81001-01
20180002	刘晨	女	1999-9-1	计算机科学与技术	20180002	81002	98	20201	81002-01
20180002	刘晨	女	1999-9-1	计算机科学与技术	20810002	81003	71	20202	81003-01
...	...	...	...	...	...	...	...	...	...

连接操作的执行过程

- 首先在表Student中找到第一个元组，然后从头开始扫描SC表，逐一查找与Student第一个元组的Sno相等的SC元组，找到后就将Student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
- SC全部查找完后，再找Student中第二个元组，然后再从头开始扫描SC，逐一查找满足连接条件的元组，找到后就将Student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Student中的全部元组都处理完毕

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

2. 自然连接查询

若在等值连接中把目标列中重复的属性列去掉则为自然连接，

如[例3.52]中去掉了SC表中的Sno

Student

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
…	…	…	…	…

SC

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semeste	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
…	…	…	…	…

自然连接查询（续）

[例3.52]查询每个学生的学号、姓名、性别、出生日期、主修专业及该学生选修课程的课程号与成绩

```
SELECT Student.Sno,Sname,Ssex,Sbirthdate,Smajor,Cno,Grade
FROM  Student,SC
WHERE Student.Sno=SC.Sno;
```

- Sname，Ssex，Sbirthdate，Smajor，Cno和Grade属性列在Student表与SC表中唯一，引用时可以去掉表名前缀
- Sno在两个表都出现，因此SELECT子句和WHERE子句在引用时必须加上表名前缀

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

3.复合条件连接查询

复合条件连接：WHERE子句是由连接谓词和选择谓词组成的复合条件

[例3.53]查询选修81002号课程且成绩在90分以上的所有学生的学号和姓名

```
SELECT Student.Sno,Sname
FROM Student,SC
WHERE Student.Sno=SC.Sno  AND                    /*连接谓词 */
      SC.Cno='81002' AND SC.Grade>90;            /*其他选择条件*/
```

优化（高效）执行过程

- 先从SC中挑选出Cno='81002'并且Grade>90的元组形成一个中间关系
- 再和Student中满足连接条件的元组进行连接得到最终的结果关系

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

4. 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3.54]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 and SECOND.Cpno IS NOT NULL;
```

自身连接（续）

- 分析
  - 查询每一门课的间接先修课，必须先对一门课找到其直接先修课Cpno
  - 再按此先修课的课程号查找它的先修课程
  - 相当于将Course表与其自身连接后，取第一个副本的课程号与第二个副本的先修课号做为目标列中的属性。
  - 可以为Course表取两个别名：FIRST和SECOND
  - 这就是Course表与其自身连接

自身连接（续）

FIRST表（Course表）

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与C语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

SECOND表（Course表）

课程号 Cno	课程名 Cname	学分 Ccredit	先修课 Cpno
81001	程序设计基础与C语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002
81004	信息系统概论	4	81003
81005	操作系统	4	81001
81006	Python语言	3	81002
81007	离散数学	4	
81008	大数据技术概论	4	81003

自身连接（续）

查询结果：

Cno	Cpno
81003	81001
81004	81002
82006	81001
81008	81002

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

5. 外连接查询

- 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
  - 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

外连接查询（续）

[例3.55]想以Student表为主体列出每个学生的基本情况及其选课情况，若某个学生没有选课，则只输出其基本情况的数据，而把选课信息填为空值NULL

```
SELECT Student.Sno,Sname,Ssex,Sbirthdate,Smajor,Cno,Grade
FROM Student LEFT OUTER JOIN SC ON (Student.Sno=SC.Sno);
```

外连接查询（续）

执行结果如下：

Student.Sno	Sname	Ssex	Sbirthdate	Smajor	Cno	Grade
20180001	李勇	男	2000-3-8	信息安全	81001	85
20180001	李勇	男	2000-3-8	信息安全	81002	96
20180001	李勇	男	2000-3-8	信息安全	81003	87
20180002	刘晨	女	1999-9-1	计算机科学与技术	81001	80
20180002	刘晨	女	1999-9-1	计算机科学与技术	81002	98
20180002	刘晨	女	1999-9-1	计算机科学与技术	81003	71
20180003	王敏	女	2001-8-1	计算机科学与技术	81001	81
20180003	王敏	女	2001-8-1	计算机科学与技术	81002	76

连接查询（续）

1.等值与非等值连接查询

2.自然连接查询

3.复合条件连接查询

4.自身连接查询

5.外连接查询

6.多表连接查询

6. 多表连接查询

❖ 多表连接：两个以上的表进行连接

[例3.56]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;
```

❖ 可能的执行方式

- 先将Student表与SC表进行连接，得到每个学生的学号、姓名、所选课程号和相应的成绩
- 再将其与Course表进行连接，得到最终结果

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

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

```
SELECT Sname                          /*外层查询或父查询*/
FROM Student
WHERE Sno IN
        ( SELECT Sno                  /*内层查询或子查询*/
          FROM SC
          WHERE Cno= ‘ 81003 ');
```

嵌套查询（续）

- 上层的查询块称为外层查询或父查询
- 下层的查询块称为内层查询或子查询
- SQL语言允许多层嵌套查询
  - 即一个子查询中还可以嵌套其他子查询
- 子查询的限制
  - SELECT语句不能使用ORDER BY子句

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

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

1. 带有IN谓词的子查询

[例3.57]查询与“刘晨”在同一个主修专业的学生学号、姓名和主修专业

此查询要求可以分步来完成

① 确定“刘晨”所在系名

```
SELECT  Smajor
FROM    Student
WHERE  Sname= ' 刘晨 ';
```

结果为： 计算机科学与技术

带有IN谓词的子查询（续）

② 查找所有主修计算机科学与技术专业的学生

```
SELECT   Sno, Sname, Smajor
FROM     Student
WHERE    Smajor= ‘ 计算机科学与技术 ';
```

结果为：

Sno	Sname	Smajor
20180002	刘晨	计算机科学与技术
20180003	王敏	计算机科学与技术
20180004	张立	计算机科学与技术

带有IN谓词的子查询（续）

构造嵌套查询语句：

```
SELECT Sno,Sname,Smajor                    /*例3.57的解法一*/
FROM Student
WHERE Smajor IN
      (SELECT Smajor
       FROM Student
       WHERE Sname='刘晨');
```

■ 子查询的查询条件不依赖于父查询，称为不相关子查询

带有IN谓词的子查询（续）

用自身连接：

```
/*例3.57的解法二 */
SELECT S1.Sno,S1.Sname,S1.Smajor
FROM Student S1,Student S2
WHERE S1.Smajor=S2.Smajor AND S2.Sname='刘晨';
```

带有IN谓词的子查询（续）

[例3.58]查询选修了课程名为“信息系统概论”的学生的学号和姓名

```
SELECT Sno,Sname
FROM Student
WHERE Sno IN
    (SELECT Sno
     FROM SC
     WHERE Cno IN
         (SELECT Cno
          FROM Course
          WHERE Cname='信息系统概论'
         )
    );
```

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81004号课程的学生学号

①首先在Course关系中找出“信息系统概论”的课程号，结果为81004

带有IN谓词的子查询（续）

连接查询：

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

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

2. 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< >）。

[例3.57]由于一个学生只可能在一个系学习，则可以用 = 代替IN：

```
SELECT Sno,Sname,Smajor          /*例3.57的解法三 */
FROM   Student
WHERE Smajor  =
          (SELECT Smajor
           FROM   Student
           WHERE Sname= '刘晨');
```

带有比较运算符的子查询（续）

[例3.59]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（20180001）传送给内层查询。

```
SELECT AVG(Grade)
FROM SC y
WHERE y.Sno='20180001';
```

带有比较运算符的子查询（续）

- ②执行内层查询，得到值89.3（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 >=89.3;
```

带有比较运算符的子查询（续）

- ③执行这个查询，得到

  （20180001,81002）

然后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:

（20180001,81002）

（20180002,81002）

（20180003,81001）

（20180004,81003）

（20180005,81003）

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

带有ANY（SOME）或ALL谓词的子查询

使用ANY或ALL谓词时必须同时使用比较运算符

语义为：

> 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

带有ANY（SOME）或ALL谓词的子查询（续）

使用ANY或ALL谓词时必须同时使用比较运算符

语义为：

<= 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3.60] 查询非计算机科学技术专业中比计算机科学技术专业任意一个学生年龄小（出生日期晚）的学生的姓名、出生日期和主修专业

```
SELECT Sname,Sbirthdate, Smajor
FROM    Student
WHERE Sbirthdate > ANY (SELECT  Sbirthdate
                        FROM    Student
                        WHERE Smajor= '计算机科学与技术')
 AND Smajor <> '计算机科学与技术 ';          /*父查询块中的条件 */
```

带有ANY（SOME）或ALL谓词的子查询 （续）

结果：

Sname	Sbirthdate	Smajor
李勇	2000-3-8	信息安全
陈新奇	2001-11-1	信息管理与信息系统
赵明	2000-6-12	数据科学与大数据技术
王佳佳	2001-12-7	数据科学与大数据技术

执行过程：

- 首先处理子查询，找出计算机科学与技术专业中所有学生的出生日期，构成一个集合(1999-9-1,2001-8-1,2000-1-8)
- 处理父查询，找所有不是计算机科学与技术专业且年龄大于集合中任意一个的学生

带有ANY（SOME）或ALL谓词的子查询 （续）

用聚集函数实现[例3.60]

```
SELECT Sname,Sbirthdate, Smajor
FROM   Student
WHERE Sbirthdate >
                    (SELECT MIN(Sbirthdate)
                     FROM Student
                     WHERE Smajor= '计算机科学与技术')
 AND Smajor <>'计算机科学与技术';
```

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.61]查询非计算机科学与技术专业中比计算机科学与技术专业所有学生年龄都小（出生日期晚）的学生的姓名及出生日期。

```
SELECT Sname,Sbirthdate
FROM Student
WHERE Sbirthdate > ALL
                (SELECT Sbirthdate
                 FROM Student
                 WHERE Smajor='计算机科学与技术')
AND Smajor <> '计算机科学与技术';
```

带有ANY（SOME）或ALL谓词的子查询 （续）

- 首先处理子查询，找出计算机科学与技术专业中所有学生的年龄，构成一个集合(1999-9-1,2001-8-1,2000-1-8)
- 处理父查询，找所有不是计算机科学与技术专业且出生日期晚于集合中所有值的学生。

查询结果为：

Sname	Sbirthdate
陈新奇	2001-11-1
王佳佳	2001-12-7

带有ANY（SOME）或ALL谓词的子查询 （续）

用聚集函数实现：

```
SELECT Sname,Sbirthdate
FROM Student
WHERE Sbirthdate >
        (SELECT MAX(Sbirthdate)
                /*计算机科学与技术专业所有学生中最晚出生日期*/
         FROM Student
         WHERE Smajor='计算机科学与技术')
AND Smajor <>'计算机科学与技术';
```

带有ANY（SOME）或ALL谓词的子查询（续）

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

3.3.3 嵌套查询

1.带有IN谓词的子查询

2.带有比较运算符的子查询

3.带有ANY（SOME）或ALL谓词的子查询

4.带有EXISTS谓词的子查询

带有EXISTS谓词的子查询

- **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。

带有EXISTS谓词的子查询（续）

- **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3.62]查询所有选修了81001号课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE EXISTS
    (SELECT *
     FROM SC
     WHERE Sno=Student.Sno AND Cno='81001');
```

- 首先取外层查询中Student表的第一个元组，根据它与内层查询相关的属性值（Sno值）处理内层查询
- 若WHERE子句返回值为真，则取外层查询中该元组的Sname放入结果表
- 再取Student表的下一个元组
- 重复这一过程，直至外层Student表全部检查完为止

带有EXISTS谓词的子查询（续）

[例3.63]查询没有选修81001号课程的学生姓名

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
      (SELECT *
       FROM SC
       WHERE Sno=Student.Sno AND Cno='81001');
```

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：

    （∀x）P ≡ ¬（∃x（¬P））

带有EXISTS谓词的子查询（续）

查询与“刘晨”在同一个主修专业的学生

可以用带EXISTS谓词的子查询替换：

```
SELECT Sno,Sname,Smajor                    /*例3.57的解法四 */
FROM Student S1
WHERE EXISTS
    (SELECT *
     FROM Student S2
     WHERE S2.Smajor=S1.Smajor AND S2.Sname='刘晨');
```

带有EXISTS谓词的子查询（续）

[例3.64] 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3.65]查询至少选修了学生20180002选修的全部课程的学生的学号

可以用逻辑蕴涵来表达：查询学号为x的学生，对所有的课程y，只要20180002学生选修了课程y，则x也选修了y。形式化表示如下：

用p表示谓词“学生20180002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: （∀y） $p \rightarrow q$

带有EXISTS谓词的子查询（续）

- 等价变换：

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

- 表达的语义为：不存在这样的课程y，学生20180002选修了y，而学生x没有选修

带有EXISTS谓词的子查询（续）

```
SELECT Sno
FROM Student
WHERE NOT EXISTS
      (SELECT *                          /*这是一个相关子查询 */
       FROM SC SCX                       /*父查询和子查询均引用了SC表*/
       WHERE SCX.Sno='20180002' AND
             NOT EXISTS
             (SELECT *
              FROM SC SCY                /*用别名SCX、SCY将父查询*/
              WHERE SCY.Sno=Student.Sno AND   /*与子查询中的SC表区分开*/
                    SCY.Cno=SCX.Cno));
```

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

3.3.4 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
  - 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

集合查询（续）

[例3.66]查询计算机科学与技术专业的学生及年龄不大于19岁（包括等于19岁）的学生

```
SELECT * FROM Student WHERE Smajor='计算机科学与技术'
UNION
SELECT * FROM Student
WHERE (extract(year from current_date) - extract(year from Sbirthdate)) <=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

集合查询（续）

[例3.67]查询2020年第2学期选修了课程81001或者选修了课程81002的学生

```
SELECT Sno
FROM SC
WHERE Semester='20202' AND Cno='81001'
UNION
SELECT Sno
FROM SC
WHERE Semester='20202' AND Cno='81002';
```

集合查询（续）

[例3.68] 查询计算机科学与技术专业的学生与年龄不大于19岁的学生的交集

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术'
INTERSECT
SELECT *
FROM Student
WHERE(extract(year from current_date)-extract(year from Sbirthdate) ) <=19;
```

集合查询（续）

[例3.68] 实际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 Smajor='计算机科学与技术' AND
        (extract(year from current_date)-extract(year from Sbirthdate) )<=19;
```

集合查询（续）

[例3.69]查询既选修了课程81001又选修了课程81002的学生。就是查询选修课程810011的学生集合与选修课程81002的学生集合的交集。

```
SELECT Sno
FROM SC
WHERE Cno='81001'
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='81002';
```

集合查询（续）

本例也可以表示为

```
SELECT Sno
FROM SC
WHERE Cno='81001' AND Sno IN
            (SELECT Sno
             FROM SC
             WHERE Cno='81002');
```

集合查询（续）

[例3.70]查询计算机科学与技术专业的学生与年龄不大于19岁的学生的差集

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术'
EXCEPT
SELECT *
FROM Student
WHERE(extract(year from current_date)-extract(year from Sbirthdate) )<=19;
```

集合查询（续）

[例3.70] 就是查询计算机科学与技术专业中年龄大于19岁的学生

```
SELECT *
FROM Student
WHERE Smajor='计算机科学与技术' AND
      (extract(year from current_date)-extract(year from Sbirthdate) )>19；
```

3.3 数据查询

3.3.1 单表查询

3.3.2 连接查询

3.3.3 嵌套查询

3.3.4 集合查询

3.3.5基于派生表的查询

3.3.5 基于派生表的查询

- 子查询不仅可以出现在**WHERE**子句中，还可以出现在**FROM**子句中，子查询生成的临时派生表（**derived table**）成为主查询的查询对象

[例3.59]找出每个学生超过他自己选修课程平均成绩的课程号

```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade) FROM SC GROUP BY Sno)
        AS Avg_SC(Avg_sno,Avg_grade)
WHERE SC.Sno = Avg_SC.Avg_sno AND SC.Grade >= Avg_SC.Avg_grade;
```

基于派生表的查询（续）

❖ 如果子查询中没有聚集函数，派生表可以不指定属性列，子查询SELECT子句后面的列名为其缺省属性。

[例3.62]查询所有选修了81001号课程的学生姓名

```
SELECT Sname
FROM   Student,
       (SELECT Sno FROM SC WHERE Cno=‘ 81001 ') AS SC1
WHERE  Student.Sno=SC1.Sno;
```

- FROM子句生成派生表时，AS关键字可省略，必须为派生关系指定一个别名
- 派生表是一个中间结果表，查询完成后派生表将被系统自动清除

二维码3.3

SELECT语句的一般格式