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

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 空值的处理

3.7 视图

3.2 学生-课程 数据库

❖学生-课程模式 **SC**：

学生表：**Student(Sno,Sname,Ssex,Sage,Sdept)**

课程表：**Course(Cno, Cname, Cpno, Ccredit)**

学生选课表：**SC(Sno,Cno,Grade)**

Student-Course-SC表

学号 Sno	姓名 Sname	性别 sex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学	NULL	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	NULL	2
7	PASCAL语言	6	4

学 号 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

第三章 关系数据库标准语言SQL

3.3 数据定义

- SQL的数据定义功能:
  - 模式定义（数据库）
  - 表定义　（基本表）
  - 视图和索引的定义

表 3.3　SQL 的数据定义语句

操作对象	操作方式		
	创　建	删　除	修　改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	ALTER INDEX

3.3 数据定义

3.3.1 模式的定义与删除

3.3.2 基本表的定义、删除与修改

3.3.1模式的定义与删除

```
create database [sc]
on primary
(
  name = N'sc' ,
  filename =N'd:\sc.mdf' ,
  size  = 8MB ,
  maxsize = unlimited,
  filegrowth =64MB )
log on
( name = N'sc_log',
  filename = N'd:\sc_log.ldf' ,
  size = 8MB ,
  maxsize = 2048GB ,
  filegrowth = 64MB )
```

3.3 数据定义

3.3.1 模式的定义与删除

3.3.2 基本表的定义、删除与修改

3.3.2 基本表的定义、删除与修改

❖定义基本表

CREATE TABLE <表名>

(<列名> <数据类型>[<列级完整性约束条件>]

[,<列名> <数据类型>[<列级完整性约束条件>]]

...

[,<表级完整性约束条件>]);

- **<表名>**：所要定义的基本表的名字
- **<列名>**：组成该表的各个属性（列）
- **<列级完整性约束条件>**：涉及相应属性列的完整性约束条件
- **<表级完整性约束条件>**：涉及一个或多个属性列的完整性约束条件
- 如果完整性约束条件涉及到该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级。

学生表Student

[例3.5] 建立“学生”表Student。学号是主码，姓名取值唯一。

```
Create table Student
(Sno char(9) primary key,
Sname char(10) unique,
Ssex char(3),
Sage smallint,
Sdept char(3) );
```

主码

UNIQUE
约束

学生表Student

[例3.5] 建立“学生”表Student。学号是主码，姓名取值唯一。

```
insert into student(sno,sname,ssex,sage,sdept)
values('201215121','李勇','男',20,'CS');
insert into student(sno,sname,ssex,sage,sdept)
values('201215122','刘晨','女',19,'CS');
insert into student(sno,sname,ssex,sage,sdept)
values('201215123','王敏','女',18,'MA');
insert into student(sno,sname,ssex,sage,sdept)
values('201215125','张立','男',19,'IS');
```

课程表Course

[例3.6] 建立一个“课程”表Course

```
Create table Course
(Cno char(4) primary key,
Cname char(12),
Cpno char(4) ,
Ccredit smallint,
Foreign key(Cpno) references Course (Cno)
);
```

先修课

Cpno是外码
被参照表是Course
被参照列是Cno

课程表Course

[例3.6] 建立一个“课程”表Course

```
insert into course(cno,cname,cpno,ccredit) values('2','数学',NULL,2);
insert into course(cno,cname,cpno,ccredit) values('6','数据处理',NULL,2);
insert into course(cno,cname,cpno,ccredit) values('4','操作系统',6,3);
insert into course(cno,cname,cpno,ccredit) values('7','PASCAL',6,4);
insert into course(cno,cname,cpno,ccredit) values('5','数据结构',7,4);
insert into course(cno,cname,cpno,ccredit) values('1','数据库',5,4);
insert into course(cno,cname,cpno,ccredit) values('3','信息系统',1,4);
```

学生选课表SC

[例3.7] 建立一个学生选课表SC

```
Create table SC
(Sno char(9),
Cno char(4),
Grade smallint,
primary key (Sno,Cno),
foreign key (Sno) references Student(Sno),
foreign key (Cno) references Course(Cno));
```

学生选课表SC

[例3.7] 建立一个学生选课表SC

```
insert into sc (sno,cno, grade)  values('201215121',1,92);
insert into sc (sno,cno, grade)  values('201215121',2,85);
insert into sc (sno,cno, grade)  values('201215121',3,88);
insert into sc (sno,cno, grade)  values('201215122',2,90);
insert into sc (sno,cno, grade)  values('201215122',3,80);
```

2. 数据类型

- SQL中域的概念用数据类型来实现
- 定义表的属性时需要指明其数据类型及长度
- 选用哪种数据类型
  - 取值范围
  - 要做哪些运算

数据类型（续）

数据类型	含义
CHAR(*n*),CHARACTER(*n*)	长度为n的定长字符串
VARCHAR(*n*), CHARACTERVARYING(*n*)	最大长度为n的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT，INTEGER	长整数（4字节）
SMALLINT	短整数（2字节）
BIGINT	大整数（8字节）
NUMERIC(*p*，*d*)	定点数，由*p*位数字（不包括符号、小数点）组成，小数后面有*d*位数字
DECIMAL(*p*, *d*), DEC(*p*, *d*)	同NUMERIC
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(*n*)	可选精度的浮点数，精度至少为*n*位数字
BOOLEAN	逻辑布尔量
DATE	日期，包含年、月、日，格式为YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为HH:MM:SS
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

第三章 关系数据库标准语言SQL

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>]... | (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]];

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基于派生表的查询

3.4.6 Select语句的一般形式

3.4.1 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

1.选择表中的若干列

- 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、所在系。

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

选择表中的若干列（续）

- 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,Sage,Sdept
FROM Student;
```

或

```
SELECT  *
FROM Student;
```

查询经过计算的值（续）

- 查询经过计算的值
  - SELECT子句的<目标列表达式>不仅可以为表中的属性列，也可以是表达式

[例3.19] 查全体学生的姓名及其出生年份。

```
SELECT Sname,2020-Sage
FROM Student;
```

查询经过计算的值（续）

[例3.20] 查询全体学生的姓名、出生年份和所在的院系，要求用小写字母表示系名。

```
Select sname,     'Year of Birth:',
                  (2020-sage) as birthyear,    lower(sdept)
from student
```

可以使用列别名改变查询结果的列标题:

```
SELECT Sname NAME,     'Year of Birth:'  BIRTH,
       2020-Sage  BIRTHYEAR,
       LOWER(Sdept)  DEPARTMENT
FROM Student;
```

3.4.1 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

2. 选择表中的若干元组

❖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;

❖指定**DISTINCT**关键词，去掉表中重复的行

SELECT DISTINCT Sno FROM SC;

（2）查询满足条件的元组

表3.6 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] 查询计算机科学系全体学生的名单。

```
select sname from student where sdept='CS'
```

[例3.23]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3.24]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3.25] 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3.26] 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>

[例3.27]查询CS系和IS系学生的姓名和性别

```
select sname,ssex from student
where sdept in ('CS','IS')
```

[例3.28]查询既不是CS、也不是IS的学生的姓名和性别。

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

④ 字符匹配

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

<匹配串>可以是一个完整的字符串，也可以含有通配符%和 _

- **%**（百分号） 代表任意长度（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） 代表任意单个字符。
  - 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201215121的学生的详细情况。

```
SELECT *
FROM  Student
WHERE  Sno LIKE '201215121';
```

等价于：

```
SELECT  *
FROM  Student
WHERE Sno = ' 201215121 ';
```

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3.32] 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

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，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;
```

3.4.1 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

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**;缺省值为升序
- 对于空值，排序时显示的次序由具体系统实现来决定

ORDER BY子句 （续）

[例3.39]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3.40]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：
  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

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1号课程的学生平均成绩。

```
select avg(grade) from sc where cno='1'
```

[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='1'
```

[例3.45] 查询学生201215121选修课程的总学分数。

```
select sum(ccredit) from sc,course
where sno='200215121' and sc.cno=course.cno
```

3.4.1 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

5. GROUP BY子句

❖ **GROUP BY子句分组：**

细化聚集函数的作用对象

- 如果未对查询结果分组，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
- 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3.46] 求各个课程号及相应的选课人数。

```
select cno,count(sno) from sc
group by cno
```

[例3.47] 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）

[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

下面的语句是不对的：

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

因为WHERE子句中是不能用聚集函数作为条件表达式

正确的查询语句应该是：

```
select sno,avg(grade) from sc
group by sno
having avg(grade)>=90
```

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。

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基于派生表的查询

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.自身连接

3.外连接

4.多表连接

1. 等值与非等值连接查询

❖等值连接：连接运算符为=

[例 3.49] 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
```

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

等值与非等值连接查询（续）

❖自然连接

[例 3.50] 对[例 3.49]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）

❖一条SQL语句可以同时完成选择和连接查询，这时WHERE子句是由连接谓词和选择谓词组成的复合条件。

[例 3.51]查询选修2号课程且成绩在90分以上的所有学生的学号和姓名。

```
select sno, sname from student,sc
where student.sno=sc.sno and sc.cno='2'
      and grade>90
```

执行过程:

- 先从SC中挑选出Cno='2'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.自身连接

3.外连接

4.多表连接

2. 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 3.52]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）

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

SECOND表（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

自身连接（续）

查询结果：

Cno	Pcno
1	7
3	5
5	6

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

3. 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
  - 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（续）

[例 3. 53] 改写**[例 3.49]**

查询每个学生及其选修课程的情况，没有选修课程的同学也显示出来

```
select student.sno,sname,ssex,sage,sdept,
       cno,grade
from student left outer join sc on
   (student.sno=sc.sno)
```

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

4. 多表连接

❖ 多表连接：两个以上的表进行连接

[例3.54]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
```

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基于派生表的查询

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

```
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）

- 上层的查询块称为外层查询或父查询
- 下层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
  - 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

3.4.3 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 3.55] 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
201215121	李勇	CS
201215122	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 from student
                where sname='刘晨')
```

此查询为不相关子查询。

带有IN谓词的子查询（续）

[例3.56]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'))
```

带有IN谓词的子查询（续）

用连接查询实现[例 3.56]：

[例3.56]查询选修了课程名为“信息系统”的学生学号和姓名

```
select student.sno,sname
from student,sc,course
where student.sno=sc.sno and
   sc.cno=course.cno
     and cname='信息系统'
```

3.4.3 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< >）。

在[例 3.55]中，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,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）

[例 3.57]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

❖可能的执行过程

■从外层查询中取出SC的一个元组x，将元组x的Sno值（201215121）传送给内层查询。

```
SELECT AVG(Grade)
FROM SC y
WHERE y.Sno='201215121';
```

带有比较运算符的子查询（续）

- 可能的执行过程（续）
  - 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）

- 可能的执行过程（续）
  - 执行这个查询，得到

    （**201215121,1**）

    （**201215121,3**）

  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:

    （**201215121,1**）

    （**201215121,3**）

    （**201215122,2**）

3.4.3 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（续）

使用ANY或ALL谓词时必须同时使用比较运算

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

语义为（续）

<= 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 3.58] 查询非计算机科学系中比计算机科学系任意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='CS') and sdept <>'CS'
```

执行过程：

（1）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 3.59] 查询非计算机科学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'
```

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3.7 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

3.4.3 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 3.60]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表
- 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送入结果表

```
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 3.61] 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：

    （∀**x**）**P** ≡ ¬ （∃ **x**（¬ **P**））

带有EXISTS谓词的子查询（续）

[例 3.62] 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 3.63]查询至少选修了学生201215122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201215122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201215122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：

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

  $\equiv \neg\ (\exists y\ (\neg\ (\neg\ p \vee\ q)\ )\ )$

  $\equiv \neg\ \exists y\ (p \wedge \neg q)$

- 变换后语义：不存在这样的课程y，学生201215122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='201215122' 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基于派生表的查询

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

集合查询（续）

[例 3.64] 查询计算机科学系的学生及年龄不大于19岁的学生。

```
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

[例 3.65] 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'
```

集合查询（续）

[例3.66] 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
```

[例 3.67]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'
intersect
select sno from sc where cno='2'
```

[例 3.68] 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
```

3.4 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基于派生表的查询

3.4.6 Select语句的一般形式

3.4.5 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，这时子查询生成的临时派生表（**Derived Table**）成为主查询的查询对象

[例3.57]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, grade)
where sc.Sno=avg_sc.avg_sno and
   sc.grade>=avg_sc.grade
```