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/DB2021.html

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>] ...

[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

- 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]];

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'\'后面接的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 ';
```

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 ASC，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;
```

查询格式

- 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]];

3.4.1 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。

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

查询格式

- 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]];

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

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

```
SELECT R.*,S.DWM FROM R, S WHERE R.DWH=S.DWH;
SELECT XM,XB,DWH FROM R,S WHERE R.DWH=S.DWH AND XB='男';
```

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(例如，=ANY等价于IN)

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

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.课程a没有被学生A选修（即，在选课记录表里不存在A对a的选课记录）

不存在一个课程，这个课程没有被选修（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?

命题逻辑

- 逻辑(Logic):表示知识并进行推理
  - 命题逻辑、谓词逻辑
- 命题：能判断真假的陈述句。
- 真值：一个命题表达的判断结果称为命题的真值。命题的真值有“真”和“假”两种，分别用True、T、1(真)和False、F、0(假)来表示。真值为真的命题称为真命题，真值为假的命题称为假命题。任何命题的真值是惟一的。
- 注: 一切没有判断内容的句子，无所谓是非的句子，如感叹句、疑问句、祈使句等都不是命题。

命题逻辑

1. 2是素数。
2. 雪是黑色的。
3. 2+3=5 。
4. 明年十月一日是晴天。
5. 这朵花多好看呀！
6. 3能被2整除.
7. 明天下午有会吗？
8. 请关上门！
9. x+y>5 。

命题判断的关键：
1.是否是陈述句；
2.真值是否是唯一的。

命题逻辑

表示法：

	表示法	例子	有关概念
简单命题	$p,q,r,\ldots,$ $p_i,q_i,r_i,\ldots$	p: 2是素数 q:雪是黑色的	命题符号化：将命题的符号放在该命题的前面
命题常项（常元）	同上	同上	真值确定的简单命题
命题变项（变元）	同上	p: x+y>5	真值可以变化的简单陈述句
复合命题	$p \wedge q$	2是素数和偶数	

注：一个符号表示的是命题常项还是命题变项由上下文决定。

•命题联结词

常用的逻辑联结词有五种：否定联结词、合取联结词、析取联结词、蕴涵联结词和等价联结词。

1. 否定联结词

定义1.1 设p为命题，则p的否定是一个复合命题，记作：$\neg p$，读作“非p”或“p的否定”。 ¬为否定联结词。$\neg p$为真当且仅当p为假。

表1.1

p	$\neg p$
0	1
1	0

【例】否定下列命题。

p：王强是一名大学生。

$\neg p$：王强不是一名大学生。

2. 合取联结词

定义1.2 设p和q均为命题，则p和q的合取是一个复合命题，记作$p \wedge q$，读作“p与q”或“p合取q”。 $\wedge$为**合取联结词**。$p \wedge q$为真当且仅当p和q同时为真。

表1.2

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

【例】设 p：北京成功举办了第29届夏季奥运会。

q：今年10月1日是我国国庆60周年。

则p∧q：北京成功举办了第29届夏季奥运会并且今年10月1日是我国国庆60周年。

3. 析取联结词

定义1.3 设p和q均为命题，则p和q的析取是一个复合命题，记作$p \vee q$，读作“p或q”或者“p析取q”。 $\vee$为**析取联结词**。$p \vee q$为真当且仅当p与q中至少一个为真。

表1.3

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

“$\vee$”与汉语中的“或”相似，但又不相同。汉语中的或有可兼或与不可兼或(排斥或)的区分。

【例】下列两个命题中的“或”，哪个是可兼或？哪个是不可兼或？

(1)在家里看奥运会或在现场看奥运会。(不可兼或)

(2)灯泡有故障或开关有故障。(可兼或)

4. 蕴涵联结词

定义1.4 设p和q均为命题，p与q的蕴涵式是个复合命题，记为：$p\rightarrow q$。读作“如果p，那么q”或“若p，则q”。$\rightarrow$为蕴涵联结词。$p\rightarrow q$为假当且仅当p为真且q为假。p称为条件命题$p\rightarrow q$的前件，q称为条件命题$p\rightarrow q$的后件。

表1.4

p	q	$p\rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

【例】p：小王努力学习。q：小王学习成绩优秀。

$p\rightarrow q$：如果小王努力学习，那么他的学习成绩就优秀。

联结词“$\rightarrow$”与汉语中的“如果…，那么…”或“若…，则…”相似，但又是不相同的。

将下列各命题符号化（并判断3-6的真值情况）

1. 只要不下雨，我就骑自行车上班.
2. 只有不下雨，我才骑自行车上班.
3. 若2+2=4，则太阳从东方升起.
4. 若2+2≠4，则太阳从东方升起.
5. 若2+2=4，则太阳从西方升起.
6. 若2+2≠4，则太阳从西方升起.

5. 等价联结词

定义1.5 设p和q均为命题，其复合命题$p\leftrightarrow q$称为等价式，$p\leftrightarrow q$读作：“p当且仅当q”。$\leftrightarrow$为**等价联结词**。$p\leftrightarrow q$为真当且仅当p和q的真值相同。

表1.5

p	q	$p\leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

【例】 p：张华是三好学生。
q：张华德、智、体全优秀。
$p\leftrightarrow q$：张华是三好学生当且仅当德、智、体全优秀。

命题逻辑

- 联结词的比较表

*p,q*为两个命题

复合命题	…式	记作	联结词	为真的条件 iff	优先级	其他
否定	非 p(p 的否定)	$\neg p$	$\neg$	p 为假	1	
合取	p 并且 q(p 和 q)	$p \wedge q$	$\wedge$	p 与 q 同时为真	2	既…又…,不仅…而且
析取	P 或 q	$p \vee q$	$\vee$	p 与 q 至少一个为真	3	相容或
蕴涵	如果 p 则 q	$p \rightarrow q$	$\rightarrow$	$\neg$(P 为真且 q 为假)	4	只要 p 就 q, p 仅当 q
等价	P 当且仅当 q	$p \leftrightarrow q$	$\leftrightarrow$	p, q 真值相同	5	

注：以上5种联结词也称真值联结词或逻辑联结词或逻辑运算符。

等值演算—逻辑等价

- 设A,B为两个命题公式，若等价式$A \leftrightarrow B$是重言式，则称A与B是等值的，记作A⇔B.
- A⇔B不是命题公式
- 可通过判断A与B的真值表是否相同，来判断A与B是否等值。

等值演算—逻辑等价

- 判断下列命题公式是否等值
  （1）¬(p∨q)与¬p∨¬q；
  （2）¬(p∨q)与¬p∧¬q；

p	q	¬p	¬q	p∨q	¬(p∨q)	¬p∨¬q	¬p∧¬q
0	0	1	1	0	1	1	1
0	1	1	0	1	0	1	0
1	0	0	1	1	0	1	0
1	1	0	0	1	0	0	0

- 24个重要的等值式：$\neg \wedge \vee \rightarrow \leftrightarrow \Leftrightarrow$

编号	表达式	中文名称
1	$A \Leftrightarrow \neg \neg A$	双重否定律
2	$A \Leftrightarrow A \vee A$	等幂律（或幂等律）
3	$A \Leftrightarrow A \wedge A$	
4	$A \vee B \Leftrightarrow B \vee A$	交换律
5	$A \wedge B \Leftrightarrow B \wedge A$	
6	$(A \vee B) \vee C \Leftrightarrow A \vee (B \vee C)$	结合律
7	$(A \wedge B) \wedge C \Leftrightarrow A \wedge (B \wedge C)$	
8	$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$	分配律
9	$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$	
10	$\neg (A \vee B) \Leftrightarrow \neg A \wedge \neg B$	德●摩根律
11	$\neg (A \wedge B) \Leftrightarrow \neg A \vee \neg B$	
12	$A \vee (A \wedge B) \Leftrightarrow A$	吸收律
13	$A \wedge (A \vee B) \Leftrightarrow A$	

14	$A \vee 1 \Leftrightarrow 1$	零律
15	$A \wedge 0 \Leftrightarrow 0$	
16	$A \vee 0 \Leftrightarrow A$	同一律
17	$A \wedge 1 \Leftrightarrow A$	
18	$A \vee \neg A \Leftrightarrow 1$	排中律
19	$A \wedge \neg A \Leftrightarrow 0$	矛盾律
20	$A \rightarrow B \Leftrightarrow \neg A \vee B$	蕴涵等值式
21	$A \leftrightarrow B \Leftrightarrow (A \rightarrow B) \wedge (B \rightarrow A)$	等价等值式
22	$A \rightarrow B \Leftrightarrow \neg B \rightarrow \neg A$	假言易位
23	$A \leftrightarrow B \Leftrightarrow \neg A \leftrightarrow \neg B$	等价否定等值律
24	$(A \rightarrow B) \wedge (A \rightarrow \neg B) \Leftrightarrow \neg A$	归谬论

- 根据已知的等值式推演出另外一些等值式的过程称为等值演算。

谓词逻辑

- 命题逻辑的问题

- 例2： 苏格拉底三段论
  - P：所有人必死 Q：苏格拉底是人 R：苏格拉底必死
  - 应该有 $(P \wedge Q) \Rightarrow R$，也就是公式$(P \wedge Q) \rightarrow R$应该是恒真的。
  - 显然该公式不是恒真的，解释{P，Q，$\neg$R}就能弄假该公式。

因为苏格拉底不等于所有人

- 原因：命题R和命题P, Q是有内在关系的，只是这种关系在命题逻辑中无法表示。

- 因此，需要对命题的成分、结构和命题间的共同特性等作进一步的分析，分析出对象(苏格拉底)、谓词（是人）和量词（所有），以期达到表达出个体与总体的内在联系和数量关系，这正是谓词逻辑所要研究的问题。

谓词逻辑

- 引入谓词后，命题P就可确切地符号化如下：

$$\forall x(H(x)\rightarrow M(x))$$

命题P的否定命题为：

$$\neg P = \neg(\forall x(H(x)\rightarrow M(x)))$$
$$= \exists x(H(x)\wedge\neg M(x))$$

亦即“至少有一个人是不死的”。这个命题才是“所有人都要死”的否定。

- 三段论的三个命题，在谓词逻辑中可以如下表示：

P：$\forall x(H(x)\rightarrow M(x))$

Q：H(苏格拉底)

R：M(苏格拉底)

谓词逻辑

- 我们只关注于量词

(1) 所有的老虎都会吃人。

（1）令 P(x)：x会吃人 U(x)：x是老虎

应该使用全称量词规则，将特性谓词作为前件加入

则符号化的正确形式应该是

$$(\forall x)(U(x)\rightarrow P(x))$$

“存在x, 若x是老虎，则x会吃人”，符合原命题的逻辑含义

谓词逻辑

- 我们只关注于量词

(2) 有些人登上过月球。

(2) 令 S(x)：x登上过月球　　U(x)：x是人

应该使用存在量词规则，将特性谓词作为合取式前项加入

则符号化正确形式为

$$(\exists x)(U(x) \wedge S(x))$$

- “存在x，x是人并且x登上过月球”，符合原命题的逻辑含义

谓词逻辑

（2）天下乌鸦一般黑

设 F(x)：x是乌鸦；G(x, y)：x与y一般黑，则：

$(\forall x)(\forall y)(F(x) \wedge F(y) \rightarrow G(x, y))$

或者$\neg (\exists x)(\exists y)(F(x) \wedge F(y) \wedge \neg G(x, y))$；

（3）没有人登上过木星

设H(x)：x是人；M(x)：x登上过木星，则：

$\neg (\exists x)(H(x) \wedge M(x)) \Leftrightarrow (\forall x)(\neg H(x) \vee (\neg M(x)))$

或者 $(\forall x)(H(x) \rightarrow \neg M(x)) \Leftrightarrow (\forall x)(\neg H(x) \vee (\neg M(x)))$；

带有EXISTS谓词的子查询(续）

[例47]查询至少选修了学生200215122选修的全部课程的学生号码。

解题思路：

- 用逻辑蕴函表达：查询学号为x的学生，对所有的课程y，只要200215122学生选修了课程y，则x也选修了y。
- 形式化表示：

  用P(y)表示谓词 “学生200215122选修了课程y”

  用q(y)表示谓词 “学生x选修了课程y”

  则上述查询为: $(\forall y)\ p \rightarrow q$

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200215122选修了y，而学生x没有选。

带有EXISTS谓词的子查询(续）

- 不存在这样的课程y，学生200215122选修了y，而学生x没选。
  - 用NOT EXISTS谓词表示：

```
SELECT DISTINCT Sno
FROM SC SCX
WHERE NOT EXISTS
        (SELECT *
         FROM SC SCY
         WHERE SCY.Sno = ' 200215122 '  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 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48] 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;
```

- 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）

方法二：

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

集合查询（续）

[例49] 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50] 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
```

集合查询（续）

- [例50] 实际上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 Sdept= 'CS' AND  Sage<=19;
```

集合查询（续）

[例51] 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

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='2 ';
```

集合查询（续）

[例51]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
                        (SELECT Sno
                         FROM SC
                         WHERE Cno=' 2 ');
```

集合查询（续）

[例52] 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;
```

集合查询（续）

[例52]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;
```

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

3.4.5 SELECT语句的一般格式

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]] …
[WHERE <条件表达式>]
[GROUP BY <列名1>
[HAVING <条件表达式>]]
[ORDER BY <列名2> [ASC | DESC]

第三章 关系数据库标准语言SQL

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.5.1 插入数据

- 两种插入数据方式

  1. 插入元组

  2. 插入子查询结果

  - 可以一次插入多个元组

一、插入元组

- 语句格式

  INSERT

  INTO <表名> [(<属性列1>[，<属性列2 >…)]

  VALUES (<常量1> [，<常量2>]　…　　)

- 功能
  - 将新元组插入指定表中

插入元组（续）

- **INTO**子句
  - 属性列的顺序可与表定义中的顺序不一致
  - 没有指定属性列
  - 指定部分属性列
- **VALUES**子句
  - 提供的值必须与**INTO**子句匹配
    - 值的个数
    - 值的类型

插入元组（续）

［例1］ 将一个新学生元组（学号：200215128；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到Student表中。

```
INSERT
INTO Student (Sno，Sname，Ssex，Sdept，Sage)
VALUES ('200215128'，'陈冬'，'男'，'IS'，18);
```

插入元组（续）

［例2］ 将学生张成民的信息插入到Student表中。

```
INSERT
INTO  Student
VALUES (‘200215126’,  ‘张成民’,  ‘男’, 18,
'CS');
```

插入元组（续）

［例3］ 插入一条选课记录('200215128'，'1 ')。

```
INSERT
INTO SC(Sno，Cno)
VALUES (‘ 200215128 ′，  ‘ 1 ′);
```

RDBMS将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT
INTO SC
VALUES (' 200215128 '，' 1 '，NULL);
```

二、插入子查询结果

- 语句格式

  INSERT

  INTO <表名> [(<属性列1> [，<属性列2>…)]

  子查询；

- 功能

  将子查询结果插入指定表中

插入子查询结果（续）

- **INTO子句(与插入元组类似)**

- 子查询
  - **SELECT子句目标列必须与INTO子句匹配**
    - 值的个数
    - 值的类型

插入子查询结果（续）

[例4] 对每一个系，求学生的平均年龄，并把结果存入数据库。

第一步：建表

```
CREATE  TABLE  Dept_age
   (Sdept  CHAR(15)              /* 系名*/
    Avg_age SMALLINT);           /*学生平均年龄*/
```

插入子查询结果（续）

第二步：插入数据

```
INSERT
INTO  Dept_age(Sdept，Avg_age)
    SELECT  Sdept，AVG(Sage)
    FROM  Student
    GROUP BY Sdept;
```

插入子查询结果（续）

RDBMS在执行插入语句时会检查所插元组是否破坏表上已定义的完整性规则

- 实体完整性
- 参照完整性
- 用户定义的完整性
  - **NOT NULL**约束
  - **UNIQUE**约束
  - 值域约束

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.4.2 修改数据

- 语句格式

UPDATE <表名>
SET <列名>=<表达式>[，<列名>=<表达式>]…
[WHERE <条件>];

- 功能
  - 修改指定表中满足**WHERE**子句条件的元组

修改数据（续）

- **SET子句**
  - 指定修改方式
  - 要修改的列
  - 修改后取值
- **WHERE子句**
  - 指定要修改的元组
  - 缺省表示要修改表中的所有元组

修改数据（续）

- 三种修改方式
  1. 修改某一个元组的值
  2. 修改多个元组的值
  3. 带子查询的修改语句

1. 修改某一个元组的值

[例5] 将学生200215121的年龄改为22岁

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 200215121 ';
```

2. 修改多个元组的值

[例6] 将所有学生的年龄增加1岁

```
UPDATE Student
SET Sage= Sage+1;
```

3. 带子查询的修改语句

[例7] 将计算机科学系全体学生的成绩置零。

```
UPDATE SC
SET  Grade=0
WHERE  'CS'=
        (SELETE Sdept
         FROM  Student
         WHERE  Student.Sno = SC.Sno);
```

修改数据（续）

RDBMS在执行修改语句时会检查修改操作是否破坏表上已定义的完整性规则

- 实体完整性
- 主码不允许修改
- 用户定义的完整性
  - **NOT NULL**约束
  - **UNIQUE**约束
  - 值域约束

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.5.3 删除数据

- 语句格式

  DELETE

  FROM <表名>

  [WHERE <条件>**];**

- 功能
  - 删除指定表中满足**WHERE**子句条件的元组
- **WHERE**子句
  - 指定要删除的元组
  - 缺省表示要删除表中的全部元组，表的定义仍在字典中

删除数据（续）

- 三种删除方式

1. 删除某一个元组的值

2. 删除多个元组的值

3. 带子查询的删除语句

1. 删除某一个元组的值

[例8] 删除学号为200215128的学生记录。

```
DELETE
FROM Student
WHERE Sno= 200215128 ';
```

2. 删除多个元组的值

[例9] 删除所有的学生选课记录。

```
DELETE
FROM SC;
```

3. 带子查询的删除语句

[例10] 删除计算机科学系所有学生的选课记录。

```
DELETE
FROM SC
WHERE  'CS'=
            (SELETE Sdept
             FROM Student
             WHERE Student.Sno=SC.Sno);
```

第三章 关系数据库标准语言SQL

3.6 视 图

视图的特点

- 虚表，是从一个或几个基本表（或视图）导出的表
- 只存放视图的定义，不存放视图对应的数据
- 基表中的数据发生变化，从视图中查询出的数据也随之改变

3.6 视　图

基于视图的操作

- 查询
- 删除
- 受限更新
- 定义基于该视图的新视图

3.6 视　图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

3.6.1 定义视图

- 建立视图
- 删除视图

一、建立视图

- 语句格式

  CREATE VIEW

  <视图名> [(<列名> [，<列名>]...)]

  AS <子查询>

  [WITH CHECK OPTION];

- 组成视图的属性列名：全部省略或全部指定
- 子查询不允许含有**ORDER BY**子句和**DISTINCT**短语

建立视图（续）

- **RDBMS执行CREATE VIEW语句时只是把视图定义存入数据字典，并不执行其中的SELECT语句。**
- 在对视图查询时，按视图的定义从基本表中将数据查出。

建立视图（续）

[例1] 建立信息系学生的视图。

```
CREATE VIEW IS_Student
AS
SELECT Sno，Sname，Sage
FROM   Student
WHERE  Sdept= 'IS';
```

建立视图（续）

[例2]建立信息系学生的视图，并要求进行修改和插入操作时仍需保证该视图只有信息系的学生 。

```
CREATE VIEW IS_Student
AS
SELECT Sno，Sname，Sage
FROM  Student
WHERE  Sdept= 'IS'
WITH CHECK OPTION;
```

建立视图（续）

对IS_Student视图的更新操作：

- 修改操作：自动加上Sdept= 'IS'的条件
- 删除操作：自动加上Sdept= 'IS'的条件
- 插入操作：自动检查Sdept属性值是否为'IS'
  - 如果不是，则拒绝该插入操作
  - 如果没有提供Sdept属性值，则自动定义Sdept为'IS'

建立视图（续）

- 基于多个基表的视图

[例3] 建立信息系统选修了1号课程的学生视图。（成绩单）

```
CREATE VIEW IS_S1(Sno，Sname，Grade)
AS
SELECT Student.Sno，Sname，Grade
FROM  Student，SC
WHERE  Sdept= 'IS' AND
          Student.Sno=SC.Sno AND
          SC.Cno= '1';
```

建立视图（续）

- 基于视图的视图

[例4] 建立信息系统选修了1号课程且成绩在90分以上的学生的视图。

```
CREATE VIEW IS_S2
AS
SELECT Sno，Sname，Grade
FROM  IS_S1
WHERE  Grade>=90;
```

建立视图（续）

- 带表达式的视图

[例5] 定义一个反映学生出生年份的视图。

```
CREATE  VIEW BT_S(Sno，Sname，Sbirth)
AS
SELECT Sno，Sname，2000-Sage
FROM  Student;
```

建立视图（续）

- 分组视图

[例6] 将学生的学号及他的平均成绩定义为一个视图

假设SC表中“成绩”列Grade为数字型

```
CREAT  VIEW S_G(Sno，Gavg)
AS
SELECT Sno，AVG(Grade)
FROM  SC
GROUP BY Sno;
```

建立视图（续）

- 不指定属性列

[例7]将Student表中所有女生记录定义为一个视图

```
CREATE VIEW F_Student(F_Sno，name，sex，age，dept)
AS
SELECT *
FROM  Student
WHERE Ssex='女'；
```

缺点：

修改基表Student的结构后，Student表与F_Student视图的映象关系被破坏，导致该视图不能正确工作。

二、删除视图

- 语句的格式：

  DROP VIEW <视图名>;

  - 该语句从数据字典中删除指定的视图定义
  - 如果该视图上还导出了其他视图，使用**CASCADE**级联删除语句，把该视图和由它导出的所有视图一起删除
  - 删除基表时，由该基表导出的所有视图定义都必须显式地使用**DROP VIEW**语句删除

删除视图(续）

［例8］ 删除视图**BT_S**： **DROP VIEW BT_S;**

删除视图**IS_S1**：**DROP VIEW IS_S1;**

- 拒绝执行
- 级联删除：

DROP VIEW IS_S1 CASCADE;

3.6 视 图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

3.6.2 查询视图

- 用户角度：查询视图与查询基本表相同
- **RDBMS**实现视图查询的方法
  - 视图消解法（**View Resolution**）
    - 进行有效性检查
    - 转换成等价的对基本表的查询
    - 执行修正后的查询

查询视图（续）

[例9] 在信息系学生的视图中找出年龄小于**20**岁的学生。

```
SELECT   Sno，Sage
 FROM     IS_Student
 WHERE   Sage<20;
```

IS_Student视图的定义 **(**参见视图定义例**1)**

查询视图（续）

视图消解转换后的查询语句为：

```
SELECT  Sno，Sage
FROM  Student
WHERE  Sdept= 'IS'  AND  Sage<20；
```

查询视图（续）

[例10] 查询选修了1号课程的信息系学生

```
SELECT  IS_Student.Sno，Sname
FROM    IS_Student，SC
WHERE  IS_Student.Sno =SC.Sno AND SC.Cno= '1';
```

查询视图（续）

- 视图消解法的局限
  - 有些情况下，视图消解法不能生成正确查询。

查询视图（续）

[例11]在S_G视图中查询平均成绩在90分以上的学生学号和平均成绩

```
SELECT *
FROM   S_G
WHERE  Gavg>=90;
```

S_G视图的子查询定义：

```
CREATE VIEW S_G (Sno，Gavg)
AS
SELECT  Sno，AVG(Grade)
FROM  SC
GROUP BY Sno;
```

查询转换

错误：

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

正确：

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

3.6 视　图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

更新视图（续）

[例12] 将信息系学生视图IS_Student中学号200215122的学生姓名改为“刘辰”。

```
UPDATE  IS_Student
SET  Sname= '刘辰'
WHERE  Sno= ' 200215122 ';
```

转换后的语句：

```
UPDATE  Student
SET Sname= '刘辰'
WHERE Sno= ' 200215122 ' AND Sdept= 'IS';
```

更新视图（续）

[例13] 向信息系学生视图IS_S中插入一个新的学生记录：
200215129，赵新，20岁

```
INSERT
INTO IS_Student
VALUES(‘95029’，‘赵新’，20)；
```

转换为对基本表的更新：

```
INSERT
INTO  Student(Sno，Sname，Sage，Sdept)
VALUES(‘200215129 '，'赵新'，20，'IS' )；
```

更新视图（续）

[例14]删除信息系学生视图IS_Student中学号为200215129的记录 。

```
DELETE
FROM IS_Student
WHERE Sno= ' 200215129 ';
```

转换为对基本表的更新：

```
DELETE
FROM Student
WHERE Sno= ' 200215129 ' AND Sdept= 'IS';
```

更新视图（续）

- 更新视图的限制：一些视图是不可更新的，因为对这些视图的更新不能唯一地有意义地转换成对相应基本表的更新

例：视图S_G为不可更新视图。

```
UPDATE  S_G
SET     Gavg=90
WHERE  Sno= '200215121';
```

这个对视图的更新无法转换成对基本表SC的更新

更新视图（续）

- 允许对行列子集视图进行更新
- 对其他类型视图的更新不同系统有不同限制

3.6 视　图

3.6.1 定义视图

3.6.2 查询视图

3.6.3 更新视图

3.6.4 视图的作用

3.6.4 视图的作用

- 1. 视图能够简化用户的操作
- 2. 视图使用户能以多种角度看待同一数据
- 3. 视图对重构数据库提供了一定程度的逻辑独立性
- 4. 视图能够对机密数据提供安全保护
- 5. 适当的利用视图可以更清晰的表达查询

什么时候使用视图

- 经常用到的查询，或较复杂的联合查询应当创立视图，这是会优化性能的

  还有就是涉及到权限管理方面，比如某表中的部分字段含有机密信息，不应当让低权限的用户访问到的情况，这时候给这些用户提供一个适合他们权限的视图，供他们阅读自己的数据就行了。

- 当一个查询逻辑复杂，且别的查询需要查询这个查询的结果时用视图。还有就是敏感数据