南京大学 NANJING UNIVERSITY

奋进行动 团结奋斗 争先进位 坐言起行 应势而动

数 据 管 理 基 础

第3章 关系数据库标准语言SQL

（3.4 SQL数据查询）

智能软件与工程学院

3.4 SQL数据查询

SQL数据查询语句概述 1

❑映像语句

目 标 子 句	SELECT * | colname { , colname ... }
范 围 子 句	FROM tablename { , tablename ... }
条 件 子 句	[WHERE search_condition]
分 组 子 句	[GROUP BY colname { , colname ... }
分组查询子句	[HAVING group_condition]]
排序输出子句	[ORDER BY colname [ASC | DESC]
	{ , colname [ASC | DESC] ... }];

SQL数据查询语句概述 2

❑映像语句各子句的执行顺序

```
SELECT  * | colname { , colname ... }
FROM  tablename { , tablename ... }
[ WHERE  search_condition ]
[ GROUP BY  colname { , colname ... } ]
    [ HAVING  group_condition ] ]
[ ORDER BY  colname [ ASC | DESC ]
        { , colname [ ASC | DESC ] ... } ];
```

动画执行过程反映了这些子句被处理的顺序！

基本映像语句的组成 1

1）目标子句：SELECT [distinct] column-name-list | expressions | *

- 定义生成结果表时目标列的投影方式

➢ 目标子句的构造方式

- 简单的属性投影：可以通过‘表名.列名’的方式来指明是对哪一张表中的哪个列的投影；
- 可以将一个表达式的计算结果投影到某个目标列上；
- 可以在目标子句中对投影得到的结果列进行重命名：
  <column_expression> AS <colname>
- 可用‘*’来代替表中的所有列；
- 可添加保留字distinct来消除结果表中的重复元组。

基本映像语句的组成 2

2) 范围子句： FROM tablename { , tablename ... }

- 指定操作对象（被访问的表）

➢ 可以在FROM子句中对一个表重新命名（即定义一个别名 alias）：
<table_name> <alias_name>

➢ 在FROM子句中对表进行重命名的作用
① 便于实现表的‘自身连接’（一个表自己与自己的连接运算）
② 可以通过重命名来简化SQL命令的书写
③ 可以提高SQL命令的可读性

❑ ‘SELECT子句’和‘FROM子句’是一条映像语句中必不可少的两个组成部分

SELECT & FROM 子句总结

- **SELECT [distinct] column-name-list | expressions | ***
  - 目标子句，用于投影生成结果表；
  - 可以是单个列投影，也可以是对一个表达式的计算结果进行投影；
  - 可以用 * 表示投影出表中的所有列（按照创建表时的列定义顺序显示）；
  - 可以用distinct谓词要求系统对结果元组进行唯一性检查（对结果元组进行去重处理）。

- **FROM tablename-list**
  - 范围子句，定义本次查询可以访问的表；
  - 如果对表进行了换名，那么必须通过定义的别名来访问对应的表；
  - FROM子句中的表（别名）不能重名，SELECT子句中的结果列不能重名；
  - 如果范围子句中的表存在同名的列，可以通过‘表名.列名’的方式来明确定义需要访问哪一张表中的列；否则可以直接通过列名来访问相关的列。

基本映像语句的组成 3

3) 条件子句： WHERE search_condition

- 是映像语句中的可选成分，用于定义查询条件（即结果表中的元组必须满足的条件）。
- 包括‘单个表中的元组选择条件’以及‘表与表之间的连接条件’都需要在**WHERE**子句中通过一定的逻辑表达式显式地表示出来。
- 在**FROM**子句中给出的表只是表明此次查询需要访问这些表，它们之间是通过笛卡儿积运算进行合并的；
- 如果需要执行它们之间的‘θ-连接’或‘自然连接’运算，则需要在**WHERE**子句中显式地给出它们的连接条件。

数据管理基础

ch3.4.1 SQL数据查询（单表）

智能软件与工程学院

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 ] ];
```

- SELECT子句：指定要显示的列
- FROM子句：指定查询对象（基本表、视图 或 子查询）
- WHERE子句：指定查询条件
- GROUP BY子句：对查询结果按指定列的值分组，在该列上取值相等的元组为一个组。通常会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：对查询结果表按指定列值的升序或降序排序

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;
```

查询全部列

- [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 1

- **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, 2014-Sage        /*假设当时为2014年*/
FROM Student;
```

  - 输出结果：

Sname	2014-Sage
李勇	1994
刘晨	1995
王敏	1996
张立	1995

查询经过计算的值 2

- [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: ',2014-Sage,LOWER(Sdept)
FROM Student;
```

- 输出结果：

Sname	'Year of Birth:'	2014-Sage	LOWER(Sdept)
李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

查询经过计算的值 3

- 使用列别名改变查询结果的列标题

```
SELECT Sname NAME, 'Year of Birth:'  BIRTH,
       2014-Sage  BIRTHDAY,
       LOWER(Sdept)  DEPARTMENT
FROM Student;
```

- 输出结果：

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

选择表中的若干元组 1

- [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;
```

执行上面的SELECT语句后，结果为：

Sno
201215121
201215121
201215121
201215122
201215122

消除取值重复的行（续）

- 指定DISTINCT关键词，去掉表中重复的行
  - 如果没有指定DISTINCT关键词，则缺省为ALL

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

- 执行结果：

Sno
201215121
201215122

查询满足条件的元组

查 询 条 件	谓　　词
比　　较	=, >, <, >=, <=, !=, <>, !>, !<
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 … 或 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]查询计算机科学系（CS）、数学系（MA）和信息系（IS）学生的姓名和性别。

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

- [例3.28]查询既不是计算机科学系、数学系，也不是信息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 1

- 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%（任意长度（长度可以为0）的字符串）和 _（任意单个字符）
    - 例如：a%b表示以a开头，以b结尾的任意长度的字符串
    - 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

- 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’;
```

④ 字符匹配 2

- **匹配串为含通配符的字符串(1)**

- **[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__';
```

④ 字符匹配 3

- 匹配串为含通配符的字符串(2)

- [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%';
```

④ 字符匹配 4

- 使用换码字符将通配符转义为普通字符

  ESCAPE '\'　表示“\”为换码字符

- [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 '\' ;
```

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;
```

- [例3.27] 查询计算机科学系（CS）、数学系（MA）和信息系（IS）学生的姓名和性别。

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

- 可改写为：

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

对查询结果排序

- ORDER BY子句
  - 可以按一个或多个列排序
  - 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

- [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;
```

聚集函数 1

聚集函数	功能描述
COUNT(*)	统计元组个数
COUNT([DISTINCT|ALL] <列名>)	统计一列中值的个数
SUM([DISTINCT|ALL] <列名>)	计算一列值的总和（此列必须为数值型）
AVG([DISTINCT|ALL] <列名>)	计算一列值的平均值（此列必须为数值型）
MAX([DISTINCT|ALL] <列名>)	求一列值中的最大值
MIN([DISTINCT|ALL] <列名>)	求一列值中的最小值

聚集函数 2

- 聚集函数的处理对象是一个元组集合

- count 统计
  - count(*)：返回集合中的元组个数
  - count(colname)：返回在colname属性上取值非空的元组个数
  - count(distinct colname)：返回colname取值非空且互不相同的元组个数

- SUM, AVG, MAX, MIN 统计
  - 分别用于统计一组元组在某个属性上的取值的总和、平均值、最大值、最小值

聚集函数 3

❑空值（NULL）在聚集函数中的处理

- ➢在使用聚集函数对一个集合中的元素进行统计计算时，将忽略其中的‘空’值元素
  - ●在一个元组集合上执行如下的统计查询，它们返回的结果可能是不一样的！

    COUNT(*)　　COUNT(sd)　　COUNT(distinct sd)

  - ●在调用**SUM, AVG, MAX, MIN**聚集函数时，将首先剔除掉集合中取值为‘空’的元素，然后再进行统计计算
- ➢在一个空集上进行统计计算时，其返回结果如下：
  - ●**COUNT( )** 返回 **0**
  - ●其它的聚集函数均返回空值 **NULL**

聚集函数 4

- [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 SC15
WHERE Cno='1';
```

- [例3.45] 查询学生201215012选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM  SC,Course
WHERE Sno='201215012' AND
SC.Cno=Course.Cno;
```

对查询结果分组 1

- 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

- 聚集函数的使用：只能用在 SELECT子句 或 HAVING子句 中

对查询结果分组 2

- [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;
```

- 查询结果可能为：

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

- [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;
```

- HAVING短语与WHERE子句区别：
  - 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。

对查询结果分组 3

- **[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**
  - 因为WHERE子句中是不能用聚集函数作为条件表达式，下面的语句是不对的：

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

  - 正确的查询语句应该是：

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

数据管理基础

ch3.4.2 SQL数据查询（连接查询）

智能软件与工程学院

连接查询

- **连接查询**：同时涉及两个或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

等值连接/自然连接查询

- **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;
```

- [例3.50] 对[例3.49]用等值连接完成。

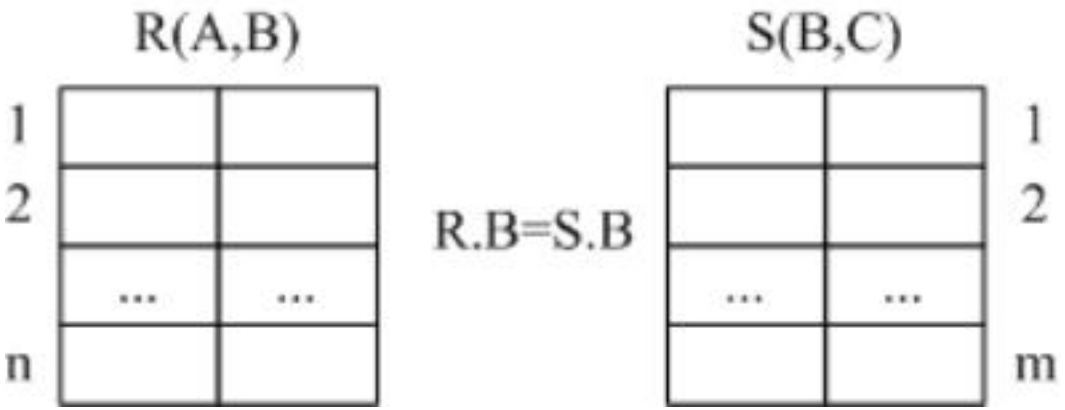
```
SELECT   Student.Sno,
     Sname, Ssex, Sage, Sdept,
     SC.Sno,  Cno, Grade
FROM    Student, SC
WHERE   Student.Sno  =  SC.Sno;
```

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
201215121	李勇	男	20	CS	201215121	1	92
201215121	李勇	男	20	CS	201215121	2	85
201215121	李勇	男	20	CS	201215121	3	88
201215122	刘晨	女	19	CS	201215122	2	90
201215122	刘晨	女	19	CS	201215122	3	80

连接操作的执行过程 1

❑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。



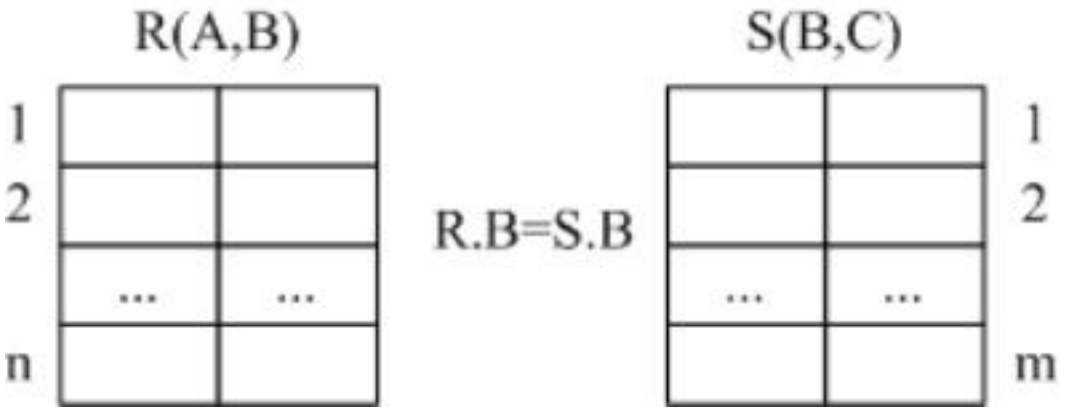
连接操作的执行过程 2

❑索引连接（INDEX-JOIN）

如果在其中一张表的连接字段上建有索引，那么可以用‘索引连接’加快连接操作的执行速度。

假设在表2中，连接字段上建有索引，那么基于连接字段的连接操作执行过程如下：

- 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



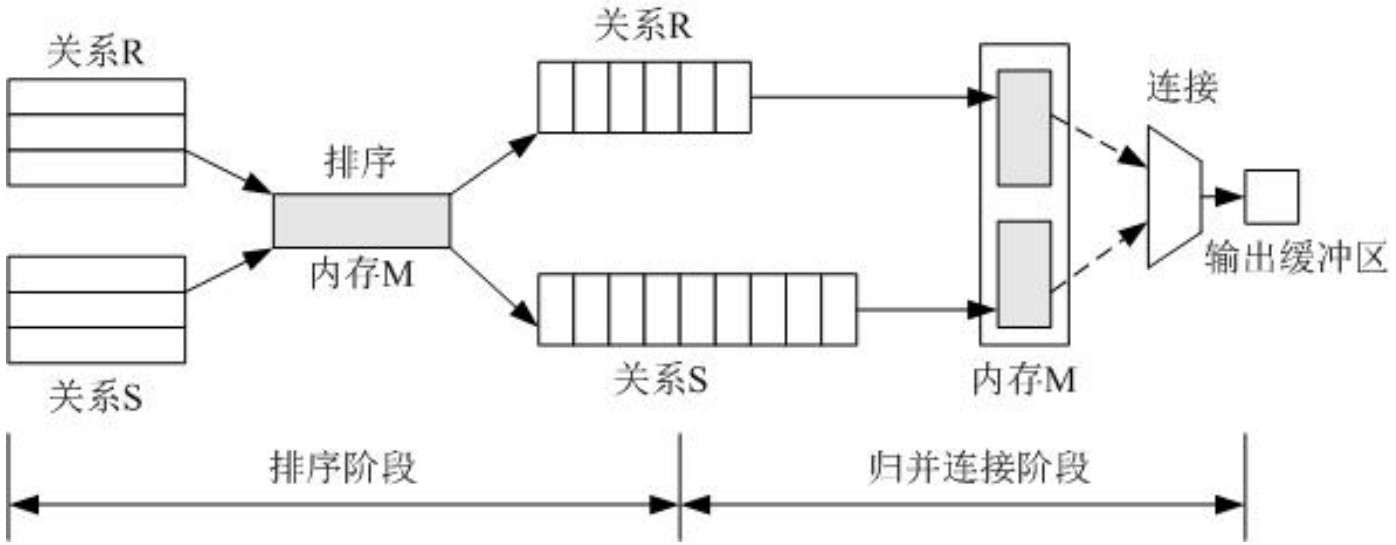
连接操作的执行过程 3

❑排序合并法（SORT-MERGE）

➢常用于“等值连接”

① 首先按连接字段对表1和表2排序；

② 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。



同时进行选择和连接

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

  - 执行过程：先从SC中挑选出Cno='2'且Grade>90的元组形成一个中间表，再和Student中满足连接条件的元组进行连接得到最终结果
  - 在实际执行过程中，会将上述两步组合形成‘嵌套循环’执行，并且以选课表SC作为外层循环表，对于满足条件的选课元组才会执行学生表Student上的内层循环

自身连接 1

- **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
  - 参与连接的两张表有不同的表名（或别名），但所有相应列的列名都是相同的，因此必须通过<表名>.<列名>的方式指定需访问的列

- [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 2

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

```
SELECT  FIRST.Cno, SECOND.Cpno
FROM  Course  FIRST, Course  SECOND
WHERE FIRST.Cpno = SECOND.Cno;
```

FIRST（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（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

查询结果

Cno	Pcno
1	7
3	5
5	6

外连接 1

- 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
      - 列出左边表中所有的元组
    - 右外连接
      - 列出右边表中所有的元组

外连接 2

- [例3.53] 用外连接改写[例3.49]

```
SELECT Student.Sno, Sname, Ssex, Sage, Sdept, Cno, Grade
FROM  Student  LEFT OUT JOIN SC ON (Student.Sno=SC.Sno);
```

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
201215121	李勇	男	20	CS	1	92
201215121	李勇	男	20	CS	2	85
201215121	李勇	男	20	CS	3	88
201215122	刘晨	女	19	CS	2	90
201215122	刘晨	女	19	CS	3	80
201215123	王敏	女	18	MA	NULL	NULL
201215125	张立	男	19	IS	NULL	NULL

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      /*多表连接*/
WHERE Student.Sno = SC.Sno
        AND SC.Cno = Course.Cno;
```

数据管理基础

ch3.4.3 SQL数据查询（嵌套查询）

智能软件与工程学院

嵌套查询概述 1

- 一个 SELECT-FROM语句 称为一个 查询块
- 将一个查询块嵌套在另一个查询块中的查询称为嵌套查询（一个查询块成为另一个查询块的组成成分）

```
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');
```

  - 最常见的是在WHERE子句、FROM子句中嵌入子查询
  - 有些数据管理系统也允许在SELECT子句、HAVING子句中嵌入子查询
  - 可以用创建的视图来代替FROM子句中的子查询

嵌套查询概述 2

- 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

```
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');
```

- 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
- 有些嵌套查询可以用连接运算替代
  - 谨慎使用嵌套查询

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
- 又被称为‘独立子查询’
- 执行过程：由里向外，逐层处理
  - 每个不相关子查询在上一级查询处理之前求解，其执行结果与父查询无关；
  - 不相关子查询的执行结果被用于建立其父查询的查找条件；
  - 在父查询的执行过程中，不需要再去执行它的不相关子查询。

相关子查询

- 子查询的查询条件依赖于父查询
- 执行过程：由外向里，嵌套循环
  - 首先取外层父查询中表的第一个元组t，根据它与内层子查询相关的列的值去处理内层的子查询。若内层子查询的执行结果能够使得外层父查询的WHERE条件表达式成立，则将此元组t放入结果表（用于投影生成最终的查询结果表）；
  - 然后再取父查询中表的下一个元组；
  - 重复上述过程，直至外层查询中表的元组全部检查完为止。

带有IN谓词的子查询 1

- [例3.55] 查询与“刘晨”在同一个系学习的学生。
  - 根据对查询语义的分析，可以先分步来完成（理解）此查询
  - 第一步：确定“刘晨”所在系名

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

结果为

Sdept
CS

  - 第二步：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

- ‘分步完成’查询不是数据库系统的方法！

带有IN谓词的子查询 2

- [例3.55] 查询与“刘晨”在同一个系学习的学生。

- 用‘IN+嵌套子查询’完成此查询

```
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept  IN (
      SELECT Sdept
      FROM Student
      WHERE Sname='刘晨');
   /* 此处为不相关子查询 */
```

- 用‘自身连接’完成此查询

```
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 3

❑[例3.55] 查询与“刘晨”在同一个系学习的学生。

❑ 使用 ‘IN + 不相关子查询’

```
SELECT  Sno, Sname, Sdept
FROM  Student
WHERE  Sdept  IN (
        SELECT  Sdept
        FROM  Student
        WHERE  Sname='刘晨');
```

❑ 使用 ‘IN + 相关子查询’

```
SELECT  x.Sno, x.Sname, x.Sdept
FROM  Student  x
WHERE  '刘晨'  IN (
        SELECT  y.Sname
        FROM  Student  y
        WHERE y.Sdept=x.Sdept);
```

❑也可以用‘相关子查询’来表示查询[例3.55]！

带有IN谓词的子查询 4

- [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')
);
/* 使用嵌套不相关子查询的执行过程 */
```

③ 最后在Student表中取出选修了3号课程的学生的Sno和Sname

② 然后在SC表中找出选修了3号课程的学生学号

① 首先在Course表中找出“信息系统”的课程号，为3号

用表连接查询 代替 IN谓词+子查询

- 用连接查询实现[例3.56]

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

- 巧用表的换名，可将该查询简写如下：

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

- 三张表之间的连接查询，常见的执行过程是三层嵌套循环连接！

多表连接操作的执行过程

- 嵌套顺序由DBMS来决定，与在FROM子句中的书写顺序无关，与WHERE子句中的条件定义顺序也无关

- 连接查询

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

- 可能执行过程

```
FOR c FROM ROWS 1 TO LAST OF Course
    if  (c.Cname = '信息系统')  {
        FOR sc FROM ROWS 1 TO LAST OF SC
            FOR s FROM ROWS 1 TO LAST OF Student
                if (s.Sno=sc.Sno  and  sc.Cno=c.Cno)
                {   /* 投影产生一条结果元组 */                }
            END FOR p
        END FOR o
    }
END FOR c
```

带有比较运算符的子查询 1

- 当能确切知道内层查询返回单值时，可用比较运算符>，<，=，>=，<=，!=或< >来代替IN谓词。（但该用法不是通用的！）
- 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');
```

- 说明：由于姓名Sname不是学生表Student的码，并不能保证此子查询一定返回单值，所以要慎用列与子查询之间的直接比较！

带有比较运算符的子查询 2

- [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

- 说明：
  - 在本例中，子查询肯定只返回一个统计结果（单值），可放心直接比较；
  - 即使子查询仅返回单值，也不是所有数据库系统都支持这种类型的表示方式，建议使用IN或带有SOME、ANY、ALL量词的量化比较！

[例3.57] 可能的执行过程

① 从外层查询中取出SC的一个元组x(201215121, 1, 92)，将元组x的Sno值传送给内层查询（用201215121代替内层查询中的x. Sno）

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

② 执行这个内层查询，得到值 88

③ 用该值代替内层查询，得到外层查询的WHERE子句：

```
WHERE   Grade >= 88;
```

④ 将元组x的Grade列值92代入上述的条件表达式进行判断。

- 条件成立，则按照SELECT子句的要求对元组x进行投影，得到一条结果元组(201215121, 1)
- 如果条件不成立，则忽略外层查询的当前元组x

❑ 不断重复上述步骤①至步骤④的过程，直到外层的SC元组全部处理完毕。最终的查询结果为：

Sno	Cno
201215121	1
201215121	3
201215122	2

带有 ANY（SOME）或 ALL 谓词的子查询 1

- 使用ANY或ALL谓词时必须同时使用比较运算（SOME与ANY是同义词）

谓词	语义	谓词	语义
> 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（）
!= ANY	不等于子查询结果中的某个值	!= ALL	不等于子查询结果中的任何一个值
<> ANY		<> ALL	

- ‘ = ALL ’与‘ != ANY ’通常没有实际使用价值。

带有 ANY（SOME）或 ALL 谓词的子查询 2

- [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' ;
```

/*父查询块中的条件 */

- 结果：

Sname	Sage
王敏	18
张立	19

- 执行过程：
  - 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}
  - 再处理父查询，找所有不是CS系且年龄 小于20 或 小于19 的学生

带有 ANY（SOME）或 ALL 谓词的子查询 3

- 用聚集函数实现[例 3.58]

```
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 4

- [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 5

表3.7 ANY(或SOME)、ALL谓词与聚集函数、IN谓词的等价转换

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

带有EXISTS谓词的子查询 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。
- **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 2

- [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表
  - 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  S
WHERE  EXISTS ( SELECT  *
                FROM  SC
                WHERE  SC.Sno = S.Sno  AND  Cno = '1');
```

带有EXISTS谓词的子查询 3

- [例3.61]查询没有选修1号课程的学生姓名。

```
SELECT  Sname
FROM     Student  S
WHERE NOT EXISTS (
                    SELECT  *
                    FROM    SC
                    WHERE SC.Sno=S.Sno  AND  Cno='1');
```

- 例3.60也可以用表连接查询来表示，但例3.61必须使用到子查询！

带有EXISTS谓词的子查询 4

❑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

- ➢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：

$$(\forall x)\, P \equiv \neg\left(\exists x\, (\neg\, P)\right)$$

带有EXISTS谓词的子查询 5

- [例3.55]查询与“刘晨”在同一个系学习的学生。

- IN + 子查询

```
SELECT  Sno, Sname, Sdept
FROM  Student
WHERE  Sdept  IN (
    SELECT  Sdept
    FROM     Student
    WHERE  Sname = ‘刘晨’ );
```

可以用‘EXISTS+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 6

- [例3.62] 查询选修了全部课程的学生姓名。
  - “选修了全部课程”等价于“不存在一门课程是该学生不选修的”

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

带有EXISTS谓词的子查询 7

- [例3.62] 查询选修了全部课程的学生姓名。
- 最里面的 NOT EXISTS 谓词可以改用 NOT IN 谓词

```
SELECT Sname
FROM Student S
WHERE NOT EXISTS (
    SELECT *
    FROM Course C
    WHERE NOT EXISTS (
        SELECT *
        FROM SC
        WHERE SC.Sno = S.Sno
            AND SC.Cno = C.Cno
    )
);
```

```
SELECT Sname
FROM Student S
WHERE NOT EXISTS (
    SELECT *
    FROM Course C
    WHERE C.Cno NOT IN (
        SELECT SC.Cno
        FROM SC
        WHERE SC.Sno = S.Sno
    )
);
```

带有EXISTS谓词的子查询 8

- [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：
  - 查询语义的解析：如果学号为x的学生满足查询条件，那么对所有的课程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 → q
    - 等价变换：

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

  - 变换后语义如下：

    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 8

- [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。
- 用NOT EXISTS谓词表示如下：

```
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 ) );
```

数据管理基础

ch3.4.4 SQL数据查询（集合操作）

智能软件与工程学院

集合查询

- 集合操作的种类
  - 并操作 UNION [ALL]
  - 交操作 INTERSECT [ALL]
  - 差操作 EXCEPT [ALL]

- 参加集合操作的各个子查询需要满足
  - 结果的列数必须相同
  - 对应列的数据类型也必须相同

并操作

- 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3.64]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;
```

- [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';
```

交操作 1

- [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 3.66] 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;
```

交操作 2

- [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3.67]也可以表示为：

```
SELECT  Sno
FROM    SC
WHERE  Cno='1' AND Sno IN (
        SELECT Sno
        FROM SC
        WHERE Cno='2');
```

差操作

- [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;
```

- [例3.68]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;
```

数据管理基础

ch3.4.5 SQL数据查询（基于派生表的查询）

智能软件与工程学院

基于派生表的查询1

- 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, avg_grade)
WHERE  SC.Sno = Avg_sc.avg_sno
       and  SC.Grade >= Avg_sc.avg_grade ;
```

基于派生表的查询 2

- 在FROM子句中，如果子查询的结果列都有列名且不存在同名现象，那么可以不用为该子查询对应的派生表指定列名，而是将子查询SELECT子句中结果列的列名作为派生表的缺省列名。

- [例3.60] 查询所有选修了1号课程的学生姓名

- 可以用如下查询完成：

```
SELECT  Sname
FROM    Student,
            (SELECT Sno FROM SC WHERE Cno=' 1 ')  AS  SC1
WHERE  Student.Sno = SC1.Sno;
```

复习思考题 1

1. 什么是子查询？什么是独立子查询？什么是相关子查询？它们之间的区别是什么？
2. 关系代数中的‘差’运算在SQL中可以用什么查询谓词来表示？是否可以不使用子查询？
3. 在关系代数中用‘除法’表示的查询，在SQL中如何表示？是否可以用SQL中的量化比较谓词ALL来描述‘除法’的语义？是否可以不使用子查询？
4. 关于表的连接查询，在SQL语言中可以有哪几种表示方法？

复习思考题 2

设有一个商品零售管理数据库MyCAP，关系模式如下：

关系	属性（列）	关系模式
客户	客户编号，姓名，所在城市，优惠折扣	customers(cid, cname, city, discnt)
供应商	供应商编号，名称，所在城市，销售提成比例	agents(aid, aname, city, percent)
商品	商品编号，名称，存放城市，库存数量，单价	products(pid, pname, city, quantity, price)
订单	订单编号，销售日期，客户编号，供应商编号，商品编号，销售数量，销售金额	orders(ordno, orddate, cid, aid, pid, qty, dols)

其中：

① cid, aid, pid, ordno分别是客户、供应商、商品、订单表的码；

② orddate是日期（DATE）类型的字段；

③ 可以根据ordno的大小来区分订单的先后，编号小的订单在前。

复习思考题 3

5. 请用SQL语言来查询满足下述条件的客户的cid和cname：

Q_1：没有通过纽约市的供应商去购买过商品；

Q_2：只购买过两种不同(pid)商品；

Q_3：所有订单的购买金额(dols)都超过1000美元；

Q_4：通过纽约市的供应商去购买过所有价格超过1美元的商品；

6. 请用SQL语言来表示下述查询 (结果返回cid和pid)

Q_5：客户C只购买过一次；

Q_6：客户C只购买过商品P 1 次；

Q_7：客户C只购买过商品P这一种商品；

7. 按照下述要求写出对应的SQL查询语句：

Q_8：在所有有客户的城市中都被销售过的商品的编号；

Q_9：返回每一个客户的编号及其最后两份订单的订购日期；

Q_{10}：查询满足下述条件的供应商编号和名称：向所有折扣discnt最高的客户都销售过商品；

Q_{11}：检索符合下述条件的商品的编号：至少有一个客户通过与该客户位于同一个城市的经销商订购过该商品；

Q_{12}：检索为居住在南京市的所有客户订购过同一种商品的经销商的编号；