

# 第3章 SQL之数据查询

# 大纲



- SQL概述
- 学生选课数据库
- 数据定义
- **数据查询**
- 数据更新
- 空值的处理
- 视图
- 本章小结

# 4.数据查询



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

## ■ 查询语句(SELECT)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]]...[(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>]];
```

- 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

# 4.数据查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

## ■ 本节主要内容：

- 单表查询
- 连接查询
- 嵌套查询
- 集合查询
- 基于派生表的查询

# 4.数据查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

## ■ openGauss之数据查询SQL

- 支持标准的SQL92/SQL99/SQL2003/SQL2011规范，支持GBK和UTF-8字符集，支持SQL标准函数与分析函数，支持存储过程。

- openGauss之SQL学习(课下自学)

  - <https://education.huaweicloud.com/courses/course-v1:HuaweiX+CBUCNXDR006+Self-paced/about>

- 华为贾军锋老师的材料(见补充)

# 4.1 单表查询

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- 单表查询指查询仅涉及一个表
  - 选择表中的若干列
  - 选择表中的若干元组
  - ORDER BY 子句
  - 聚集函数
  - GROUP BY子句
  - LIMIT子句

## 4.1.1 单表查询-选择表中的若干列

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 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;
```

### ■ 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;
```

## 4.1.1 单表查询-选择表中的若干列(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 查询经过计算的值

– SELECT子句的<目标列表表达式>可以是算术表达式、字符串常量、函数等

– [例3.19] 查询全体学生的姓名及其年龄

```
SELECT Sname, (extract(year from current_date) - extract(year from Sbirthdate)) "年龄"
FROM Student; --extract函数从日期或时间的数值里抽取子域，比如年、小时等
```

别名

– [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; -- 'Date of Birth:'是字面值(literal value)，即常量
```

| 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 |

| 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 |



## 4.1.2单表查询-选择表中的若干元组

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 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; ⇔ SELECT ALL Sno FROM SC;
```

– 指定DISTINCT关键词, 去掉表中重复的行

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

• 查询结果:

| Sno      |
|----------|
| 20180001 |
| 20180002 |
| 20180003 |
| 20180004 |
| 20180005 |

• 不存在重复的行

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 查询满足条件的元组
  - 通过WHERE子句实现

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                                 |

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### 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';
```

- RDBMS完成查询的实现途径：全表扫描或采用索引技术
- 字符串常数要用英文单引号括起来

- [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;
```

- [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;
```

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### 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、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;
```

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### 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');
```

## 4.1.2单表查询-选择表中的若干元组(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### 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)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
- \_ (下划线)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

– 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';
```

- 如果LIKE后面的匹配串不含通配符, 则可用=取代LIKE, 用!=或<>取代NOT LIKE。

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### – 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%' ;
```

- [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' ; --cno的长度为5个字符，所以中间用两个下划线
```

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

- [例3.33] 查询所有不姓刘学生的姓名、学号和性别

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

如果要查询的字符串本身包含通配符，则需用ESCAPE'<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'\'; --- ESCAPE后面的"\\"为换码字符
```

- [例3.35] 查询以 "DB\_" 开头，且倒数第三个字符为 i 的课程的具体情况

```
SELECT *  
FROM Course  
WHERE Cname LIKE 'DB\__%i__' ESCAPE '\ ' ;
```



## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### ⑤ 涉及空值(NULL)的查询

– 谓词: IS NULL 或 IS NOT NULL

- “IS”不能用“=”代替
- 空值和任何数据之间的比较是没有意义的。如果数学表达式中包含空值, 则结果都为NULL
- 空值与数字0和空字符串所代表的意义不同, 它代表的是未定义的值, 不占用存储空间的, 而0和空格是有意义的且占用存储空间

– [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; ----不能用=NULL
```

– [例3.37] 查询所有有成绩的学生学号和课程号

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

## 4.1.2单表查询-选择表中的若干元组(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### ⑥ 多重条件查询

- 逻辑运算符：AND和 OR来连接多个查询条件
  - 运算符的优先级：括号 > AND > OR
- [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，也可用以下语句实现

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

## 4.1.3单表查询-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RDBMS实现来决定
- [例3.39] 查询选修了81003号课程的学生学号及其成绩, 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;
```

## 4.1.4单表查询-聚集函数

### ■ 聚集函数

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

- 指定**DISTINCT**短语，表示取消指定列的重复值（去重）；指定ALL，表示允许重复值（默认值）
- 当聚集函数遇到空值时，除count(\*)外，都**跳过空值**而**只处理非空值**
- **count(\*)**是对元组进行计数，某个元组的一个或部分列取空值不影响其统计结果

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

## 4.1.4单表查询-聚集函数(cont.)

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 聚集函数示例

– [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';
```

– [例3.44] 查询选修81001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;
```

## 4.1.5单表查询-GROUP BY子句

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ GROUP BY子句分组

- 按指定的一列或多列值分组，**值相等**的为的一组
  - 如果**未对查询结果分组**，聚集函数将作用于**整个查询结果**
  - 对**查询结果分组**后，聚集函数将**分别作用于每个组**
- **分组的**目的之一：**细化聚集函数的作用对象**
  - 如果SQL中出现GROUP BY子句则该SQL中必须**出现至少一种聚集函数**
  - 否则，如果只有GROUP BY而**无聚集函数**则该SQL语句**一定是错误的**
- **GROUP BY**后面的子句**必须出现在SELECT的第一个位置**
- **[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;
```

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

## 4.1.5单表查询-GROUP BY子句(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [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'  
GROUP BY Sno HAVING COUNT(*) > 10;
```

- [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;
```

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

- HAVING短语与WHERE子句的区别

| 类型  | 作用对象  | 含义        |
|---|-------|-----------|
| WHERE子句   | 基表或视图 | 选择满足条件的元组 |
| HAVING短语  | 组     | 选择满足条件的组  |
| • RDBMS处理数据的顺序：先处理WHERE子句得到临时表 > 使用GROUP BY对临时表数据分组 > 对每组根据HAVING子句条件筛选出满足条件的数据 |       |           |

## 4.1.6单表查询-LIMIT子句

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- LIMIT子句用于限制SELECT语句查询结果的(元组)数量

- 格式: **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
    - 实际应用中LIMIT子句经常和ORDER BY子句一起使用

- [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;
```

- [例3.50] 查询平均成绩排名在3-7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;
```



## 4.2 连接查询

- **连接查询**：同时涉及两个以上表的查询
  - 连接查询是关系数据库中最常用的查询
- **连接条件或连接谓词**：用来连接两个表的条件
  - [**<表名1>.**]**<列名1>** **<比较运算符>** [**<表名2>.**]**<列名2>**
    - 比较运算符主要包括：=、>、<、>=、<=、!=(或<>)
  - [**<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

## 4.2连接查询(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

- 本节主要内容：
  - 等值与非等值连接查询
  - 自然连接查询
  - 复合条件连接查询
  - 自身连接查询
  - 外连接查询
  - 多表连接查询

## 4.2.1 等值与非等值连接查询

- 1. SQL概述
- 2. 学生选课数据库
- 3. 数据定义
- 4. 数据查询
- 5. 数据更新
- 6. 空值的处理
- 7. 视图
- 8. 本章小结

### ■ 等值连接查询

- 连接运算符为 “=” 的查询

### ■ 非等值连接查询

- 连接运算符不是 “=” 的查询
- [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;
```

| 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      |
| ...         | ...   | ...  | ...        | ...      | ...      | ...   | ...   | ...      | ...           |

## 4.2.1等值与非等值连接查询(cont.)

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

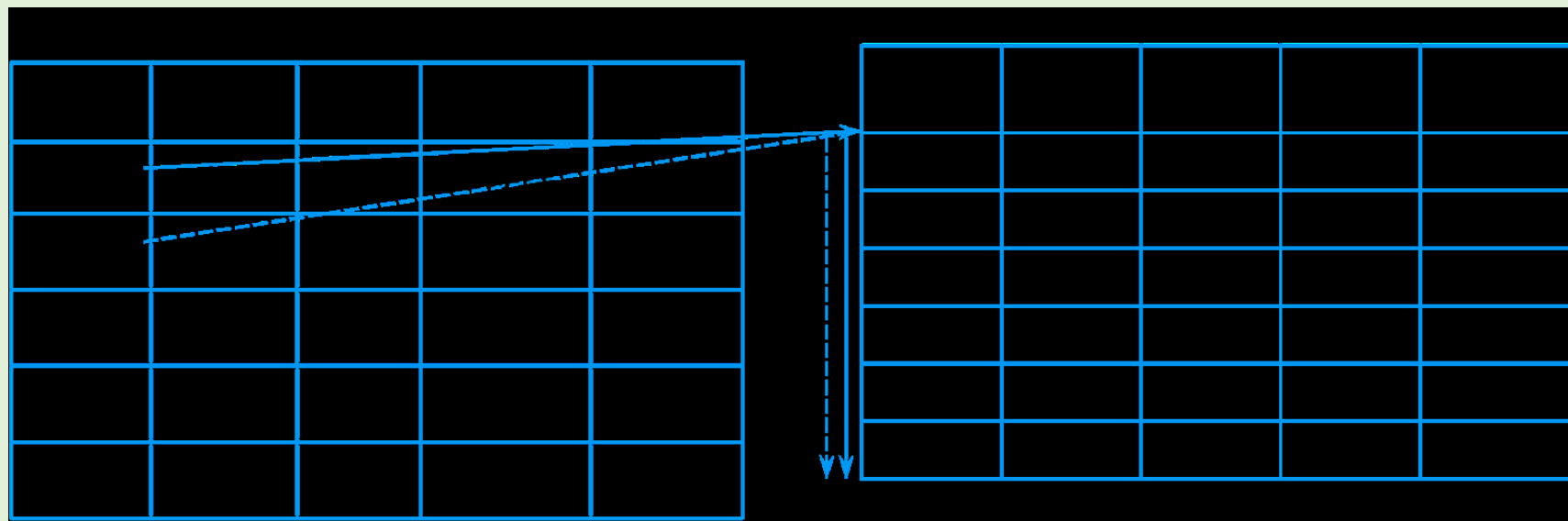
5.数据更新

6.空值的处理

7.视图

8.本章小结

- 连接操作的执行过程
  - 嵌套循环法(NESTED-LOOP)



- 索引连接(Index-Join)
  - 对SC表按连接字段建立索引
  - 对Student表中的每个元组，依次根据其连接字段值查询SC表的索引，从中找到满足条件的元组，找到后就将Student表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

## 4.2.2 自然连接查询

1. SQL概述

2. 学生选课数据库

3. 数据定义

4. 数据查询

5. 数据更新

6. 空值的处理

7. 视图

8. 本章小结

- 把结果表目标列中重复的属性列去掉的等值连接查询称为**自然连接查询**.

- 自然连接后的表头显示顺序: 

|      |          |          |
|------|----------|----------|
| 公共属性 | 第一张表剩余属性 | 第二张表剩余属性 |
|------|----------|----------|

- [例3.52] 查询每个学生的学号, 姓名, 性别, 出生日期, 主修专业及选课的课  
程号与成绩.

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

- 如何判断一个给定的SQL语句中的连接是等值连接还是自然连接?
  - 两张表有相同的属性名和数据类型, 使用**natural join**关键词表明

## 4.2.3 复合条件连接查询

- WHERE子句中有多个条件的连接查询称为复合条件连接查询。

- 复合条件由WHERE子句的连接谓词和选择谓词组成

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

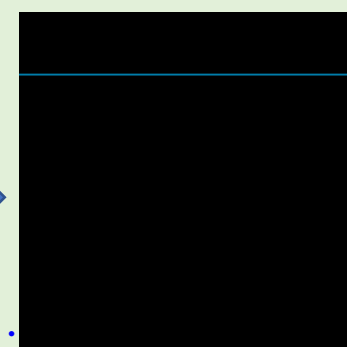
- 优化(高效)执行过程

- 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)

## 4.2.4 自身连接查询

- 一个表与其自己进行的连接称为**自身连接**.
  - 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;
```



**FIRST表(Course表)**

| 课程号<br>Cno | 课程名<br>Cname | 学分<br>Ccredit | 先修课<br>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<br>Cno | 课程名<br>Cname | 学分<br>Ccredit | 先修课<br>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       |

## 4.2.5外连接查询

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 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
  - 悬浮元组：连接时被舍弃的元组
- 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

### ■ 外连接类型

- (全)外连接、左外连接和右外连接

| 连接类型      | 特点  |
|-----------|---|
| 左外连接      | • Left outer join, 列出左边关系中所有的元组   |
| 右外连接      | • Right outer join, 列出右边关系中所有的元组  |
| 外连接       | • Outer join, full outer join<br>• 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<br>• 为左外连接与右外连接的并集 |
| (内)(普通)连接 | • Inner join, join<br>• 只输出满足连接条件的元组  |



## 4.2.5外连接查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

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

| 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    |
| 20180004    | 张立    | 男    | 2000-1-8   | 计算机科学与技术   | 81001 | 56    |
| 20180004    | 张立    | 男    | 2000-1-8   | 计算机科学与技术   | 81002 | 97    |
| 20180205    | 陈新奇   | 男    | 2001-11-1  | 信息管理与信息系统  | 81003 | 68    |
| 20180306    | 赵明    | 男    | 2000-6-12  | 数据科学与大数据技术 | NULL  | NULL  |
| 20180307    | 王佳佳   | 女    | 2001-12-7  | 数据科学与大数据技术 | NULL  | NULL  |

– 思考：若将上述LEFT OUTER JOIN改为RIGHT OUTER JOIN将是何种语义？

## 4.2.6多表连接查询

- 两个以上表的连接称为**多表连接**.

– [例3.56] 查询每个学生的学号、姓名、选修的课程名及成绩

```
SELECT S.Sno, Sname, Cname, Grade  
FROM Student S, SC, Course C      /*三张表的连接*/  
WHERE S.Sno=SC.Sno AND SC.Cno=C.Cno;
```

- openGauss的连接

<https://www.opengauss.org/zh/docs/3.0.0/docs/BriefTutorial/JOIN.html>

## 4.3 嵌套查询

- 1. SQL概述
- 2. 学生选课数据库
- 3. 数据定义
- 4. 数据查询
- 5. 数据更新
- 6. 空值的处理
- 7. 视图
- 8. 本章小结

### ■ 嵌套查询(nested query)概述

- 一个SELECT-FROM-WHERE (简写成S-F-W) 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 ');
```

- SQL语言允许多层嵌套查询  
但至多不超过3层

- 构造子查询应遵循的几个规则：
  - 子查询必须用括号括起来
  - 子查询的SELECT后面只能有一列
  - 子查询返回多行，只能与多值运算符一起使用，如IN运算符
  - 子查询不能用Order by子句，order by只能在主查询中使用，且至多1次

## 4.3 嵌套查询(cont.)

1. SQL概述

2. 学生选课数据库

3. 数据定义

4. 数据查询

5. 数据更新

6. 空值的处理

7. 视图

8. 本章小结

### ■ 子查询分类

- 不相关子查询：子查询的查询条件不依赖于父查询
- 相关子查询：子查询的查询条件依赖于父查询

### ■ 不相关子查询的求解方法

- 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。

### ■ 相关子查询的求解方法

- 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。

## 4.3嵌套查询(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

- 本小节主要内容：
  - 帶有IN谓词的子查询
  - 帶有比较运算符的子查询
  - 帶有ANY(SOME)或ALL谓词的子查询
  - 帶有EXISTS谓词的子查询

## 4.3.1 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

```
SELECT Sno, Sname, Smajor
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**

解法二：用自身连接

```
SELECT S1.Sno, S1.Sname, S1.Smajor
FROM Student S1, Student S2
WHERE S1.Smajor=S2.Smajor AND S2.Sname='刘晨';
```

## 4.3.1 带有IN谓词的子查询(cont.)



- 1. SQL概述
- 2. 学生选课数据库
- 3. 数据定义
- 4. 数据查询
- 5. 数据更新
- 6. 空值的处理
- 7. 视图
- 8. 本章小结

- [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'));
```

解法二：用连接查询

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

## 4.3.2带有比较运算符的子查询

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理


7.视图

8.本章小结

- 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
- [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



- (20180001,81002)
- (20180002,81002)
- (20180003,81001)
- (20180004,81003)
- (20180005,81003)

- x, y称为元组变量

- 求解相关子查询不能像求解不相干子查询那样，先一次将子查询求解出来，然后求解父查询。由于内层查询与外层查询有关，因此必须反复求值。



# 课堂练习



- 1. SQL概述
- 2. 学生选课数据库
- 3. 数据定义
- 4. 数据查询
- 5. 数据更新
- 6. 空值的处理
- 7. 视图
- 8. 本章小结

- 设有book表(类编号, 图书名, 出版社, 价格), 要求查询book表中所有大于同类图书平均价格的图书名称。

```
SELECT 图书名
FROM book a
WHERE 价格 > (SELECT avg(价格)
               FROM book b
               WHERE a.类编号=b.类编号);
```

## 4.3.3带有ANY(SOME)或ALL谓词的子查询



- 当使用ANY或ALL谓词时必须同时使用比较运算

| 运算符          | 语义              | 运算符   | 语义                   |
|--------------|-----------------|-------|----------------------|
| > ANY        | 大于子查询结果中的某个值    | >=ANY | 大于等于子查询结果中的某个值       |
| > ALL        | 大于子查询结果中的所有值    | >=ALL | 大于等于子查询结果中的所有值       |
| < ANY        | 小于子查询结果中的某个值    | <=ANY | 小于等于子查询结果中的某个值       |
| < ALL        | 小于子查询结果中的所有值    | <=ALL | 小于等于子查询结果中的所有值       |
| !=或<br><>ANY | 不等于子查询结果中的某个值   | =ANY  | 等于子查询结果中的某个值         |
| !=或<br><>ALL | 不等于子查询结果中的任何一个值 | =ALL  | 等于子查询结果中的所有值，通常无实际意义 |

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

## 4.3.3~(cont.)

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [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' ; --父查询块条件
```



```
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' ;
```

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 4.3.3~(cont.)

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [例3.61] 查询非计算机科学技术专业中比计算机科学技术专业**所有**学生年龄都小(出生日期晚)的学生的姓名和出生日期.

```
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' ; --父查询块条件
```



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

|  |  |
|--|--|
|  |  |
|  |  |
|  |  |

## 4.3.3~(cont.)

表3.6 ANY/SOME/ALL谓词与聚集函数、IN谓词的等价转换关系

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

- =ANY 等价于 IN谓词
- <ANY 等价于 <MAX
- <>ALL 等价于NOT IN谓词
- <ALL 等价于 <MIN

## 4.3.4带有EXISTS谓词的子查询

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ EXISTS谓词

- 存在量词  $\exists$
- 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。

- 可以利用EXISTS来判断  $x \in S, S \subseteq R, S = R, S \cap R$  非空是否成立。

## 4.3.4带有EXISTS谓词的子查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [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');
```

是否为相关子查询?

- 练习：请写出使用连接运算实现的SQL

- [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');
```

## 4.3.4带有EXISTS谓词的子查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 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

### ■ [例3.57解法4] 查询与“刘晨”在同一个主修专业的学生(用EXISTS谓词完成)

```
SELECT Sno, Sname, Smajor
FROM Student S1
WHERE EXISTS
    (SELECT *
     FROM Student S2
     WHERE S2.Smajor=S1.Smajor AND S2.name='刘晨' );
```

- 由于带EXISTS量词的相关子查询只关心内层查询是否有返回值，并不需要查具体值，因此其效率不一定低于不相关子查询，有时是高效的方法。



## 4.3.4带有EXISTS谓词的子查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 用EXISTS/NOT EXISTS实现全称量词(难点)

- SQL语言中没有全称量词 $\forall$ (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:

$$(\forall x)P \equiv \neg(\exists x(\neg P))$$

### ■ [例3.64] 查询选修了全部课程的学生姓名

转换为等价的用存在量词的形式: 查询这样的学生, 没有一门课程是他不选修的

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

## 4.3.4带有EXISTS谓词的子查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 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:  $p \rightarrow q \equiv \neg p \vee q$

### ■ [例3.65] 查询至少选修了学生20180002选修的全部课程的学生们的学号

#### – 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20180002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:  $(\forall y) p \rightarrow q$

## 4.3.4带有EXISTS谓词的子查询(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

### ■ 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: 不存在这样的课程y, 学生20180002选修了y, 而学生x没有选修

```
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 SCY.Cno=SCX.Cno)); /*与子查询中的SC表区分开*/
```

## 4.4集合查询

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- 集合操作是SQL的一个重要特点
- 集合操作分类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT ---openGauss差集操作符为MINUS，不是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
- 所有的集合操作具有相同的优先级，除非碰到括号

## 4.4集合查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [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;
```

- [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';
```

- **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**

## 4.4集合查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [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;
```

- [例3.68的另一解法] 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;
```

## 4.4集合查询(cont.)

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [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' ;
```

- [例3.69] 的另外一种表示

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

## 4.4集合查询(cont.)



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

- [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;
```

- [例3.70的另一种解法] 实际就是查询计算机科学系中年龄大于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;
```



## 4.5基于派生表的查询

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

### ■ 复杂查询

- 很难或根本不可能用一个查询块或几个查询块的并、交、差来解决的查询。

### ■ 基于派生表的查询

- 子查询出现在FROM子句中的查询
  - 此时子查询生成的临时派生表(Derived table)成为主查询的查询对象
  - 是表达复杂查询的一种方法
- 通常，FROM<子查询>表达的查询结果可用新的关系进行命名，包括属性的重新命名
- 用AS子句实现

## 4.5基于派生表的查询(cont.)



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- [例3.59的另一种解法] 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;
```

- 课堂练习：找出所有选课平均成绩超过80分的同学姓名（要求使用派生表实现）

```
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 >80;
```

- 小结：派生查询的典型应用场景：涉及相关子查询和聚集函数的使用。
- 处理方法：先用派生表得到聚集结果，然后再将派生表的结果与其他条件结合

## 4.5基于派生表的查询(cont.)

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

- 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的另一种解法]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

# SELECT语句的一般格式



1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

**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]];

# 目标列表表达式的可选格式



- 1.SQL概述
- 2.学生选课数据库
- 3.数据定义
- 4.数据查询
- 5.数据更新
- 6.空值的处理
- 7.视图
- 8.本章小结

## ■ 目标列表表达式格式:

- (1) \*
- (2) <表名>.\*
- (3) COUNT([DISTINCT|ALL]\* )
- (4) [<表名>.]<属性列名表达式>[,<表名>.]<属性列名表达式>]...

- 其中<属性列名表达式>可以是由属性列、作用于属性列的聚集函数和常量的任意算术运算 (+, -, \*, /) 组成的运算公式

# 聚集函数的一般格式

1.SQL概述

2.学生选课数据库

3.数据定义

4.数据查询

5.数据更新

6.空值的处理

7.视图

8.本章小结

**COUNT**  
**SUM**  
**AVG**  
**MAX**  
**MIN**

**[DISTINCT|ALL] <列名>)**

# WHERE子句的条件表达式的可选格式

1. SQL概述

2. 学生选课数据库

3. 数据定义

4. 数据查询

5. 数据更新

6. 空值的处理

7. 视图

8. 本章小结

- (1)
- $$\langle \text{属性列名} \rangle \theta \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ [\text{ANY}|\text{ALL}] (\text{SELECT语句}) \end{array} \right\}$$
- (2)
- $$\langle \text{属性列名} \rangle [\text{NOT}] \text{BETWEEN} \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ (\text{SELECT语句}) \end{array} \right\} \text{AND} \left\{ \begin{array}{l} \langle \text{属性列名} \rangle \\ \langle \text{常量} \rangle \\ (\text{SELECT语句}) \end{array} \right\}$$
- (3)
- $$\langle \text{属性列名} \rangle [\text{NOT}] \text{IN} \left\{ \begin{array}{l} (\langle \text{值1} \rangle [, \langle \text{值2} \rangle] \dots) \\ (\text{SELECT语句}) \end{array} \right\}$$
- (4)  $\langle \text{属性列名} \rangle [\text{NOT}] \text{LIKE} \langle \text{匹配串} \rangle$
- (5)  $\langle \text{属性列名} \rangle \text{IS} [\text{NOT}] \text{NULL}$
- (6)  $[\text{NOT}] \text{EXISTS} (\text{SELECT语句})$
- (7)
- $$\langle \text{条件表达式} \rangle \left\{ \begin{array}{l} \text{AND} \\ \text{OR} \end{array} \right\} \langle \text{条件表达式} \rangle \left\{ \begin{array}{l} \text{AND} \\ \text{OR} \end{array} \right\} \langle \text{条件表达式} \rangle \dots$$

# 课堂练习

- 关系R包含A, B, C三个属性:

| A  | B    | C    |
|----|------|------|
| 10 | NULL | 20   |
| 20 | 30   | NULL |

- 写出对查询语句SELECT \* FROM R WHERE X; 当X为下列条件的查询结果
  - A IS NULL
  - $A > 8$  AND  $B < 20$
  - $C + 10 > 25$
  - EXISTS (SELECT B FROM R WHERE  $A = 10$ )
  - C IN (SELECT B FROM R)