# 大学计算机基础

哈尔滨工业大学

2011

# 第8章 信息管理与数据库

1. 数据库系统的基本概念
2. 关系模型与关系数据库
3. 关系数据库标准语言----SQL
4. 关系数据库设计初步*
5. 典型数据库管理系统介绍

# 关系数据库标准语言----SQL

SQL是集DDL、DML、DCL于一体的语言,有8种语句

- 定义(Definition)　　CREATE(创建)　DROP(撤消)
- 操纵(Manipulation)　　INSERT(插入)　UPDATE(更改)　DELETE(删除)
- 查询(Query):　　SELECT(查询)
- 控制(Control)　　GRANT(授权)　REVOKE(收回授权)

# 用英语词汇按结构化的方式表达查询
# SQL----Structural Query Language

$\pi_{姓名,课程名}(\sigma_{课程号=c2}(R \bowtie S))$

表达为：

```
SELECT  姓名,课程名
  FROM  R , S
 WHERE  课程号=c2
        AND  R.学号 = S.学号
```

R=9811班学生

| 学号 | 姓名 | 年龄 | 住址 |
|---|---|---|---|
| 981101 | 李四 | 22 | 3010 |
| 981103 | 李三 | 21 | 3011 |
| 981105 | 李六 | 22 | 3011 |

S=选课表

| 学号 | 课程号 | 课程名 | |
|---|---|---|---|
| 981101 | C1 | 计算机 | 教1 |
| 981105 | C2 | 物理 | 教2 |
| 981101 | C3 | 高数 | 教5 |
| 981101 | C4 | 化学 | 教5 |
| 981103 | C2 | 物理 | 教2 |

# SQL数据库操作的步骤

1） 定义数据库结构，定义关系模式；

| 班级 | 课程 | 教师 | 学期 | 学号 | 姓名 | 成绩 |
|---|---|---|---|---|---|---|
| | | | | | | |

数据格式

2） 向已定义的数据库中添加、删除和修改数据；

数据格式

| 班级 | 课程 | 教师 | 学期 | 学号 | 姓名 | 成绩 |
|---|---|---|---|---|---|---|
| 981101 | 数据库 | 李四 | 98秋 | 01 | 张三 | 100 |
| 981101 | 数据库 | 李四 | 98秋 | 02 | 张四 | 90 |
| 981101 | 数据库 | 李四 | 98秋 | 03 | 张五 | 80 |
| 981101 | 计算机 | 李五 | 98秋 | 01 | 张三 | 89 |
| 981101 | 计算机 | 李五 | 98秋 | 02 | 张四 | 98 |
| 981101 | 计算机 | 李五 | 98秋 | 03 | 张五 | 72 |
| 981102 | 数据库 | 李四 | 99秋 | 01 | 王三 | 30 |
| 981102 | 数据库 | 李四 | 99秋 | 02 | 王四 | 90 |
| 981102 | 数据库 | 李四 | 99秋 | 03 | 王武 | 78 |

数据

981102 数据库 李四 99秋 02 王四 90

3） 对数据库进行各种查询与统计

# 定义数据表---CREATE TABLE语句

CREATE TABLE 表名（列名1 类型 [NOT NULL]
[，列名2 类型 [NOT NULL]]）；

例 定义学生表，（学号，姓名，年级，专业），要求属性“学号”不允许空值。

**CREATE TABLE 学生 (学号 char (6) not null，**
**姓名 char (10)，**
**年级 char (4)，**
**专业 char (30))；**

学生

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| | | | |

# 插入数据---INSERT INTO语句

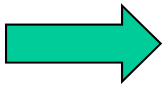
INSERT INTO 表名[(列名1，...，列名n)]
VALUES (对应列名1的值，...，对应列名n的值);

例　向“学生”表中插入一条记录的数据。

**INSERT INTO 学生**
**VALUES (‘890183’，‘张文青’，‘89’，‘软件’);**

（学生）操作前

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王磊 | 90 | 软件 |
| 918327 | 刘玉 | 91 | 应用 |



（学生）操作后

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王磊 | 90 | 软件 |
| 918327 | 刘玉 | 91 | 应用 |
| 890183 | 张文清 | 89 | 软件 |

# 修改数据---UPDATE语句

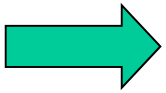
UPDATE 表名
SET 列名1=表达式1[, 列名2=表达式2[, ……]]
[WHERE 条件];

例　将“选课”表中的1001号课的所有成绩提高5分。

```
UPDATE 选课
SET 成绩=成绩+5
WHERE 课号=‘1001’;
```

（选课）操作前

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |



（选课）操作后

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 90 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 77 |
| 902783 | 2001 | |
| 903829 | 1001 | 87 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 92 |

# 删除数据---DELETE语句

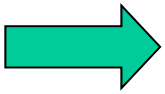
DELETE FROM 表名 [WHERE 条件];

例 从“必修课”表中将2002号课去掉

**DELETE FROM 必修课 WHERE 课号=‘2002’;**

（必修课）操作前

| 课号 | 必修专业 |
|---|---|
| 1001 | 软件 |
| 1001 | 应用 |
| 1002 | 软件 |
| 2001 | 软件 |
| 2001 | 应用 |
| 2002 | 应用 |



（必修课）操作后

| 课号 | 必修专业 |
|---|---|
| 1001 | 软件 |
| 1001 | 应用 |
| 1002 | 软件 |
| 2001 | 软件 |
| 2001 | 应用 |

# 数据库查询----SELECT语句基本格式

基本子句

SELECT [DISTINCT] 列名1 [，列名2...]

Where子句

FROM 表名1[，表名2...]

[WHERE 条件1]

[GROUP BY 列名i1

Having子句

GroupBy子句 [，列名i2 ...] [HAVING 条件2] ]

[ORDER BY 表达式1 [ASC / DESC] ...]

OrderBy子句

基本格式简单，但其功能丰富

# 简单条件查询---查询若干列

## 例1　列出全部学生的学号和姓名

```
SELECT 学号，姓名
FROM 学生；
```

（学生）操作前

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王磊 | 90 | 软件 |
| 918327 | 刘玉 | 91 | 应用 |

查询结果

| 学号 | 姓名 |
|---|---|
| 890237 | 陈莉 |
| 902783 | 李玉刚 |
| 903829 | 王磊 |
| 918327 | 刘玉 |

# 简单条件查询---查询全部列

例2　列出全部学生的信息

**SELECT** 学号，姓名，年级，专业
**FROM** 学生；

或者

**SELECT** *
**FROM** 学生；

注：“*”代表表格中的所有列

查询结果

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈　莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王　磊 | 90 | 软件 |
| 918327 | 刘　玉 | 91 | 应用 |

# DISTINCT的使用-去掉重复记录

## 例3 列出所有必修课的课号

```
SELECT DISTINCT 课号
FROM    必修课;
```

（必修课）操作前

| 课号 | 必修专业 |
|---|---|
| 1001 | 软件 |
| 1001 | 应用 |
| 1002 | 软件 |
| 2001 | 软件 |
| 2001 | 应用 |
| 2002 | 应用 |

未采用 DISTINCT 的查询结果

| 课号 |
|---|
| 1001 |
| 1001 |
| 1002 |
| 2001 |
| 2001 |
| 2002 |

采用 DISTINCT 的查询结果

| 课　号 |
|---|
| 1001 |
| 1002 |
| 2001 |
| 2002 |

# 查询条件的书写---WHERE子句

- ✓每一行都要检查满足与否的条件要用**WHERE**子句表达
- ✓**Where**子句中包含多个查询条件时，使用**AND**、**OR**和括号串接
  - **AND**：两端的查询条件都必须满足
  - **OR**：两端的查询条件有一个满足即可
  - 括号：改变逻辑计算的顺序，括号内的优先计算

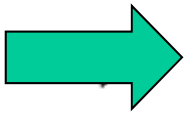
# 简单条件查询

例4 列出“软件”专业全部学生的学号及姓名

```
SELECT  学号，姓名
FROM    学生
WHERE   专业＝‘软件’；
```

（学生）操作前

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王磊 | 90 | 软件 |
| 918327 | 刘玉 | 91 | 应用 |



查询结果

| 学号 | 姓名 |
|---|---|
| 890237 | 陈莉 |
| 903829 | 王磊 |

# 范围区间处理--“既...又...”查询

选课

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | 0 |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |

例5 查找成绩在70到80分之间的学生选课得分情况

```
SELECT *
  FROM  选课
 WHERE  成绩>=70 AND 成绩<=80;
```

# “或者...或者...”查询

例6 列出89级和91级全体学生的学号和姓名

学生

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈　莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王　磊 | 90 | 软件 |
| 918327 | 刘　玉 | 91 | 应用 |

```
SELECT  学号，姓名
  FROM  学生
WHERE   年级 = '89' OR 年级 = '91';
```

# 多条件综合查询

选课

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |

例7 求选修了“1001”号课程，而且成绩或者大于80分或者小于60分的学生学号及成绩

```
SELECT   学号，成绩
  FROM   选课
WHERE   课号 = '1001' AND
          (成绩>80 OR 成绩<60);
```

注意括号的使用，有无括号含义不同

# 同一字段“既...又...”查询与“或者...或者...”查询的区别

选课

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |

例8 或者学过“1001”号课程或者学过“2002”号课程的同学学号

```
SELECT 学号 FROM 选课
 WHERE 课号= '1001' OR 课号='2002';
```

例9 既学过“1001”号课程又学过“2002”号课程的同学学号

```
SELECT 学号 FROM 选课
 WHERE 课号= '1001' AND 课号='2002';
```

```
SELECT 学号 FROM 选课
 WHERE 课号= '1001'
   AND 学号 IN
      (SELECT 学号 FROM 选课
       WHERE 课号= '2002');
```

# 模糊查询:LIKE

例10 列出姓王的所有学生

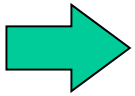
```
SELECT *
FROM 学生
WHERE 姓名 LIKE '王%';
```

列出姓李，并且全名为三个字的学生

```
SELECT *
FROM 学生
WHERE 姓名 LIKE '李_ _';
```

（学生）

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 890237 | 陈莉 | 89 | 软件 |
| 902783 | 李玉刚 | 90 | 应用 |
| 903829 | 王磊 | 90 | 软件 |
| 918327 | 刘玉 | 91 | 应用 |



查询结果

| 学号 | 姓名 | 年级 | 专业 |
|---|---|---|---|
| 903829 | 王磊 | 90 | 软件 |

下划线“_”：表示任意匹配一个字符

百分号“%”：表示匹配0个或多个字符的字符串

# ORDER BY的使用－显示结果的排序

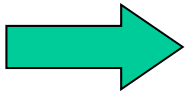
例11　求“1001”号课成绩大于80分的学生学号及成绩，并按成绩由高到低列出

```
SELECT    学号，成绩
FROM      选课
WHERE     课号＝‘1001’ and 成绩>80
ORDER BY  成绩  DESC;
```

ASC 代表升序排列
DESC代表降序排列

（选课）

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |



查询结果

| 学号 | 成绩 |
|---|---|
| 918327 | 87 |
| 890237 | 85 |
| 903829 | 82 |

注意:ORDER BY语句始终写在SQL语句的最后

# 嵌套查询

## 例12 列出选修“汇编语言”的所有学生的学号及成绩

```
SELECT 学号，成绩
  FROM  选课
WHERE  课号 IN (SELECT  DISTINCT 课号
                FROM  课程
                WHERE 课名=“汇编语言”)
```

→查询“汇编语言”课的课号

子查询结果

| 课号 |
|---|
| 1002 |

| 号 | 课名 | 开 |
|---|---|---|
| 1 | 程序设计 | 王 |

子查询结果

| 课号 |
|---|
| 1002 |

| 学分 |
|---|
| 4 |

最终查询结果

| 学号 | 成绩 |
|---|---|
| 890237 | 78 |
| 903829 | 83 |

（选课）

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |

再看例9

# 多表查询：连接条件的书写

例13 列出选修“1001”号课的学生姓名及成绩

**SELECT** 姓名，成绩
**FROM** 学生，选课
**WHERE** 学生.学号=选课.学号 **AND** 课号＝’1001’;

```
Select name,score
From student,khao
Where no in ( select no from sel_course where
khao='1001')
```

# 数据库统计操作—集函数

- MIN( ) ---- 求（字符、日期、数值）的最小值
- MAX( ) ----求（字符、日期、数值）的最大值
- COUNT ( ) ----计算所选数据的行数
- SUM ( ) ---- 计算数值列的总和
- AVG ( ) ---- 计算数值列的平均值

# 统计操作，GROUP BY子句用法

选课

| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | 0 |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |

例14 求总平均成绩

```
SELECT AVG(成绩) FROM 选课;
```

例15 求各门课的平均成绩

```
SELECT 课号, AVG(成绩)
FROM 选课
GROUP BY 课号;
```

| 课号 | 平均成绩 |
|---|---|
| 1001 | 81.5 |
| 1002 | 80.5 |
| 2002 | 75 |
| 2001 | 0 |

例16 求每个同学的平均成绩

```
SELECT 学号, AVG(成绩)
FROM 选课
GROUP BY 学号;
```

| 学号 | 平均成绩 |
|---|---|
| 890237 | 69.3 |
| 902783 | 36 |
| 903829 | 82.5 |
| 918327 | 87 |

# HAVING子句的使用

选课

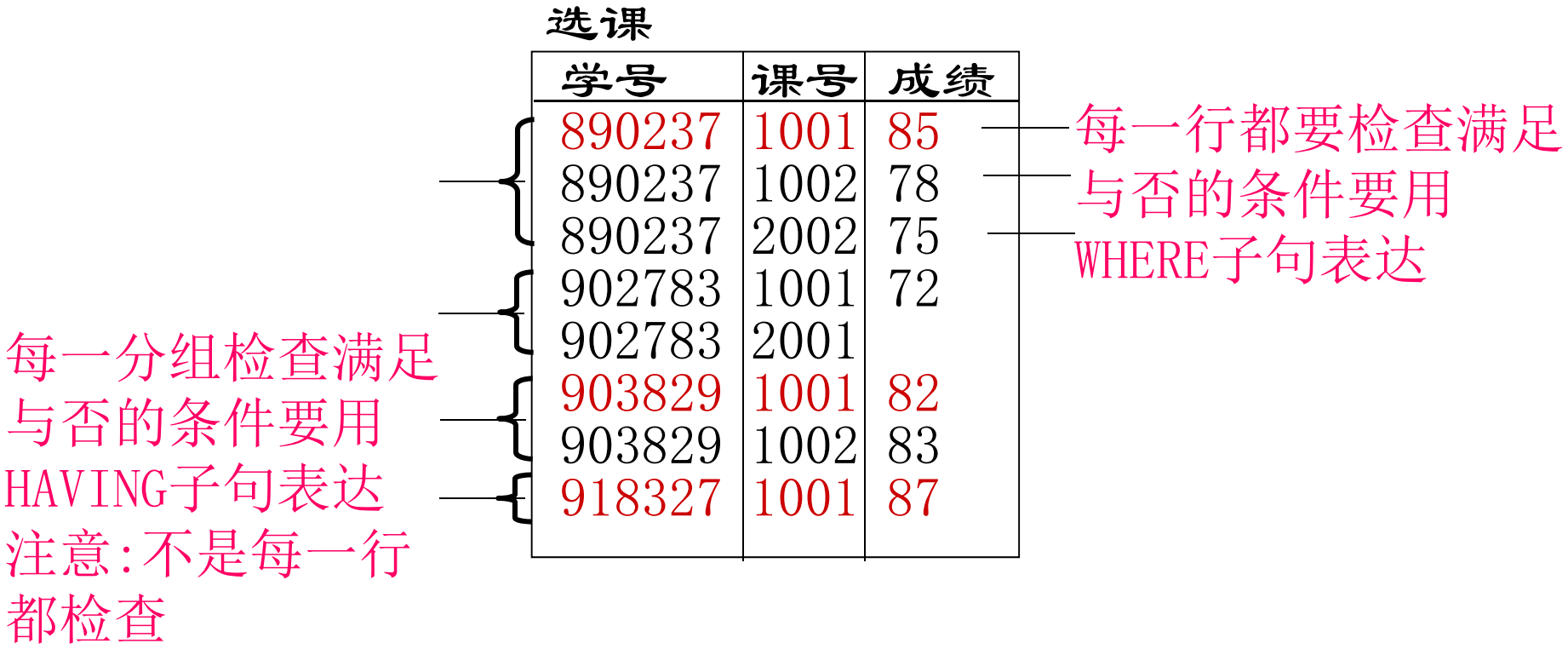
| 学号 | 课号 | 成绩 |
|---|---|---|
| 890237 | 1001 | 85 |
| 890237 | 1002 | 78 |
| 890237 | 2002 | 75 |
| 902783 | 1001 | 72 |
| 902783 | 2001 | |
| 903829 | 1001 | 82 |
| 903829 | 1002 | 83 |
| 918327 | 1001 | 87 |

例17 列出最少选修了三门课程的学生姓名

```
SELECT  姓名
  FROM  学生
 WHERE  学号 IN
      (SELECT 学号
         FROM 选课
        GROUP BY 学号
        HAVING COUNT(*) >= 3);
```

注：COUNT的特殊形式是COUNT(*)，用于统计满足WHERE子句中逻辑表达式的元组行数。

# HAVING子句与WHERE子句表达条件的区别




HAVING子句一定在GROUP BY子句中使用

# SQL训练(1) 检索条件书写训练示例

S(Sno, Sname, Sage, Sclass, Saddr)
C(Cno, Cname, Cteacher)
SC(Sno, Cno, Scgrade)

例1：检索年龄小于18岁的所有学生的姓名和其家庭住址

```
Select  Sname, Saddr
From  S
Where  Sage<18
```

例2：检索年龄小于18岁或者年龄大于25岁的所有学生姓名及其家庭住址

```
Select  Sname, Saddr
From  S
Where  Sage<18  or  Sage>25
```

例3：检索出010201班年龄小于18岁或者年龄大于25岁的所有学生姓名及其家庭住址

```
Select  Sname, Saddr
From  S
Where  Sclass = '010201'  and  (Sage<18  or  Sage>25)
```

# SQL训练(2) 结果唯一性训练示例：

S(Sno, Sname, Sage, Sclass, Saddr)
C(Cno, Cname, Cteacher)
SC(Sno, Cno, Scgrade)

例4：检索出成绩高于80分的所有学生的学号

```
Select  Sno
From  SC
Where  Scgrade > 80

Select  Distinct  Sno
From  SC
Where  Scgrade > 80
```

# SQL训练(3) 结果排序训练示例：

S(Sno, Sname, Sage, Sclass, Saddr)
C(Cno, Cname, Cteacher)
SC(Sno, Cno, Scgrade)

例5：按学号由小到大的顺序显示出所有学生的信息

```
Select *
From S
Order By Sno ASC
```

例6：按学号由小到大的顺序显示出010201班所有学生的信息

```
Select *
From S
Where Sclass = '010201'
Order By Sno ASC
```

例7：按“1”号课成绩由高到低的顺序显示出所有学生的学号

```
Select Sno
From SC
Where Cno = '1'
Order By Scgrade DESC
```

# SQL训练(4) 多表连接训练示例：

S(Sno, Sname, Sage, Sclass, Saddr)
C(Cno, Cname, Cteacher)
SC(Sno, Cno, Scgrade)

例8：按“1”号课成绩由高到低的顺序显示出所有学生的姓名(二表连接)

```
Select  Sname
From  SC, S
Where  Cno = '1'  and  SC.Sno = S.Sno
Order  By  Scgrade  DESC
```

例9：按“计算机原理”课成绩由高到低的顺序显示出所有学生的姓名(三表连接)

```
Select  Sname
From  SC, S, C
Where  Cname = '计算机原理'  and  SC.Sno = S.Sno
          and  SC.Cno = C.Cno
Order  By  Scgrade  DESC
```

# SQL训练(5) 多表连接训练示例:

S(Sno, Sname, Sage, Sclass, Saddr)
C(Cno, Cname, Cteacher)
SC(Sno, Cno, scgrade)

例10：列出“1”号课成绩比“2”号课成绩高的所有学生的学号(同一表的二次连接)

```
Select  X1.Sno
From  SC  X1, SC  X2
Where  X1.Cno = '1'  and  X1.Sno = X2.Sno
          and  X2.Cno='2'  and  X1.Scgrade > X2.scgrade
```

例11：列出“计算机原理”课成绩比“张三”同学成绩高的所有学生的姓名

```
Select  S1.Sname
From  SC  X1, SC  X2, C, S  S1, S  S2
Where  C.Cname = '计算机原理'  and X1.Cno = C.Cno
          and  X1.Cno = X2.Cno   and  X2.Sno=S2.Sno
          and  S2.Sname='张三' and  X1.Scgrade > X2.Scgrade
          and  S1.Sno=X1.Sno
```