

## 3.1 SQL概述

## 3.2 学生-课程数据库

## 3.3 数据定义

## 3.4 数据查询

## 3.5 数据更新

## 3.6 空值的处理

## 3.7 视图



# 数据查询

- 查询语句基本结构

```
SELECT A1,A2,...An  
FROM r1,r2,...rm  
WHERE P
```

其中：A<sub>i</sub>表示属性，r<sub>i</sub>表示关系，P是一个谓词

- 查询语句语义

等价于关系代数表达式  $\pi_{A_1,A_2,\dots,A_n}(\sigma_P(r_1 \times r_2 \times \dots \times r_m))$

从表r<sub>1</sub>,r<sub>2</sub>,...r<sub>m</sub>中，查找出满足P的元组的属性A<sub>1</sub>,A<sub>2</sub>,...A<sub>n</sub>

select 对应投影运算，用来列出查询结果；

from 对应笛卡尔乘积，列出求值时需要的关系表；

where 对应选择运算，包括作用于from中关系的谓词。



# 数据查询

- 如何学习SELECT语句?
  - 学习代数运算符所对应的SELECT
  - 将解决问题的代数思维过程用SELECT语句写出
- DBMS构造SELECT过程
  - 1) from子句 首先构造from子句, 确定操作对象
  - 2) where子句 其次构造where子句, 从操作对象中选取元组
  - 3) select子句 最后构造select子句, 确定输出关系模式



# SELECT构造过程示例

[例] 查询选修2号课程且成绩在90分以上的所有学生学号和姓名

```
SELECT Student.Sno, Sname
FROM Student, SC
WHERE Student.Sno = SC.Sno AND SC.Cno=2 AND SC.Grade > 90;
该SELECT语句等价于
```

$\pi_{\text{Student.Sno, Sname}}(\sigma_{\text{SC.Cno='2'} \wedge \text{SC.Grade} > 90}(\text{student} \bowtie \text{SC}))$

转化过程

$\text{student} \bowtie \text{SC}$  被转化成

$\text{FROM Student, SC where Student.Sno = SC.Sno}$

$\sigma_{\text{SC.Cno='2'} \wedge \text{SC.Grade} > 90}$  被转换成

$\text{where SC.Cno='2' AND SC.Grade} > 90$

$\pi_{\text{Student.Sno, Sname}}$  被转换成

$\text{SELECT Student.Sno, Sname}$

构造过程

DBMS首先将Student和SC调入内存进行笛卡尔乘，再选择元组，最后投影



# 单表查询

- 为简单起见，首先学习查询仅涉及一个表的单表查询SQL语句，再学习多表查询SQL语句
- 单表查询SQL语句学习内容
  - 投影运算对应的SQL (SELECT子句)  
选择表中的若干列
  - 选择运算对应的SQL语句 (WHERE子句)  
选择表中的若干元组
  - 排序运算对应的SQL语句 (ORDER BY子句)  
对表进行排序运算
  - 聚集函数对应的SQL语句 (SUM (), COUNT ())  
对分组后元组进行集函数运算
  - 分组运算对应的SQL语句 (GROUP BY子句)  
对表进行分组运算



➤图为学生-课程数据库中的student关系、Course关系、SC关系

SC:

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	90

Student:

Sno	Sname	Ssex	Sage	Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

Course:

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

# 选择表中的若干列

- 如何查询一个关系的指定列？

在关系代数中使用运算符 $\pi$ ，在SQL中使用SELECT

[例] 查询全体学生的学号与姓名。

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

- 若需选出所有属性列时如何办？

在SELECT关键字后面指定为 \*，或列出所有列名

[例] 查询全体学生的详细记录。

```
SELECT Sno, Sname, Ssex, Sage, Sdept  
FROM Student;  
SELECT *  
FROM Student;
```



# 重复元组

- 问题

对一个关系SELECT后，结果关系中可能出现重复元组，实现中，为提高效率，SELECT并不消除重复元组。

[例] 查询全体学生的成绩。

```
SELECT grade
```

```
FROM SC;
```

结果中可以有多多个相同元组

- 解决

对应于关系代数的去重操作，SQL提供了distinct解决去重问题

[例] 

```
SELECT distinct grade
```

```
FROM SC;
```

如果没有指定DISTINCT关键词，则缺省为ALL





# 查询经过计算的值

- **问题**: 若希望的查询结果表中属性无法直接用SELECT得出, 但可以通过运算得出, 如何处理这种派生属性?
- **解决**: 对应于关系代数中的广义投影运算, 在SELECT子句中, 其目标表达式可以为:  
算术表达式、字符串常量、函数、列别名

**[例] 查询学生的姓名、出生年份和系, 要求用小写字母表示系名**

```
SELECT Sname, 'Year of Birth: ', 2022-Sage, ISLOWER(Sdept)
FROM Student;
```

**输出结果:**

Sname	'Year of Birth:'	无列名	无列名
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



# 更名运算

- **问题：**为方便起见，关系代数提供了命名运算，可以对结果关系重新命名，SQL中如何处理重命名？
- **解决：**SQL中提供了别名功能，可以对属性、表重新命名。

[例] 使用列别名（更名运算）改变查询结果的列标题：

```
SELECT Sname NAME, 'Year of Birth: ' BIRTH,  
       2022-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

注 更名运算也可作用在一张表上

[例] SELECT Sname FROM Student S, SC WHERE S.Sno=SC.Sno



# where子句

- 如何从表中选择指定元组？

对应于关系代数运算 $\sigma_p$ ，SQL提供where子句解决表元组的选择。对一张表实施where相当于做选择。

- 格式：

WHERE <条件表达式>

<条件表达式>是包含属性名的逻辑表达式P，通过P对元组进行筛选。

查 询 条 件	谓 词
比 较	=, >, <, >=, <=, !=, <>, !>, !<; NOT
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



# 确定范围

- 谓词: 关系运算符 <、>.....  
BETWEEN ... AND ...  
NOT BETWEEN ... AND ...

[例] 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;
```

[例] 查询考试成绩有不及格的学生的学号

```
SELECT DISTINCT Sno  
FROM SC  
WHERE Grade < 60;
```



# 确定集合

- SQL中提供了元素与集合之间的比较运算符
- 谓词：x IN <值表>, x NOT IN <值表>

其中 <值表> 是一个集合，从关系代数的角度看，它是一个代数式，从SQL角度看，它是一个SELECT语句

[例] 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' );
```

[例] 查询既不是信息系、数学系，也不是计算机科学系的学生们的姓名和性别。

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



# 字符匹配

- 为什么提供字符匹配运算符？

关系数据库支持对集合的运算，实际应用中，往往需要从集合中找出类似于某个条件的元组，即模糊查询。SQL的字符匹配运算符为解决这类问题而提出

- 谓词：

[NOT] LIKE ' <匹配串>' [ESCAPE ' <换码字符>' ]

- 通配符：

SQL规定符号百分号%及下划线\_\_具有其他含义

百分号%     代表任意长度的字符串

下划线\_\_     代表任意一个字符

<匹配串>    为可以含有通配符的字符串

ESCAPE       是将百分号% 或下划线\_\_转回其本意



# 字符匹配示例

[例] 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] 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__';
```

[例] 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] 查询所有不姓刘的学生姓名。

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



# 字符匹配示例

- **问题：** 因为SQL占用了符号%和\_，若查询中希望使用这两个符号的本意时，该如何解决？
- **解决：** 使用换码字符将通配符转义为普通字符

[例] 查询课程名为DB\_Design的课程号和学分。

```
SELECT Cno, Ccredit  
FROM Course  
WHERE Cname LIKE 'DB\_Design' ESCAPE '\ ';
```

[例] 查询以“DB\_”开头，且倒数第3个字符为i的课程的具体情况。

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

ESCAPE '\ ' 表示 “ \ ” 为换码字符





# 涉及空值的查询

- **动机：** 因为数据库中存在NULL值，而NULL与其它值具有不同含义，所以SQL提供了专门对NULL的运算符
- **谓词：** IS NULL 或 IS NOT NULL  
“IS” 不能用 “=” 代替

[例] 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] 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;
```



# 多重条件查询

- 逻辑运算符：AND和OR可以用来将多个简单查询条件复合成更加复杂的条件，也可用来实现多种其他谓词

**[例] 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] 查询信息系、数学系和计算机系学生的姓名和性别。**

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

改为

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



# 排列元组的显示次序

- **动机**：若希望将查询结果的元组按照排列的次序显示，则可以通过ORDER BY子句。
- **格式**：  
SELECT 块  
ORDER BY 子句；
- **语义**：对SELECT块的结果进行排序，先做SELECT，再ORDER
- **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
- **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子句示例

[例] 查询选修了3号课程的学生学号及其成绩，查询结果按分数降序排列。

```
SELECT Sno, Grade  
FROM SC  
WHERE Cno= ' 3 '  
ORDER BY Grade DESC;
```

[例] 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

- **问题：**一个查询子句的结果是一个集合，有时需要计算出该集合的某个特征值，如个数、最小值等，该如何解决？
- **解决：**SQL支持**聚集函数**来解决求集合特征值问题。
- **聚集函数：**以值的一个集合为输入，返回单个值的函数。
- **基本聚集函数：**
  - 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>)`
- **特殊聚集函数：**
  - `SOME (SELECT 块 )`
  - `ANY (SELECT 块 )`



# 聚集函数示例

[例] 查询学生总人数。

```
SELECT COUNT(*)  
FROM Student;
```

[例] 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT Sno)  
FROM SC;
```

[例] 计算1号课程的学生平均成绩。

```
SELECT AVG(Grade)  
FROM SC  
WHERE Cno= ' 1 ';
```



# 聚集函数示例

[例] 查询选修1号课程的学生最高分数。

```
SELECT MAX(Grade)
FROM SC
WHERE Cno= '1' ;
```

[例] 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;
```



# GROUP BY子句

- **动机**: 对应于分组运算G, SQL提供了GROUP BY子句, 可以将一个查询结果集合进行分组。
- **格式**: GROUP BY  $A_1, A_2, \dots, A_n$   
其中:  $A_i$ 为属性名
- **语义**: 按指定的一列或多列, 对一个SELECT块按值分组, 值相等的为一组
- **GROUP BY子句与聚集函数配合使用, 可以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





# GROUP BY子句示例

[例] 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



SC(S# , C# , SCORE)

列出每个学生的平均成绩

```
SELECT s#, AVG(score)
FROM SC
GROUP BY s#
```

查询 — HUST-26M1ZYR73T\WEB

```
SELECT s#, AVG(score)
FROM SC
GROUP BY s#
```

	s#	(无列名)
1	s1	90
2	s2	85
3	s3	92

90

85

92

S#	C#	score
s1	c1	84
s1	c2	90
s1	c3	96
s2	c1	80
s2	c2	90
s3	c2	96
s3	c3	88

SC(S# , C# , SCORE)

—列出每门课程的平均成绩

```
SELECT c#, AVG(score)
FROM SC
GROUP BY c#
```

查询 — HUST-26M1ZYR73T\WEBKITT			
<pre>SELECT c#, AVG(score) FROM SC GROUP BY c#</pre>			
	c#	(无列名)	
1	c1	82	
2	c2	92	
3	c3	92	

S#	C#	score
s1	c1	84
s1	c2	90
s1	c3	96
s2	c1	80
s2	c2	90
s3	c2	96
s3	c3	88

82

92

92

# Having 子句

**问题：** 有时，对于一个分组以后的**结果集合**希望使用限定条件选择部分分组，则可以使用Having 子句。

**格式：** Having P; P是谓词

**注意：** 由于Having 子句中的谓词P是在分组以后起作用的，因此P中可以使用聚集函数

**[例]** 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子句的区别：**
  - 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。



## 查询语句练习

**cia**表包含**250**多条记录，每个记录代表一个国家。表由**5**个字段组成，字段的值有的是字符串类型，有的是数字类型。

<b>name</b>	<b>region</b>	<b>area</b>	<b>population</b>	<b>gdp</b>
-----				
<b>Yemen</b>	<b>Middle East</b>	<b>527970</b>	<b>14728474</b>	<b>234000000000</b>
<b>Zaire</b>	<b>Africa</b>	<b>2345410</b>	<b>44060636</b>	<b>188000000000</b>
<b>Zambia</b>	<b>Africa</b>	<b>752610</b>	<b>9445723</b>	<b>79000000000</b>
<b>Zimbabwe</b>	<b>Africa</b>	<b>390580</b>	<b>11139961</b>	<b>174000000000</b>

1. 中国的GDP是多少？
2. 给出每个地区的国家数和人口总数。
3. 显示每个地区的总人口数和总面积.仅显示那些面积超过1000000的地区。



# 查询语句练习

1. 中国的GDP是多少？

查询用的SQL语句为：

```
select gdp from cia where name='china'
```

2. 给出每个地区的国家数和人口总数。

查询用的SQL语句为：

```
SELECT region, COUNT(name), SUM(population)  
FROM cia  
GROUP BY region
```



## 查询语句练习

3 显示每个地区的总人口数和总面积。仅显示那些面积超过10000000的地区。

```
SELECT region, SUM(population), SUM(area)
FROM cia
GROUP BY region
HAVING SUM(area)>10000000
```

在这里，我们不能用where来筛选超过10000000的地区，因为表中不存在这样一条记录。



# 连接查询

- 问题**：对应于关系代数中的笛卡尔乘、条件连接、自然连接等运算符，若查询同时涉及多个表，则对应的SQL语句是什么样的？
- 解决**：SQL提供了FROM子句，可以将多个表笛卡尔连接，并配合WHERE子句可以将多张表条件连接
- 语法**：FROM  $r_1, r_2, \dots, r_n$
- 语义**：FROM子句定义了关系表 $r_1, r_2, \dots, r_n$ 的笛卡尔乘积
- 注意**：自然连接、条件连接是由笛卡尔积与选择运算复合完成，所以在SQL中，是由FROM配合WHERE完成






# FROM子句示例

[例] 查询选过课的学生学姓名、课程号及成绩  
将Student表与SC表合并，再选择出选过课的学生，最后投影出所需信息

- 1) 合并 FROM Student, SC
- 2) 选择 WHERE Student.Sno = SC.Sno
- 3) 投影 SELECT Student.name, SC.cno, SC.grade

```
SELECT Student.name, SC.cno, SC.grade  
FROM Student, SC  
WHERE Student.Sno = SC.Sno;
```



➤图为学生-课程数据库中的student关系、Course关系、SC关系

Course:

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

SC:

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

Student:

Sno	Sname	Ssex	Sage	Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS



# 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

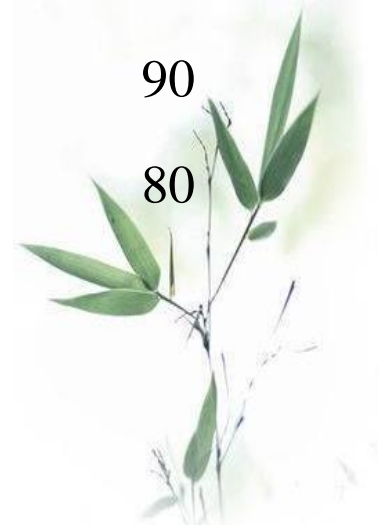
```
SELECT Student.*, SC.*  
FROM Student, SC  
WHERE Student.Sno = SC.Sno;
```



## 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] 对[例33]用自然连接完成。

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



## 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

[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;
```



例：查询每门课的间接先修课

FIRST(COURSE)

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	6	4

SECOND(COURSE)

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	6	4





## 自身连接（续）

查询结果：

Cno	Pcno
1	7
3	5
5	6



## ➤ 自身连接

求同时选修了1号和2号课程的学生学号

```
SELECT A.sno  
FROM SC A, SC B  
WHERE A.sno=B.sno and A.cno='1' and B.cno='2'
```



■查至少选修1号和2号课程的学生学号

$\Pi_{\text{Sno}}(\sigma_{1=4 \wedge 2='1' \wedge 5='2'}(\text{SC} \times \text{SC}))$

A:

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	80

B:

Sno	Cno	Grade
95001	1	92
95001	2	85
95001	3	88
95002	2	90
95002	3	80

Sno1	Cno1	Grade1	Sno2	Cno2	Grade2
95001	1	92	95001	1	92
95001	1	92	95001	2	85
95001	1	92	95001	3	88
95001	1	92	95002	2	90
95001	1	92	95002	3	80



# 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
- 左外连接
  - 列出左边关系中所有的元组
- 右外连接
  - 列出右边关系中所有的元组



# 外连接 (outer join)

A	B	C
a	b	c
b	b	f
c	a	d

R

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null

$R \bar{\bowtie} S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
null	e	f	g

$R \bar{\bowtie} S$

$R \bowtie S$



# 外连接（续）

[例 36] 改写[例33]查询每个学生及其选修课程的情况包括没有选修课程的学生

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

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

## 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;
```

