

# 数据库原理

## The Theory of Database System

### 第三章 关系数据库标准语言SQL



中国矿业大学计算机学院



中国矿业大学数据库原理精品课程

# 本讲主要内容

## 数据查询——多表查询



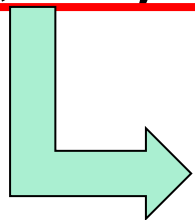
# 引例

- 同时涉及多个表的查询称为连接查询

【例】 查询每个学生及其选修课程的情况。

```
SELECT 学生.*, 学习.*
```

```
FROM 学生, 学习;
```



笛卡尔积



# 等值连接

## ■ 连接的方式

### ■ 通过INNER JOIN ... ON

SELECT 学生.\*, 学习.\*

FROM 学生 INNER JOIN 学习 ON 学生.学号=学习.学号;

### ■ 通过WHERE子句

SELECT 学生.\*, 学习.\*

FROM 学生, 学习

WHERE 学生.学号=学习.学号;



# 自然连接

## 自然连接

- 一种特殊的等值连接
- 将重复的属性列只保留一份

```
SELECT 学生.学号, 姓名, 性别, 出生年份, 籍贯,  
学院, 课程号, 成绩  
FROM 学生 INNER JOIN 学习 ON 学生.学号=学  
习.学号;
```



# 连接条件

- 用来连接两个表的条件称为连接条件
- 一般格式：

[<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>

比较运算符：=、>、<、>=、<=、!=



# 连接操作的执行过程

表1

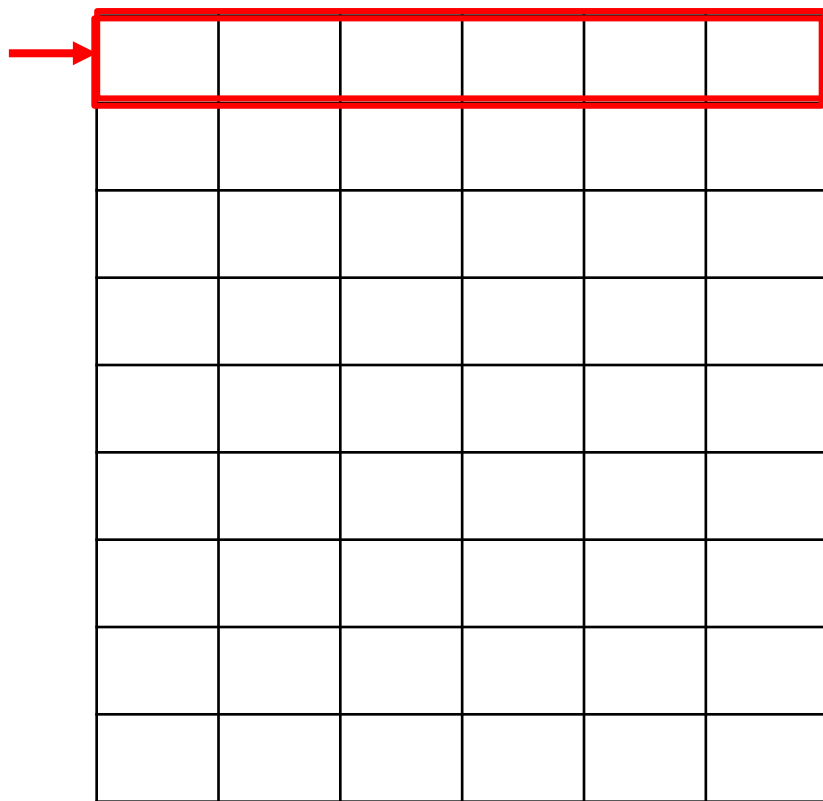
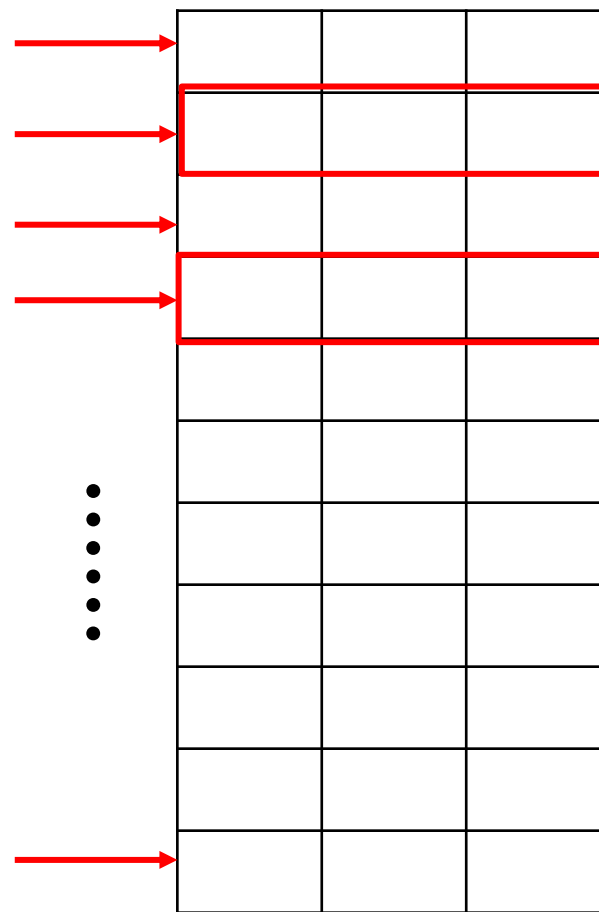



表2




送入结果集

送入结果集



# 复杂的连接查询

【例】查询选修180101号课程且成绩在90分以上的学生学号，姓名及成绩。

形式一：

```
SELECT 学生.学号, 姓名, 成绩  
FROM 学生, 学习  
WHERE 学生.学号=学习.学号  
AND 学习.课程号='180101'  
AND 学习.成绩>90;
```





# 复杂的连接查询(续)

【例】查询选修180101号课程且成绩在90分以上的学生学号，姓名及成绩。

形式二（推荐）：

```
SELECT 学生.学号, 姓名, 成绩  
FROM 学生 INNER JOIN 学习  
ON 学生.学号=学习.学号  
WHERE 学习.课程号='180101'  
AND 学习.成绩>90;
```



# 复杂的连接查询 (续)

【例】查询选修了数据库原理课程的学生姓名和成绩。

形式一：

```
SELECT 学生.学号, 姓名, 成绩
FROM   学生, 学习, 课程
WHERE  学生.学号=学习.学号
AND    学习.课程号=课程.课程号
AND    课程名='数据库原理';
```



# 复杂的连接查询（续）

【例】查询选修了数据库原理课程的学生姓名和成绩。

形式二（推荐）：

```
SELECT 学生.学号, 姓名, 成绩
```

```
FROM 学生
```

```
INNER JOIN 学习 ON 学生.学号=学习.学号
```

```
INNER JOIN 课程 ON 学习.课程号=课程.课程号
```

```
WHERE 课程名='数据库原理';
```



# 带有分组的连接查询

【例】统计每门课的平均分。

```
SELECT 课程名, avg(成绩) AS 平均分  
FROM 课程 JOIN 学习  
ON 学习.课程号=课程.课程号  
GROUP BY 课程.课程号, 课程名 ;
```



# 带有分组的连接查询(续)

某些DBMS规定：

带有分组的查询语句中，**SELECT**子句后面只能出现两类信息：

- (1) 用于分组的属性列；
- (2) 集函数；



## 带有分组的连接查询(续)

【例】查询平均分高于90分的学生学号、姓名以及他的平均分，并按平均分降序排列。

```
SELECT 学生.学号, 姓名, avg(成绩) AS 平均分  
FROM 学生 JOIN 学习  
ON 学生.学号=学习.学号  
GROUP BY 学生.学号, 姓名  
HAVING avg(成绩) >90  
ORDER BY avg(成绩) DESC;
```



【例】查询平均分高于90分的前五名学生的学号、姓名以及他的平均分。

```
SELECT 学生.学号, 姓名, avg(成绩) AS 平均分  
FROM 学生 JOIN 学习  
ON 学生.学号=学习.学号  
GROUP BY 学生.学号, 姓名  
HAVING 平均分>90  
ORDER BY 平均分 DESC  
LIMIT 5;
```



【例】统计选修数据库课程学生的成绩等级，显示学号、姓名、授课教师姓名、成绩、成绩等级。其中，成绩等级的划分原则是：

$$\begin{cases} 100 - 85: \text{优秀} \\ 84 - 75: \text{良好} \\ 74 - 60: \text{中等} \\ < 60: \text{不及格} \end{cases}$$

```
SELECT 学生.学号, 姓名, 成绩,
CASE
    WHEN 成绩 between 85 and 100 THEN '优秀'
    WHEN 成绩 between 75 and 84  THEN '良好'
    WHEN 成绩 between 60 and 74  THEN '中等'
    ELSE '不及格'
END AS 成绩等级
FROM 学生,课程,学习
WHERE 学生.学号=学习.学号 and 课程.课程号=学习.课程号
and 课程名='数据库';
```





# 自连接

- 一个表与其自己进行连接，称为表的自身连接；
- 关系代数：笛卡尔积
- SQL：需要给表起别名以示区别



# 自连接（续）

【例】 查询和刘晨在同一个学院学习的学生的姓名。

分析：

- （1）求得刘晨所在的学院名称；
- （2）在学生表中，用该值筛选符合条件的元组；
- （3）投影出学生姓名。



## 自连接（续）

【例】查询和刘晨在同一个学院学习的学生  
的姓名。

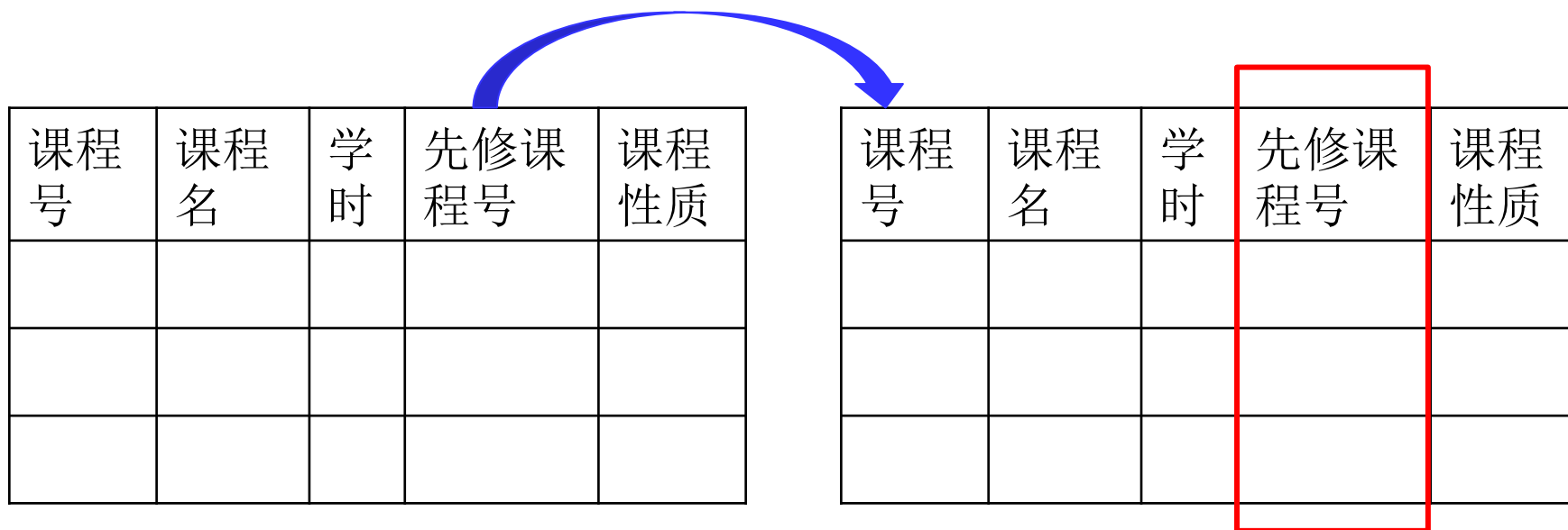
```
SELECT s2.姓名  
FROM 学生 AS s1 JOIN 学生 AS s2  
ON s1.学院=s2.学院  
WHERE s1.姓名='刘晨';
```



# 自连接（续）

【例】查询每一门课的间接先修课（即先修课的先修课）。

分析：



## 自连接（续）

**【例】** 查询每一门课的间接先修课（即先修课的先修课）。

```
SELECT FIRST.课程号, FIRST.课程名, SECOND.先  
修课程号 AS 间接先修课程号  
FROM 课程 AS FIRST JOIN 课程 AS SECOND  
ON FIRST.先修课程号=SECOND.课程号;
```



# 外连接 (Outer Join)

- 外连接是对自然连接的补充;
- 外连接以指定表作为连接主体, 将主体表中的所有元组输出;
- 非主体表有一个“万能”的虚行, 该行由空值组成; 虚行可以和主体表中所有不满足连接条件的元组进行连接。



# 外连接(续)

连接方式:

- 通过OUTER JOIN ... ON
  - 左外连接: LEFT OUTER JOIN ... ON
  - 右外连接: RIGHT OUTER JOIN ... ON
  - 全外连接: FULL OUTER JOIN ... ON



# 左外连接

【例】查询所有学生的姓名以及他们选修课程的课程号和成绩。

SELECT 姓名, 学习.课程号, 成绩

FROM 学生

LEFT JOIN 学习 ON 学生.学号=学习.学号;





# 右外连接

【例】查询所有的课程信息及选修该课程的学生学号及成绩。

```
SELECT 课程名, 学习.学号, 成绩
```

```
FROM 学习
```

```
RIGHT JOIN 课程 ON 学习.课程号=课程.课程号;
```



# 全外连接

- 很多系统不支持全外连接
- 可以通过左外连接和右外连接来间接实现



# 嵌套查询

- 嵌套查询概述
- 嵌套查询分类
- 嵌套查询求解方法
- 引导子查询的谓词



# 嵌套查询的概念

- 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



# 嵌套查询的概念(续)

【例】查询选修了180101号课程的学生姓名。

方法一：连接查询

```
SELECT 姓名  
FROM 学生 JOIN 学习  
ON 学生.学号=学习.学号  
WHERE 学习.课程号='180101' ;
```



# 嵌套查询的概念(续)

【例】 查询选修了180101号课程的学生姓名。

## 方法二：嵌套查询

```
SELECT 姓名  
FROM 学生  
WHERE 学号 IN  
      ( SELECT 学号  
        FROM 学习  
        WHERE 课程号='180101' );
```



# 嵌套查询的概念(续)

- 上层的查询块又称为“外层查询”或“父查询”。
- 下层查询块又称为“内层查询”或“子查询”。
- 以层层嵌套的方式来构造程序正是 SQL(Structured Query Language)中“结构化”的含义所在。



# 嵌套查询分类

- 不相关子查询

- 每个子查询可以在父查询之前先独立的完成，子查询的结果用于建立其父查询的查找条件。

- 相关子查询

- 最内层子查询的执行需要用到上层父查询中的某些属性值，因此最内层的子查询不能独立于父查询先完成。





# 嵌套查询求解方法

- 总原则：由里向外逐层处理。
- 不相关子查询  
先从最内层开始，每层执行完，然后上一层开始执行，直到最外层执行完。
- 相关子查询  
上层每取一条记录，子查询执行一遍，重复这个过程，直到上层表里记录全部访问完。



# 引导子查询的谓词

- 带有IN谓词的子查询
- 带有比较运算符的子查询
- 带有ANY或ALL谓词的子查询
- 带有EXISTS谓词的子查询



# 带有IN谓词的子查询

- IN 后面跟的是集合，表示取值在这个集合中。

【例】查询与刘晨在同一个学院学习的学生  
的学号和姓名。

此查询要求可以分步来完成：



# 带有IN谓词的子查询(续)

```
SELECT 学号, 姓名  
FROM 学生  
WHERE 学院 IN  
      ( SELECT 学院  
        FROM 学生  
        WHERE 姓名 = '刘晨' );
```



# 带有IN谓词的子查询(续)

【例】 查询选修了数据库原理课程的学生们的学号和姓名。

```
SELECT 学号, 姓名  
FROM 学生  
WHERE 学号 IN  
      (SELECT 学号  
       FROM 学习  
       WHERE 课程号 IN  
            (SELECT 课程号  
             FROM 课程  
             WHERE 课程名= '数据库原理' ));
```

③ 最后在学生表中找出这些学号对应的学号和姓名

② 然后在学习表中找出选修了这个课程号的学生学号

① 首先在课程表中找出“数据库原理”的课程号



## 带有IN谓词的子查询(续)

【例】查询没有选修数据库原理课程的学生学号和姓名。

分析：

- 表中存储的数据对应实际发生的事情；
- 差运算；
- NOT IN 表示不在集合中；



# 带有IN谓词的子查询(续)

【例】 查询没有选修数据库原理课程的学生们的学号和姓名。

```
SELECT 学号, 姓名  
FROM 学生  
WHERE 学号 NOT IN  
      (SELECT 学号  
       FROM 学习  
       WHERE 课程号 IN  
            (SELECT 课程号  
             FROM 课程  
             WHERE 课程名= '数据库原理' ));
```



# 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<>）。
- 可以与ANY或ALL谓词配合使用





## 带有比较运算符的子查询(续)

【例】查询180106课程中成绩高于课程平均分的学生的学号、姓名和成绩。

```
SELECT 学生.学号, 姓名, 成绩
FROM 学生 JOIN 学习 ON 学生.学号=学习.学号
WHERE 课程号='180106'
      AND 成绩>
      ( SELECT avg(成绩)
        FROM 学习
        WHERE 课程号='180106' );
```



# 带有ANY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或ALL谓词的子查询(续)

运算符	ANY	ALL
>	大于子查询结果中的某个值	大于子查询结果中的所有值
<	小于子查询结果中的某个值	小于子查询结果中的所有值
>=	大于等于子查询结果中的某个值	大于等于子查询结果中的所有值
<=	小于等于子查询结果中的某个值	小于等于子查询结果中的所有值
=	等于子查询结果中的某个值	通常没有实际意义
!=	不等于子查询结果中的某个值	不等于子查询结果中的任何一个值



## 带有ANY或ALL谓词的子查询(续)

【例】 查询其他学院中比计算机学院某个学生年龄小的学生名单。

SELECT 姓名

FROM 学生

WHERE year(now())-出生年份 < ANY

(SELECT year(now())-出生年份

FROM 学生

WHERE 学院='计算机' )

AND 学院 <> '计算机' ;

/\* 注意这是父查询块中的条件 \*/



## 带有ANY或ALL谓词的子查询(续)

【例】查询其他学院中比计算机学院某个学生年龄小的学生名单。

用集函数**MAX**等价实现：

```
SELECT 姓名
FROM 学生
WHERE year(now())-出生年份<
      (SELECT MAX(year(now())-出生年份)
      FROM 学生
      WHERE 学院='计算机' )
AND 学院 <> '计算机' ;
```



## 带有ANY或ALL谓词的子查询(续)

【例】 查询其他学院中比计算机学院所有学生年龄都小的学生名单。

```
SELECT 姓名  
FROM 学生  
WHERE year(now())-出生年份 < ALL  
      (SELECT year(now())-出生年份  
        FROM 学生  
        WHERE 学院='计算机' )  
AND 学院 <> '计算机';
```



## 带有ANY或ALL谓词的子查询(续)

【例】查询其他学院中比计算机学院所有学生年龄都小的学生名单。

用集函数**MIN**等价实现:

```
SELECT 姓名  
FROM 学生  
WHERE year(now())-出生年份<  
      (SELECT MIN(year(now())-出生年份)  
       FROM 学生  
       WHERE 学院='计算机' )  
AND 学院 <> '计算机';
```



# 带有ANY或ALL谓词的子查询(续)

- ANY和ALL谓词有时可以用集函数实现
  - ANY与ALL与集函数的对应关系

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





## 带有ANY或ALL谓词的子查询(续)

【例】 查询课程平均成绩最高的课程号和课程名。

SELECT 课程号, 课程名

FROM 课程, 学习

WHERE 课程.课程号=学习.课程号

GROUP BY 课程号, 课程名

HAVING avg(成绩)=MAX(avg(成绩));



## 带有ANY或ALL谓词的子查询(续)

【例】 查询课程平均成绩最高的课程号和课程名。

```
SELECT 课程号, 课程名  
FROM 课程, 学习  
WHERE 课程.课程号=学习.课程号  
GROUP BY 课程号, 课程名  
HAVING avg(成绩)>=ALL(SELECT avg(成绩)  
                        FROM 学习  
                        GROUP BY 课程号);
```

