

# 数据库系统



**SiChuan University**

# 数据库系统

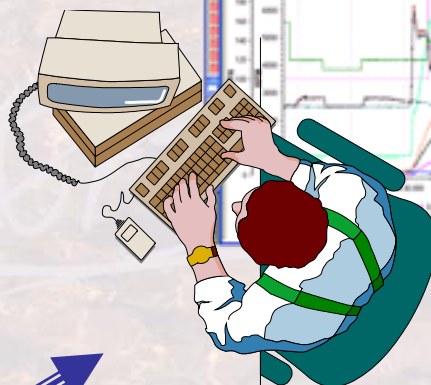
- 一. 引言
- 二. 数据库和数据系统
- 三. 关系模型概念
- 四. 关系运算之并、差、交、积
- 五. 关系运算之选择、投影、连接
- 六. 应用关系运算进行数据库的查询
- 七. **SQL**数据库的操作

# 数据为什么要管理--数据自有黄金屋

## (1)信息社会的工作方式？



传统社会：业务工作



信息社会：业务工作 + 计算机支持

- 网络/Internet
- 数据库

**Everything Over DB**

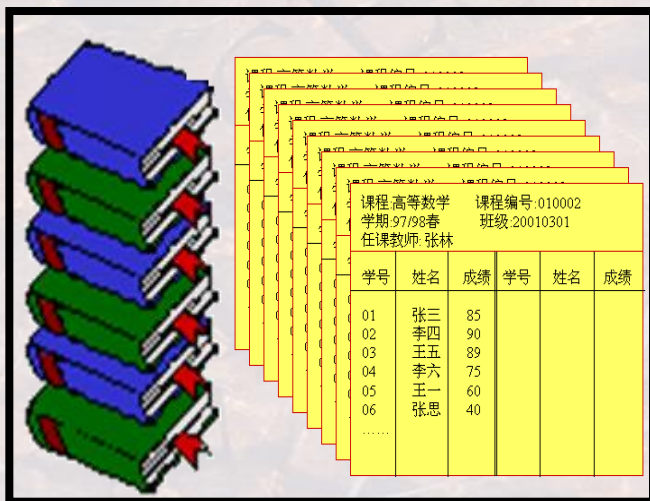




# 数据为什么要管理--数据自有黄金屋

## (2)数据为什么要管理？

### 数据与数据库



## 数据

形成“库”，实现“积累”  
应用“库”，实现积累的效益  
“库”的管理与控制

- 纸面数据 vs. 电子数据
- 单一数据文件 vs. 数据库
- 数据产生的分散化 vs. 数据应用的共享化
- 小规模数据 vs. 大规模数据

### 各种“资源”库

- 图像数据库、音乐数据库与多媒体数据库
- 工程数据库
- 地理信息数据库
- 文献数据库
- Web数据库。又称为Internet数据库
- 数据仓库
- 车辆数据库
- 产品数据库
- 机床数据库
- 信用数据库
- 烟酒数据库
- ... ..

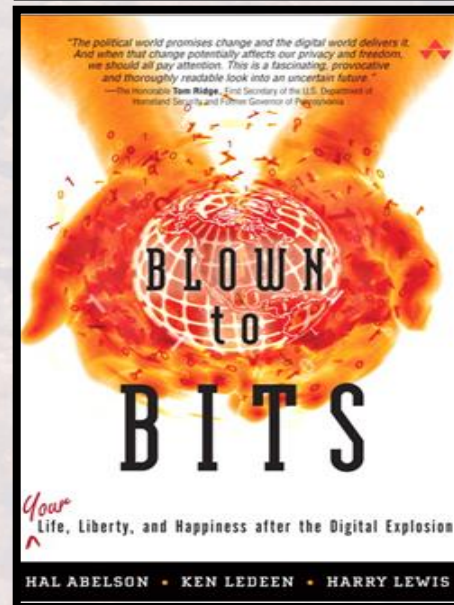
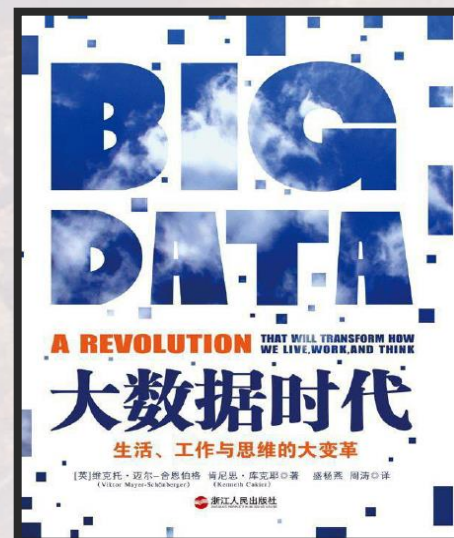
## (4)数据自有黄金屋？

2008年全球产生的数据量为0.49ZB( $2^{50}$ MB)  
2009年的数据量为0.8ZB  
2010年增长为1.2ZB  
2011年的数量更是高达1.82ZB  
2020年为止，人类所有印刷材料的数据量是44ZB

一个例子。大家乘坐飞机时都希望买到更便宜的机票，可能都相信“购买机票，越早预订越便宜”，果真然否？2003 年 Farecast 公司创始人奥伦·埃齐奥尼(Oren Etzioni)提前几个月在网上订了一张机票，在飞机上与邻座若干乘客交谈时，他发现尽管很多人机票比他买的更晚，但票价却比他的便宜得多。出了什么问题？是航空公司或者网站有意“欺诈”，还是常识“购买机票，越早越好”错了？

受此启发，埃齐奥尼开始思考：机票价格是否合理呢？又该如何判断机票何时该降价、什么原因降价，只有航空公司才掌握机票价格变化的规律。埃齐奥尼要做的是仅仅依靠数据“猜”出机票价格变化的规律。经过一段时间内会上涨还是下降的明智选择。反过来说，如果机票价格预测错了，他开发了一个软件，能预测未来 41 天内价格波动产生的 12000 条数据。它建立了在一个旅游网站上收集来的 41 天内价格波动产生的 12000 条数据与提前购买天数之间的关系。它不知道是哪些因素导致了机票价格的波动，还是所谓的周六晚上不出门，它只知道利用其他航班的价格来预测未来机票价格的走势以及增减幅度，能帮助消费者抓住最佳时机。

为了构建这个数据库，系统必须收集所有行业机票预订数据库。有了这个数据库，系统就能预测出未来 41 天内每条航线上的每一架飞机内的每一个座位一天内的综合票价记录。如今，Farecast 已经拥有惊人的约 2000 亿条飞行数据。到 2012 年为止，Farecast 预测的准确率为 95%。Farecast 票价预测系统平均每张机票可节省 50 美元。这项技术也可以延伸到其他领域，如宾馆预订、二手车购买等。只要这些领域内的产品差异不大，同时存在大幅度的价格差和大量可适用的数据，就都可以应用这项技术。





# 数据为什么要管理--数据自有黄金屋

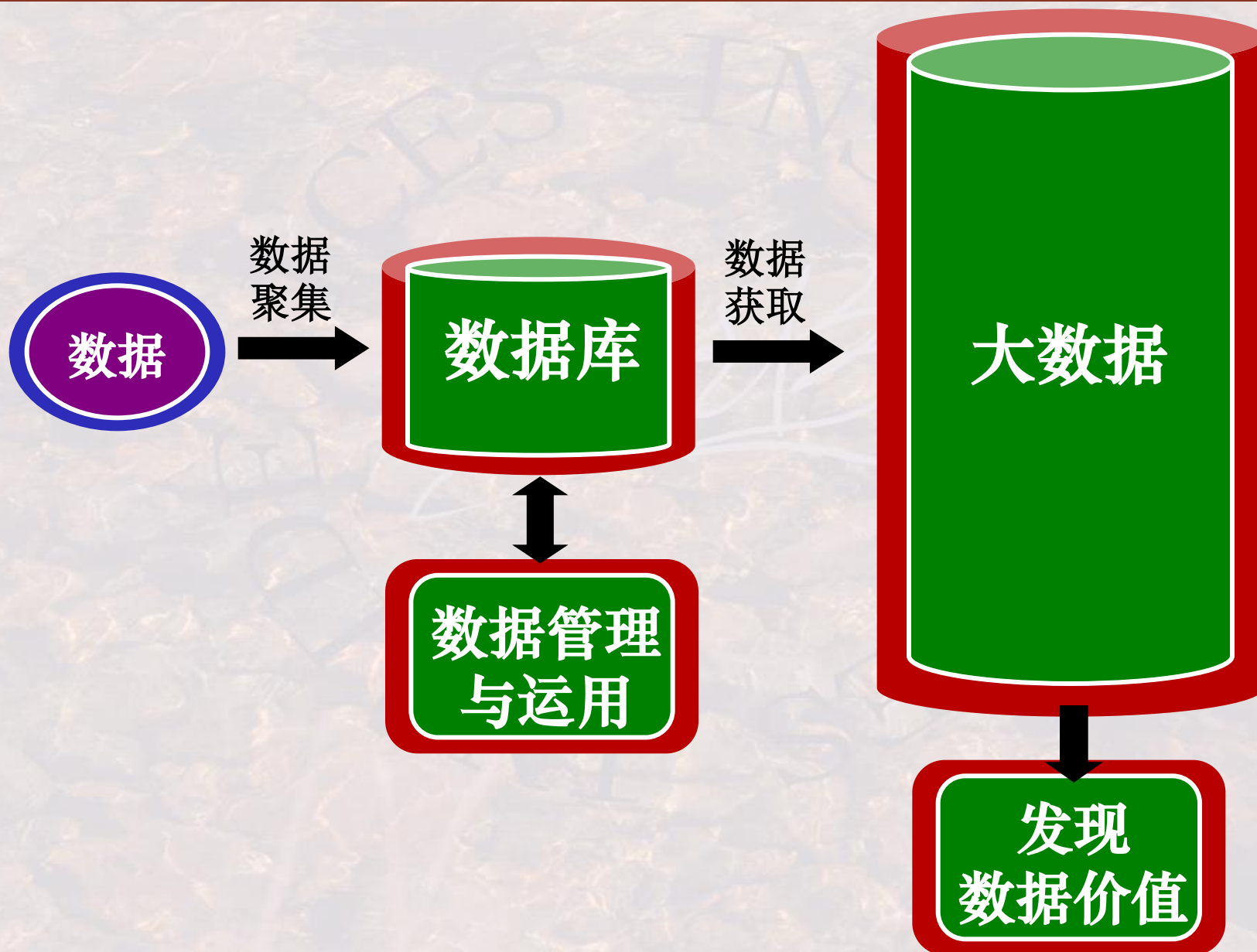
## (4)数据自有黄金屋？

### 大数据价值发现

- 华尔街金融家利用电脑程序分析全球**3.4亿**微博账户的留言，根据民众情绪抛售股票；
- 银行根据求职网站的岗位数量，推断就业率；
- 投资机构搜集并分析上市企业声明，从中寻找破产的蛛丝马迹；
- 美国总统奥巴马的竞选团队依据选民的微博，实时分析选民对总统竞选人的喜好，基于数据对竞选议题的把握，成功赢得总统大选。
- 中国网民发动的“人肉搜索”，已成功地使若干“表哥”“表叔”“房叔”“房妹”等腐败官员落入法网。
- .....

# 数据为什么要管理--数据自有黄金屋

## (6)小结？





# 什么是数据库与数据库系统？

# 什么是数据库与数据库系统

## (1)数据库？

数据库：相互有关联关系的数据的集合

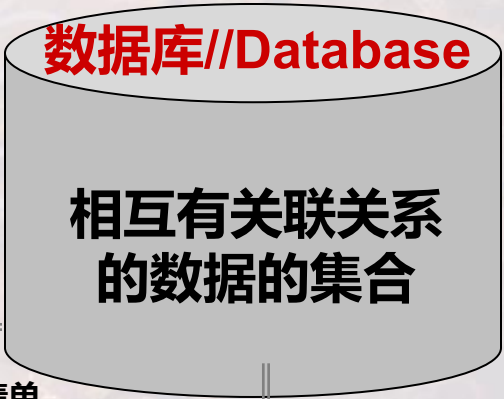
- 一个表聚集了具有相同结构类型的若干个对象
- 一行数据反映了某一对象的相关内容
- 一列数据具有相同的数据类型
- 表与表间也存在着相互关联

学生登记表

学号	姓名	性别	出生年月	入学日期	家庭住址
98110101	张三	男	1980.10	1998.09	黑龙江省哈尔滨市
98110102	张四	女	1980.04	1998.09	吉林省长春市
98110103	张五	男	1981.02	1998.09	黑龙江省齐齐哈尔市
98110201	王三	男	1980.06	1998.09	辽宁省沈阳市
98110202	王四	男	1979.01	1998.09	山东省青岛市
98110203	王武	女	1981.06	1998.09	河南省郑州市

学生成绩单

班级	课程	教师	学期	学号	姓名	成绩
981101	数据库	李四	98秋	98110101	张三	100
981101	数据库	李四	98秋	98110102	张四	90
981101	数据库	李四	98秋	98110103	张五	80
981101	计算机	李五	98秋	98110101	张三	89
981101	计算机	李五	98秋	98110102	张四	98
981101	计算机	李五	98秋	98110103	张五	72
981102	数据库	李四	99秋	98110201	王三	30
981102	数据库	李四	99秋	98110202	王四	90
981102	数据库	李四	99秋	98110203	王武	78

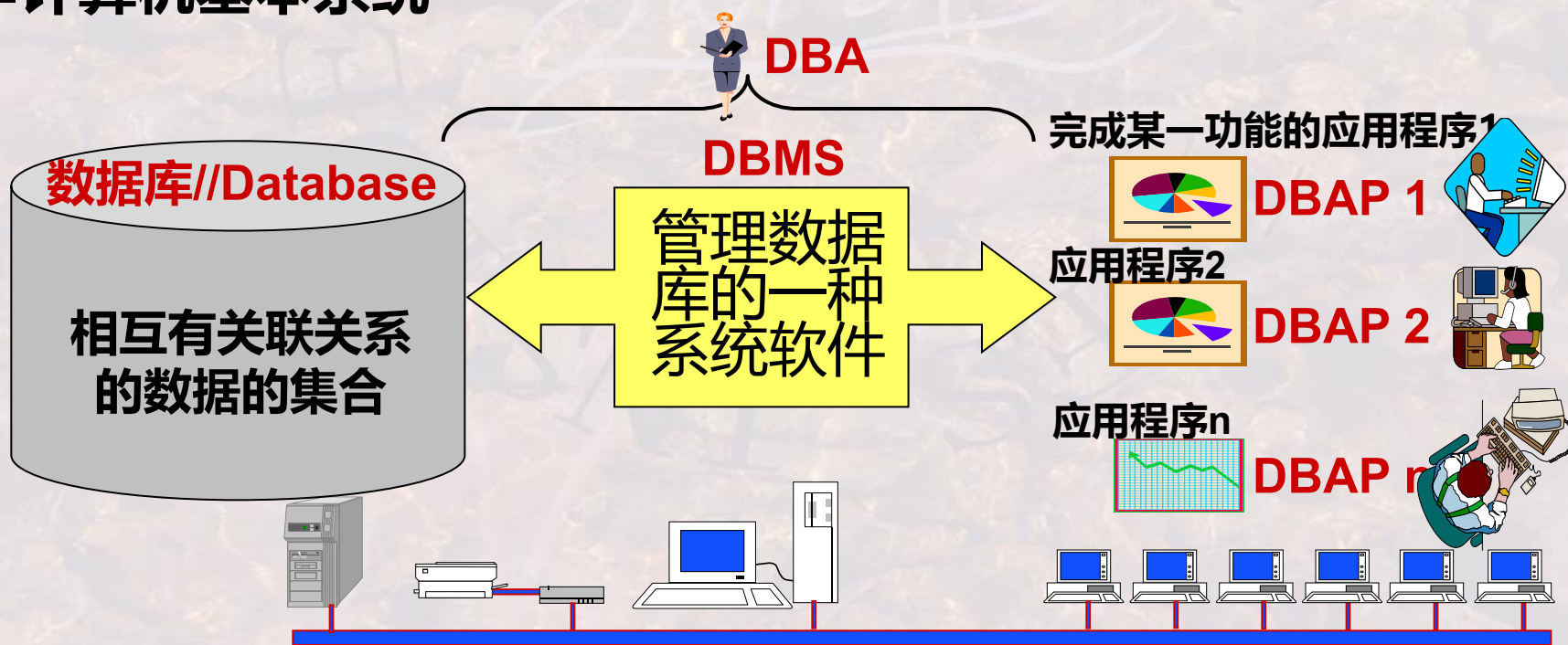


# 什么是数据库与数据库系统

## (2)数据库系统的几个构成部分？

### 数据库系统(工作环境)

- 数据库(DB): Database
- 数据库管理系统(DBMS): Database Management System
- 数据库应用(DBAP): DataBase Application
- 数据库管理员(DBA): DataBase Administrator
- 计算机基本系统

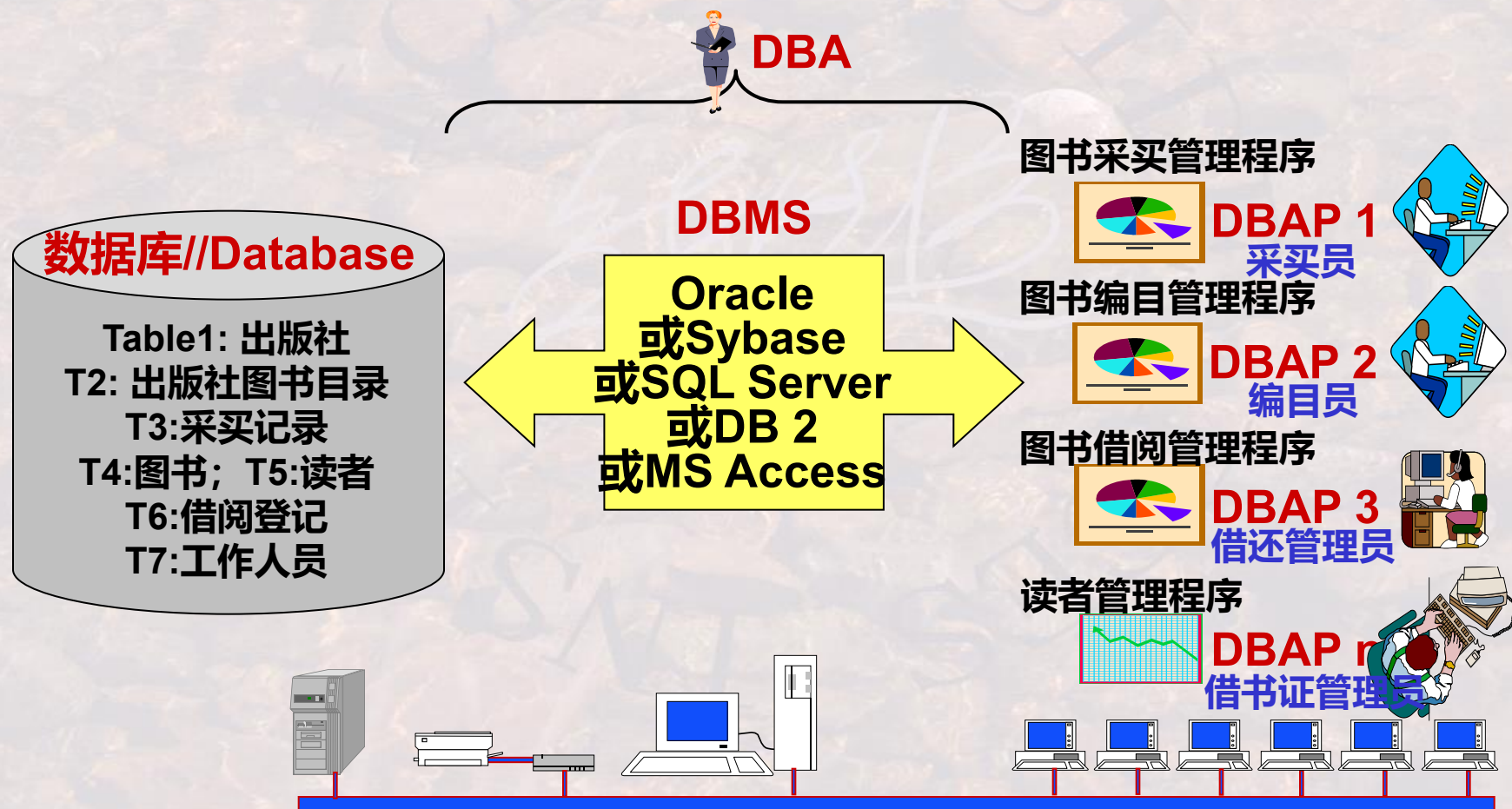




# 什么是数据库与数据库系统

## (2)数据库系统的几个构成部分？

### 数据库系统(工作环境)示例：图书管理数据库系统

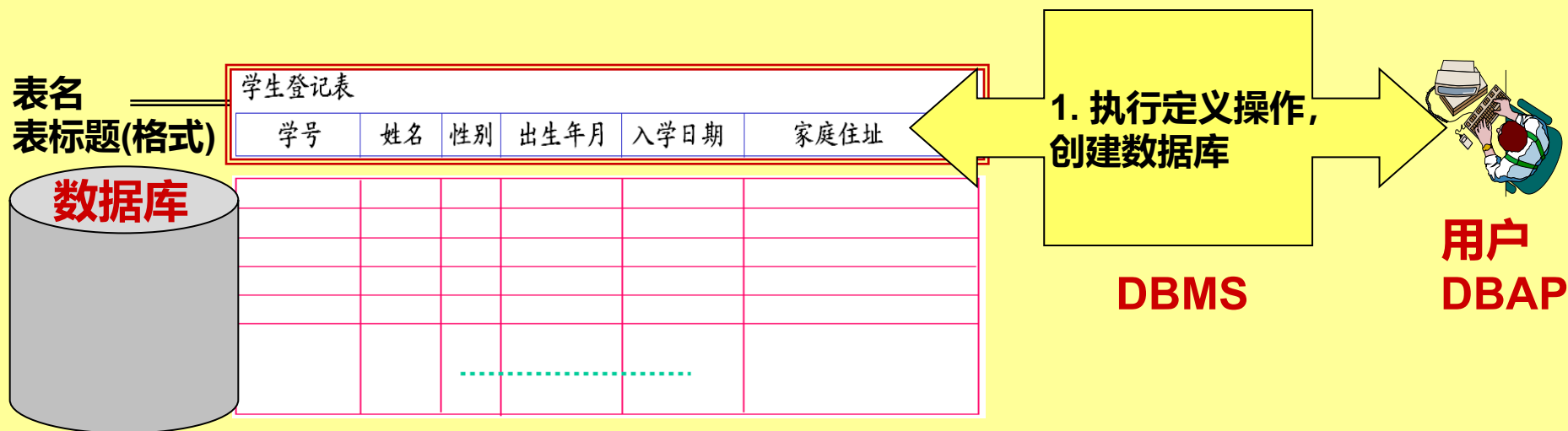


### (3)数据库管理系统的基本功能

❑ DBMS提供一套数据定义语言(DDL:Data Definition Language)给用户

## □ 用户使用DDL描述其所要建立表的格式

## ❑ DBMS依照用户的定义，创建数据库及其中的Table

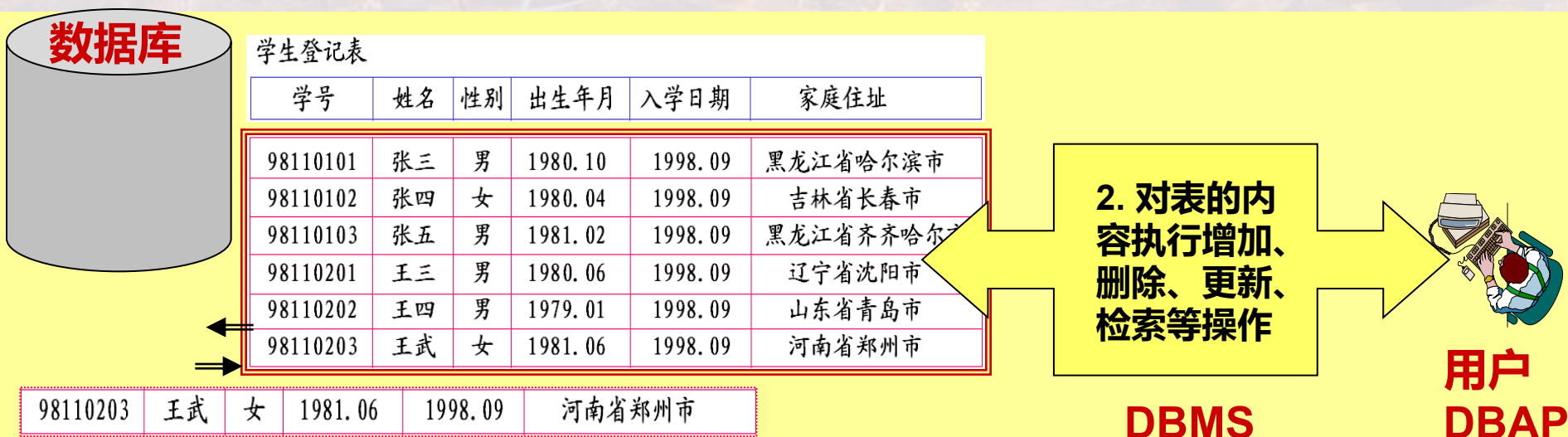


# 什么是数据库与数据库系统

## (3)数据库管理系统的基本功能

### 数据库操纵: 向数据库的Table中增加/删除/更新数据及对数据进行查询、检索、统计等

- DBMS提供一套数据操纵语言(DML:Data Manipulation Language)给用户
- 用户使用DML描述其所要进行的增、删、改、查等操作
- DBMS依照用户的操作描述, 实际执行这些操作



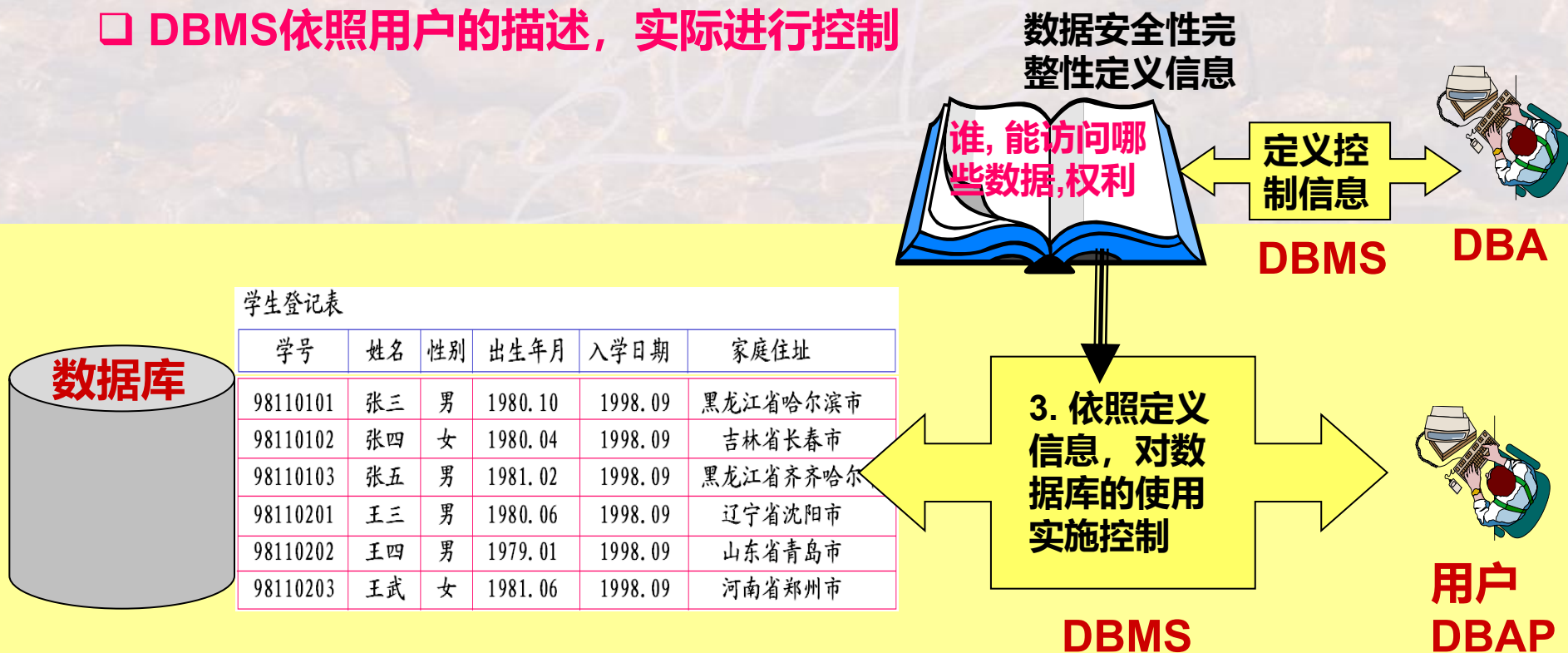


# 什么是数据库与数据库系统

## (3)数据库管理系统的基本功能

### 数据库控制: 控制数据库中数据的使用---哪些用户可以使用,哪些不可以

- DBMS提供一套数据控制语言(DCL:Data Control Language)给用户
- 用户使用DCL描述其对数据库所要实施的控制
- DBMS依照用户的描述, 实际进行控制



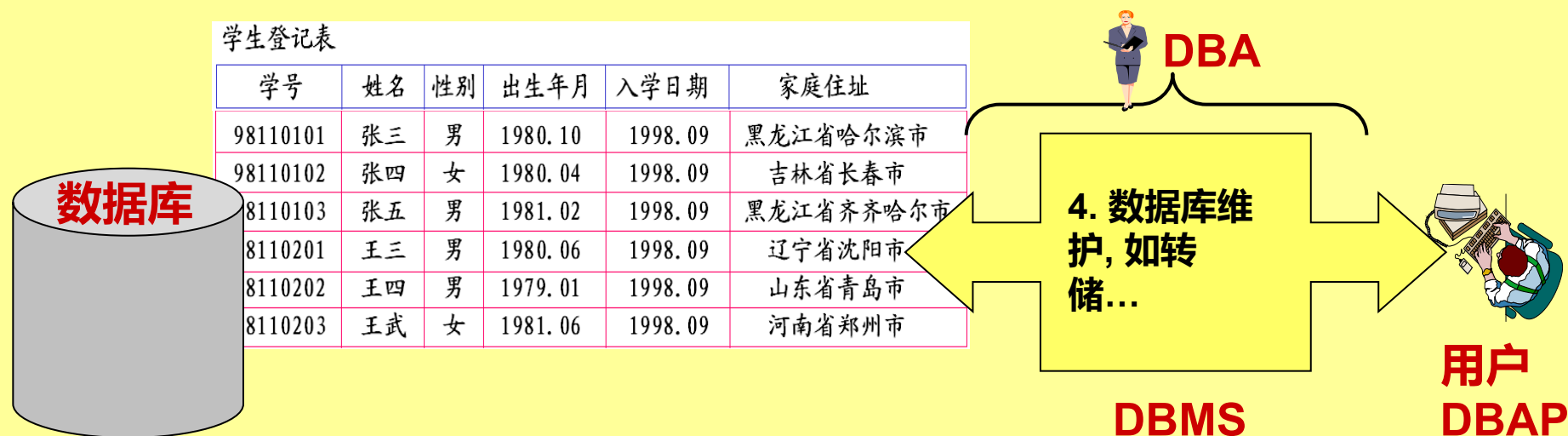
# 什么是数据库与数据库系统

## (3)数据库管理系统的基本功能

### 数据库维护: 转储/恢复/重组/性能监测/分析...

- DBMS提供一系列程序(实用程序/例行程序)给用户
- 在这些程序中提供了对数据库维护的各种功能
- 用户使用这些程序进行各种数据库维护操作

➤数据库维护的实用程序，一般都是由数据库管理员(DBA)来使用和掌握的



# 什么是数据库与数据库系统

## (3)数据库管理系统的基本功能

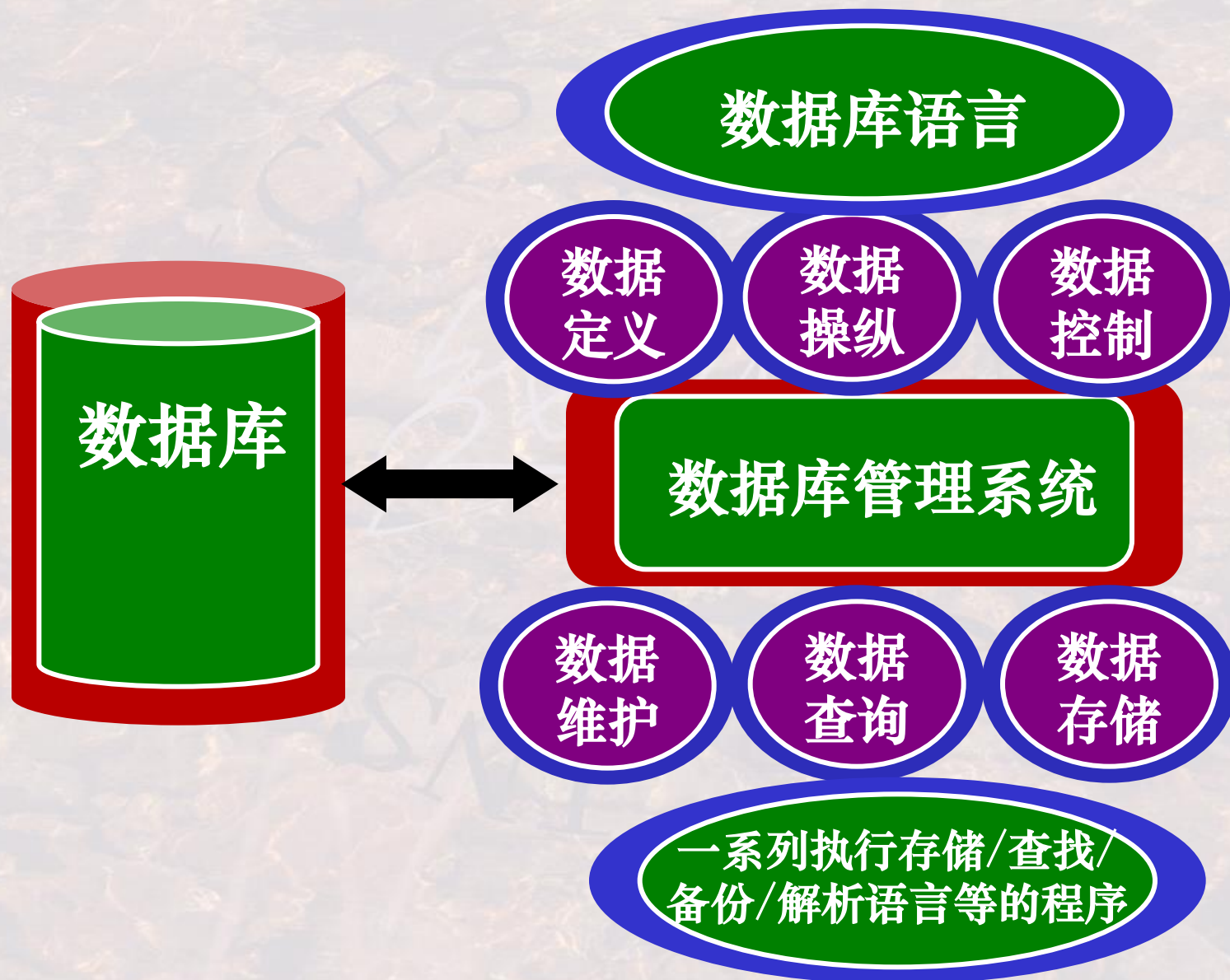
**DBMS为完成DB管理，在后台运行着一系列程序...**

- ☐数据库物理存储
- ☐数据库查询执行及查询优化
- ☐并发控制
- ☐故障恢复
- ☐安全性控制
- ☐完整性控制
- ☐数据字典管理
- ☐应用程序接口(API)
- ☐... ..



# 什么是数据库与数据库系统

## (4)小结



# 基本数据模型：关系模型I

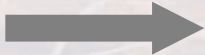
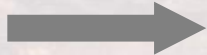
## -什么是关系

# 基本数据模型：关系模型I-什么是关系

## (1)什么是数据模型？

**数据模型** 刻画信息世界或数据世界的一组严格定义的概念的集合

- ① **数据结构** 有哪些格式的数据
- ② **数据操作** 对这些格式的数据都可能有哪些操作
- ③ **完整性约束** 为保证操作后和操作过程中产生的数据仍符合规定所必须遵守的约束条件

现实世界            信息世界            数据世界

概念数据模型(简称概念模型)

数据库三大经典的数据模型

关系模型

层次模型

网状模型



# 基本数据模型：关系模型I-什么是关系

## (2)你理解关于关系的一些术语的含义吗？

数据库的关系模型起源于规范化 “表(Table)”的处理

Table: 以按行按列形式组织及展现的数据

列/字段/属性/数据项(column/field/attribute/data item)

列名

学生成绩单

班级	课程	教师	学期	学号	姓名	成绩
981101	数据库	李四	98秋	98110101	张三	100
981101	数据库	李四	98秋	98110102	张四	90
981101	数据库	李四	98秋	98110103	张五	80
981101	计算机	李五	98秋	98110101	张三	89
981101	计算机	李五	98秋	98110102	张四	98
981101	计算机	李五	98秋	98110103	张五	72
981102	数据库	李四	99秋	98110201	王三	30
981102	数据库	李四	99秋	98110202	王四	90
981102	数据库	李四	99秋	98110203	王武	78

行/  
元组/  
记录  
( row /  
tuple /  
record )

列值

Table中描述了一批相互有关联关系的数据==>关系

# 基本数据模型：关系模型I-什么是关系

## (3)如何用数学来定义关系呢？

### 用数学严格地定义Table

怎样把一张表格定义清楚呢？

家庭		
丈夫	妻子	子女
李基	王方	李键
张鹏	刘玉	张睿
张鹏	刘玉	张峰

#### 2. 值域(Domain)

说清楚每一列数据可能的取值

#### 1. 指出有多少列

#### 4.指出关系中的元组

关系中元组是有意义的组合  
----笛卡尔积的子集

#### 3.指出所有可能的元组

元组是值的一个组合；值域中值的所有可能的组合----笛卡尔积

# 基本数据模型：关系模型I-什么是关系

## (3)如何用数学来定义关系呢？

### 用数学严格地定义Table

➤首先定义“列”的取值范围“域(Domain)”

➤域(Domain)

□一组值的集合，这组值具有相同的数据类型

□如整数的集合、字符串的集合、全体学生的集合

□再如, 由8位数字组成的数字串的集合，由0到100组成的整数集合

□集合中元素的个数称为域的基数(Cardinality)

家庭		
丈夫	妻子	子女
李基	王方	李健
张鹏	刘玉	张睿
张鹏	刘玉	张峰

$D_3 = \text{儿童集合(CHILD)} = \{\text{李健, 张睿, 张峰}\}$

$D_2 = \text{女人集合(WOMAN)} = \{\text{王芳, 刘玉}\}$

$D_1 = \text{男人集合(MAN)} = \{\text{李基, 张鹏}\}$



# 基本数据模型：关系模型I-什么是关系

## (3)如何用数学来定义关系呢？

### 用数学严格地定义Table

➤再定义“元组”及所有可能组合成的元组：笛卡尔积

➤笛卡尔积(Cartesian Product)

□一组域 $D_1, D_2, \dots, D_n$ 的笛卡尔积为：

$$D_1 \times D_2 \times \dots \times D_n = \{ (d_1, d_2, \dots, d_n) \mid d_i \in D_i, i=1, \dots, n \}$$

□笛卡尔积的每个元素 $(d_1, d_2, \dots, d_n)$ 称作一个n-元组 (n-tuple)



# 基本数据模型：关系模型I-什么是关系

## (3)如何用数学来定义关系呢？

### 用数学严格地定义Table

➤由于笛卡尔积中的所有元组并不都是有意义的，因此...

➤关系(Relation)

□一组域 $D_1, D_2, \dots, D_n$ 的笛卡尔积的子集:

□笛卡尔积中具有某一方面意义的那些元组被称作一个关系(Relation)

□由于关系的不同列可能来自同一个域，为区分，需要为每一列起一个名字，该名字即为属性名。不同列名的列值可以来自相同域。

例如：家庭(丈夫:男人, 妻子:女人, 子女:儿童)或家庭(丈夫, 妻子, 子女)

笛卡尔积

男人	女人	儿童
李基	王方	李键
李基	王方	张睿
李基	王方	张峰
李基	刘玉	李键
李基	刘玉	张睿
李基	刘玉	张峰
张鹏	王方	李键
张鹏	王方	张睿
张鹏	王方	张峰
张鹏	刘玉	李键
张鹏	刘玉	张睿
张鹏	刘玉	张峰



家庭

丈夫	妻子	子女
李基	王方	李键
张鹏	刘玉	张睿
张鹏	刘玉	张峰

列名(属性名)

列值：来自域

# 基本数据模型：关系模型I-什么是关系

## (4)关系有什么性质？

### 关系的性质

列是同质的(Homogeneous)，即每一列中的分量是同一类型数据，来自同一个域

不同的列可出自同一个域，每一列称为属性，要给予不同的属性名

列的顺序可以任意交换，行的顺序也可以任意交换

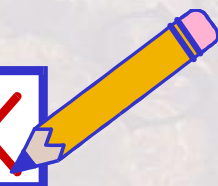
任意两个元组不能完全相同

每一分量必须是不可分的数据项

学生成绩单

班级	课程	学号	姓名	成绩
981101	数据库	01	张三	100
981101	数据库	02	张四	90
981101	数据库	03	张五	80
981101	计算机	01	张三	89
981101	计算机	02	张四	98
981101	计算机	03	张五	72
981102	数据库	01	王三	30
981102	数据库	02	王四	90
981102	数据库	03	王武	78

丈夫	妻子	孩子	
		第一个	第二个
李基	王芳	李健	张峰
张鹏	刘玉	张睿	





#### 候选码(Candidate Key)/候选键

□关系中的一个属性组，其值能唯一标识一个元组，若从该属性组中去掉任何一个属性，它就不具有这一性质了，这样的属性组称作候选码。

学生(S#, Sname, Sage, Sclass)

课程(C#, Cname, Credit, T#)

# 基本数据模型：关系模型I-什么是关系

## (6)关系中的外键？

### 外码(Foreign Key)/外键

□关系R中的一个属性组，它不是R的候选码，但它与另一个关系S的候选码相对应，则称这个属性组为R的外码或外键。

□外码是两个关系(数据表)的连接纽带

合同					
主码	合同号	合同名称	合同签定人	客户号	外码
	HT0001	购煤合同	张三	CUST01	
	HT0002	销售机床合同	李四	CUST01	
	HT0003	购钢材合同	张五	CUST02	

两个关系可以靠外码联接起来

主码

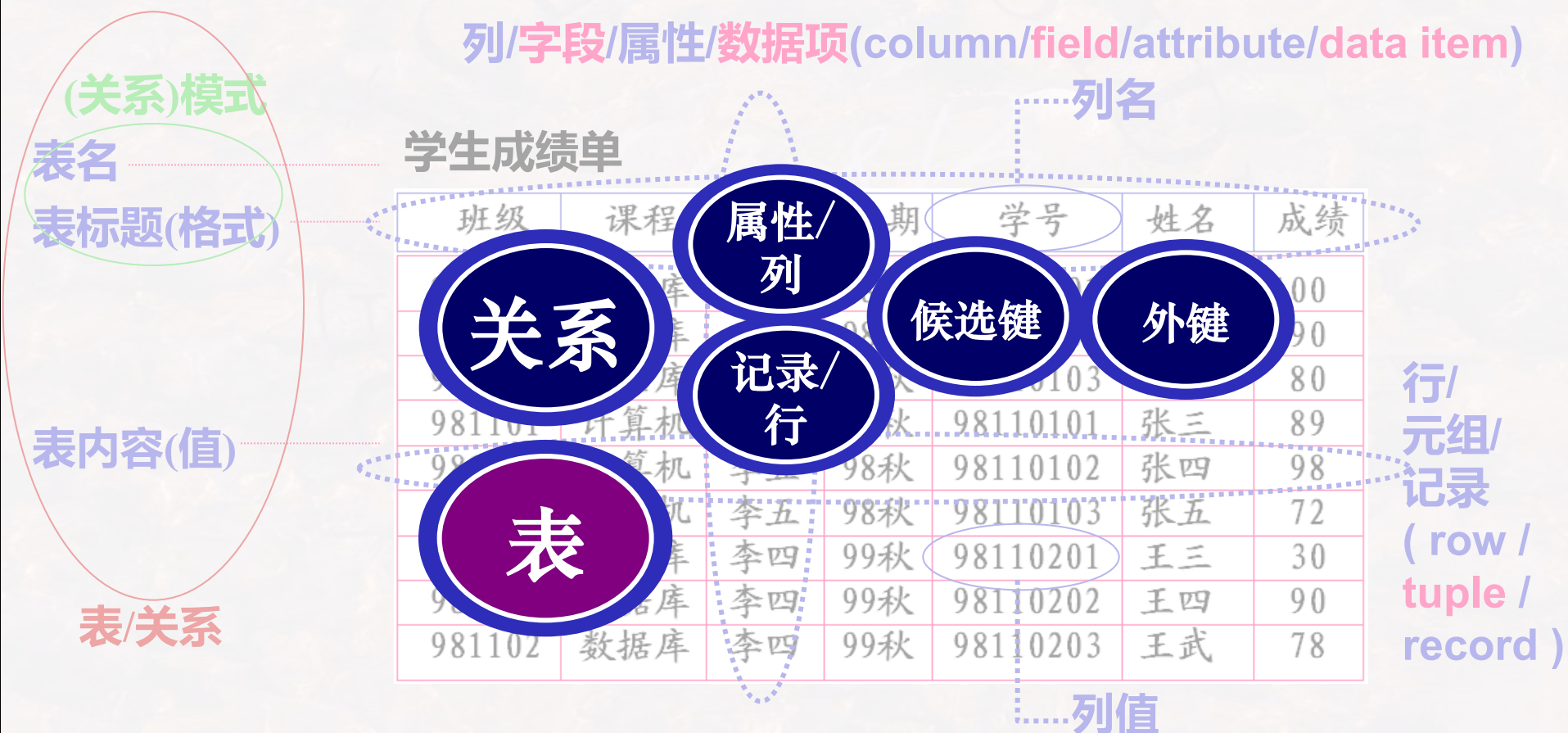
客户				
客户号	客户名称	客户地址	联系人	
CUST01	依兰煤矿	哈尔滨市	王三	
CUST02	长春电机厂	长春市	赵六	
CUST03	鞍钢集团	鞍山市	钱七	

# 基本数据模型：关系模型I-什么是关系

## (7)小结？

数据库的关系模型起源于规范化 “表(Table)” 的处理

Table: 以按行按列形式组织及展现的数据



Table中描述了一批相互有关联关系的数据==>关系



# 基本数据模型：关系模型II

## -关系运算之并、差、交、积

# 基本数据模型：关系模型II-关系运算之并-差-交-积

## (1)什么是关系运算？

## 什么是关系运算？

学生登记表					
学号	姓名	性别	出生年月	入学日期	家庭住址
98110101	张三	男	1980.10	1998.09	黑龙江省哈尔滨市

学生成绩单							
	班级	课程	教师	学期	学号	姓名	成绩
98110102	981101	数据库	李四	98秋	98110101	张三	100
98110103	981101	数据库	李四	98秋	98110102	张四	90
98110201	981101	数据库	李四	98秋	98110103	张五	80
98110202	981101	计算机	李五	98秋	98110101	张三	89
98110203	981101	计算机	李五	98秋	98110102	张四	98
	981101	计算机	李五	98秋	98110103	张五	72
	981102	数据库	李四	99秋	98110201	王三	30
	981102	数据库	李四	99秋	98110202	王四	90
	981102	数据库	李四	99秋	98110203	王武	78

有哪些运算？

并：  $R \cup S$

差：  $R - S$

积：  $R \times S$

选择：  $\sigma(R)$

投影：  $\pi(R)$

连接：  $R \bowtie S$

交：  $R \cap S$

## (2)什么情况用并运算呢？

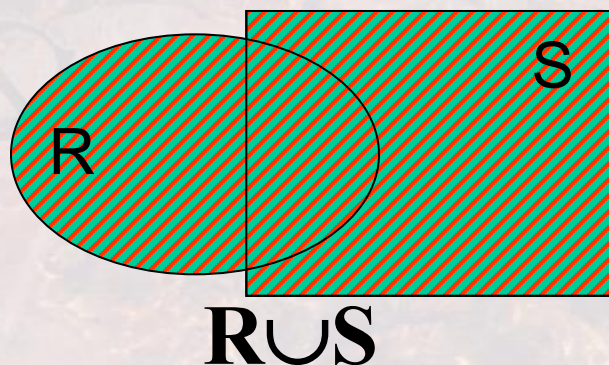
### 并(Union)

□ **定义**：设关系R和关系S是并相容的(即：属性数目相同，其对应的域也相同)，则关系R与关系S的并运算结果也是一个关系，记作：

$R \cup S$ ，它由**或者出现在关系R中，或者出现在S中的元组**构成

□ **数学描述**： $R \cup S = \{ t \mid t \in R \vee t \in S \}$ ，其中t是元组

□  $R \cup S$  与  $S \cup R$  运算的结果是同一个关系





# 基本数据模型：关系模型II-关系运算之并-差-交-积

## (2)什么情况用并运算呢？

### 并(Union)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

RUS		
C1	C2	C3
a	b	c
a	d	g
f	b	e
a	b	e
h	d	g

### R(参加体育队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	20	04	980402
98040202	王四	男	21	04	980402
98040203	王五	女	19	04	980402

### S(参加文艺队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98020101	孙三	女	18	02	980201
98020102	孙四	男	20	02	980201
98020103	孙五	女	19	02	980201
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301

### RUS(或者参加体育队或者文艺队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	20	04	980402
98040202	王四	男	21	04	980402
98040203	王五	女	19	04	980402
98020101	孙三	女	18	02	980201
98020102	孙四	男	20	02	980201
98020103	孙五	女	19	02	980201

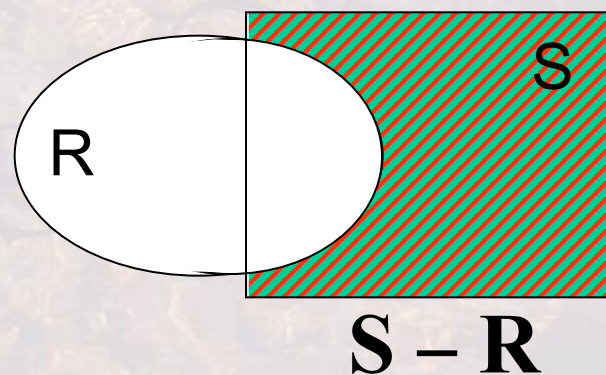
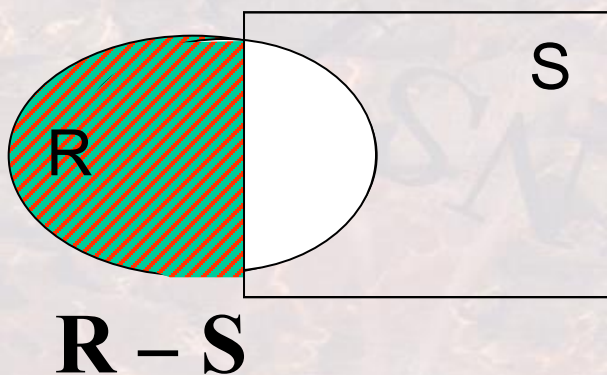
## (3)什么情况用差运算呢？

### 差(Difference)

□ **定义**：假设关系R 和关系S是并相容的，则关系R 与关系S 的差运算结果也是一个关系，记作： $R - S$ ，它由**出现在关系R 中但不出现在关系S中的元组**构成

□ **数学描述**： $R - S = \{ t \mid t \in R \wedge t \notin S \}$ ，其中t是元组

□ **注意**： $R - S$  与  $S - R$  是不同的



# 基本数据模型：关系模型II-关系运算之并-差-交-积

## (3)什么情况用差运算呢？

### 差(Difference)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

R - S		
D1	D2	D3
f	b	e

S - R		
E1	E2	E3
a	b	e
h	d	g

### R(参加体育队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	20	04	980402
98040202	王四	男	21	04	980402
98040203	王五	女	19	04	980402

### R - S(参加体育队而未参加文艺队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98040201	王三	男	20	04	980402
98040202	王四	男	21	04	980402
98040203	王五	女	19	04	980402

### S(参加文艺队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98020101	孙三	女	18	02	980201
98020102	孙四	男	20	02	980201
98020103	孙五	女	19	02	980201
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301

### S - R(参加文艺队而未参加体育队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98020101	孙三	女	18	02	980201
98020102	孙四	男	20	02	980201
98020103	孙五	女	19	02	980201



## (4)什么情况用交运算呢？

### 交(Intersection)

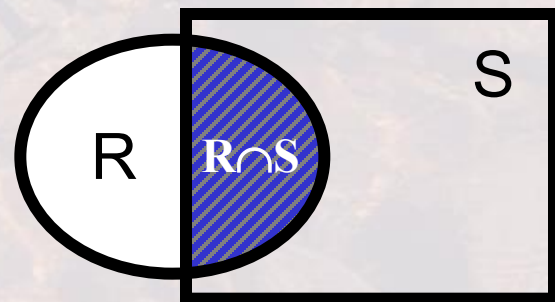
□ **定义**：假设关系R和关系S是并相容的，则关系R与关系S的交运算结果也是一个关系，记作： $R \cap S$ ，它由**同时出现在**关系R和关系S中的元组构成

□ **数学描述**： $R \cap S = \{ t \mid t \in R \wedge t \in S \}$ ，其中t是元组

□  $R \cap S$  和  $S \cap R$  运算的结果是同一个关系

□ 交运算可以通过差运算来实现：

$$R \cap S = R - (R - S) = S - (S - R)$$



# 基本数据模型：关系模型II-关系运算之并-差-交-积

## (4)什么情况用交运算呢？

### 交(Intersection)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

$R \cap S$		
F1	F2	F3
a	b	c
a	d	g

### R(参加体育队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	20	04	980402
98040202	王四	男	21	04	980402
98040203	王五	女	19	04	980402

### $R \cap S$ (既参加体育队又参加文艺队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301

### S(参加文艺队的学生)

S#	Sname	Ssex	Sage	D#	Sclass
98020101	孙三	女	18	02	980201
98020102	孙四	男	20	02	980201
98020103	孙五	女	19	02	980201
98030101	张三	男	20	03	980301
98030102	张四	女	20	03	980301
98030103	张五	男	19	03	980301

## (5)什么情况用笛卡尔积运算呢？

### 广义笛卡尔积 (Cartesian Product)

□ **定义**：关系 $R(<a_1, a_2, \dots, a_n>)$  与关系 $S(<b_1, b_2, \dots, b_m>)$  的广义笛卡尔积 (简称广义积) 运算结果也是一个关系，记作： $R \times S$ ，它由关系 $R$ 中的元组与关系 $S$ 的元组进行所有可能的拼接(或串接)构成。

□ **数学描述**： $R \times S = \{ <a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m> \mid <a_1, a_2, \dots, a_n> \in R \wedge <b_1, b_2, \dots, b_m> \in S \}$

□ 笛卡尔积可将两个表串接起来作为一个表进行操作

学生表			
学号	姓名	年龄	住址
981101	李四	22	3010
981103	李三	21	3011
981105	李六	22	3011

课程表			
课程号	课程名	教师	学时
C1	计算机	教师 1	52
C2	物理	教师 2	36
C3	高数	教师 5	40

(所有学生)的(所有课程)

学号	姓名	年龄	住址	课程号	课程名	教师	学时
981101	李四	22	3010	C1	计算机	教师 1	52
981101	李四	22	3010	C2	物理	教师 2	36
981101	李四	22	3010	C3	高数	教师 5	40
981103	李三	21	3011	C1	计算机	教师 1	52
981103	李三	21	3011	C2	物理	教师 2	36
981103	李三	21	3011	C3	高数	教师 5	40
981105	李六	22	3011	C1	计算机	教师 1	52
981105	李六	22	3011	C2	物理	教师 2	36
981105	李六	22	3011	C3	高数	教师 5	40



# 基本数据模型：关系模型II-关系运算之并-差-交-积

## (5)什么情况用笛卡尔积运算呢？

### 广义笛卡尔积 (Cartesian Product)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

S		
B1	B2	B3
a	b	c
a	b	e
a	d	g
h	d	g

R × S					
A1	A2	A3	B1	B2	B3
a	b	c	a	b	c
a	b	c	a	b	e
a	b	c	a	d	g
a	b	c	h	d	g
a	d	g	a	b	c
a	d	g	a	b	e
a	d	g	a	d	g
a	d	g	h	d	g
f	b	e	a	b	c
f	b	e	a	b	e
f	b	e	a	d	g
f	b	e	h	d	g

Relations  $r, s$ :

A	B
$\alpha$	1
$\beta$	2

C	D	E
$\alpha$	10	a
$\beta$	10	a
$\beta$	20	b
$\gamma$	10	b

$r \times s$ :

A	B	C	D	E
$\alpha$	1	$\alpha$	10	a
$\alpha$	1	$\beta$	19	a
$\alpha$	1	$\beta$	20	b
$\alpha$	1	$\gamma$	10	b
$\beta$	2	$\alpha$	10	a
$\beta$	2	$\beta$	10	a
$\beta$	2	$\beta$	20	b
$\beta$	2	$\gamma$	10	b

# 基本数据模型：关系模型II-关系运算之并-交-差-积

## (4)小结？

数据库的关系模型起源于规范化 “表(Table)”的处理

Table: 以按行按列形式组织及展现的数据



Table中描述了一批相互有关联关系的数据==>关系

# 基本数据模型：关系模型III

## -关系运算之选择、投影、连接



□ **定义**：给定一个关系R, 同时给定一个选择的条件condition(简记con), 选择运算结果也是一个关系, 记作 $\sigma_{con}(R)$ , 它从关系R中选择出满足给定条件condition的元组构成

□ 数学描述:  $\sigma_{\text{con}}(R) = \{t \mid t \in R \wedge \text{con}(t) = \text{'真'}\}$ ,

✓ 设  $R(A_1, A_2, \dots, A_n)$ ,  $t$  是  $R$  的元组,  $t$  的分量记为  $t[A_i]$ , 或简写为  $A_i$

✓ 条件con由逻辑运算符连接算术/比较表达式组成

✓ 逻辑运算符:  $\wedge$ ,  $\vee$ ,  $\neg$  或写为 and, or, not

✓ 算术/比较表达式:  $X \theta Y$ , 其中 $X, Y$ 是 $t$ 的分量、常量或简单函数,  $\theta$ 是比较运算符,  $\theta \in \{>, \geq, <, \leq, =, \neq\}$

[illegible]

# 基本数据模型：关系模型III-关系运算之选择-投影-连接

## (1)什么情况用选择运算呢？

### 选择(Selection)

R		
A1	A2	A3
a	a	10
a	d	-4
f	b	5

$\sigma_{A3>0}(R)$		
A1	A2	A3
a	a	10
f	b	5

$\sigma_{A2="a" \vee A2="b"}(R)$		
A1	A2	A3
a	a	10
f	b	5

$\sigma_{A3>0 \wedge A1=A2}(R)$		
A1	A2	A3
a	a	10

### R(学生表)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	21	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	18	04	980402
98040202	王四	男	21	04	980402
98050104	孙六	女	19	05	980501

### 查询所有年龄小于20同学的信息

$\sigma_{Sage<20}(R)$

S#	Sname	Ssex	Sage	D#	Sclass
98030103	张五	男	19	03	980301
98040201	王三	男	18	04	980402
98050104	孙六	女	19	05	980501

### 查询所有3系或5系的同学信息

$\sigma_{D\#="03" \vee D\#="05"}(R)$

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	21	03	980301
98030103	张五	男	19	03	980301
98050104	孙六	女	19	05	980501

□ **定义**：给定一个关系R, 投影运算结果也是一个关系，记作  $\Pi_A(R)$ ，它从关系R中选出属性包含在A中的列构成

□ 数学描述:  $\Pi_{A_{i1}, A_{i2}, \dots, A_{ik}}(R) = \{ \langle t[A_{i1}], t[A_{i2}], \dots, t[A_{ik}] \rangle \mid t \in R \}$

- ✓ 设  $R(A_1, A_2, \dots, A_n)$
- ✓  $\{A_{i1}, A_{i2}, \dots, A_{ik}\} \subseteq \{A_1, A_2, \dots, A_n\}$
- ✓  $t[A_i]$  表示元组  $t$  中相应于属性  $A_i$  的分量
- ✓ 投影运算可以对原关系的列在投影后重新排列

R		
A1	A2	A3



# 基本数据模型：关系模型III-关系运算之选择-投影-连接

## (2)什么情况用投影运算呢？

### 投影(Projection)

R		
A1	A2	A3
a	b	c
a	d	g
f	b	e

$\Pi_{A3}(R)$
A3
c
g
e

$\Pi_{A3, A1}(R)$	
A3	A1
c	a
g	a
e	f

### R(学生表)

S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	21	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	18	04	980402
98040202	王四	男	21	04	980402
98050104	孙六	女	19	05	980501

### $\Pi_{Sname, Sage}(R)$

查询所有学生的姓名和年龄

Sname	Sage
张三	20
张四	21
张五	19
王三	18
王四	21
孙六	19

### $\Pi_{Sname, D#}(R)$

查询所有学生的姓名及其所在的系

Sname	D#
张三	03
张四	03
张五	03
王三	04
王四	04
孙六	05

## (3)什么情况用连接运算呢？

### $\theta$ -连接( $\theta$ -Join)

□ **定义**：给定关系R和关系S, R与S的 $\theta$ 连接运算结果也是一个关系，记作  $R \bowtie_{A \theta B} S$ ，它由关系R和关系S的笛卡尔积中，选取R中属性A与S中属性B之间满足 $\theta$ 条件的元组构成。

□ **数学描述**：
$$R \bowtie_{A \theta B} S = \sigma_{t[A] \theta s[B]} (R \times S)$$

✓ 设 $R(A_1, A_2, \dots, A_n)$ ,  $A \in \{A_1, A_2, \dots, A_n\}$

✓  $S(B_1, B_2, \dots, B_m)$ ,  $B \in \{B_1, B_2, \dots, B_m\}$

✓ t是关系R中的元组，s是关系S中的元组

✓ 属性A和属性B具有可比性

✓  $\theta$ 是比较运算符,  $\theta \in \{>, \geq, <, \leq, =, \neq\}$

□ 在实际应用中， $\theta$ -连接操作经常与投影、选择操作一起使用

## (3)什么情况用连接运算呢？

### 自然连接(Natural-Join)

□ **定义**：给定关系R和关系S, R与S的自然连接运算结果也是一个关系，记作  $R \bowtie S$  它由关系R和关系S的笛卡尔积中选取相同属性组B上值相等的元组所构成。

□ **数学描述**：

$$R \bowtie S = \sigma_{t[B] = s[B]} (R \times S)$$

- ✓ 自然连接是一种特殊的连接运算
- ✓ 要求关系R和关系S必须有相同的属性组B(如R,S共有一个属性 $B_1$ ,则B是 $B_1$ ，如R, S共有一组属性 $B_1, B_2, \dots, B_n$ ，则B是这些共有的所有属性)
- ✓ R, S属性相同，值必须相等才能连接，即 $R.B_1 = S.B_1$  and  $R.B_2 = S.B_2 \dots$  and  $R.B_n = S.B_n$ 才能连接
- ✓ 要在结果中去掉重复的属性列(因结果中 $R.B_i$ 始终是等于 $S.B_i$ 所以可只保留一列即可)



# 基本数据模型：关系模型III-关系运算之选择-投影-连接

## (3)什么情况用连接运算呢？

### $\theta$ -连接 vs. 连接 vs. 笛卡尔积

R	
A	B
a	1
b	2

S	
B	C
1	x
1	y
3	z

$R \times S$			
A	B	B	C
a	1	1	x
a	1	1	y
a	1	3	z
b	2	1	x
b	2	1	y
b	2	3	z

$R \bowtie S$		
A	B	C
a	1	x
a	1	y

R	
A	B
a	1
b	2

S	
H	C
1	x
1	y
3	z

$R \times S$			
A	B	H	C
a	1	1	x
a	1	1	y
a	1	3	z
b	2	1	x
b	2	1	y
b	2	3	z

$R \bowtie_{B \leq H} S$			
A	B	H	C
a	1	1	x
a	1	1	y
a	1	3	z
b	2	3	z

# 基本数据模型：关系模型III-关系运算之选择-投影-连接

## (4)小结？

数据库的关系模型起源于规范化 “表(Table)” 的处理

Table: 以按行按列形式组织及展现的数据



Table中描述了一批相互有关联关系的数据==>关系

# 应用关系运算进行数据库的查询



# 应用关系运算进行数据库的查询

## (1)利用关系运算进行查询?

### 查询表达式

### 组合各种运算

Student					
S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	21	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	18	04	980402
98040202	王四	男	21	04	980402
98050104	孙六	女	19	05	980501

SC		
S#	C#	Score
98030101	001	92.0
98030101	002	85.0
98030101	003	88.0
98040202	002	90.5
98040202	003	80.0
98040202	001	55.0
98050104	003	56.0
98030102	001	54.0
98030102	002	85.0
98030102	003	48.0

- 查询学习课程号为002的学生学号和成绩

$\pi_{S\#, Score}(\sigma_{C\#="002"}(SC))$

- 查询学习课程号为001的学生学号、姓名

$\pi_{S\#, Sname}(\sigma_{C\#="001"}(Student \bowtie SC))$

Course				
C#	Cname	Chours	Credit	T#
001	数据库	40	6	001
003	数据结构	40	6	003
004	编译原理	40	6	001
005	C 语言	30	4.5	003
002	高等数学	80	12	004

- 查询学习课程名称为数据结构的学生学号、姓名和这门课程的成绩

$Student \bowtie SC \bowtie Course$

$\sigma_{Cname="数据结构"}(Student \bowtie SC \bowtie Course)$

$\pi_{S\#, Sname, Score}(\sigma_{Cname="数据结构"}(Student \bowtie SC \bowtie Course))$

# 应用关系运算进行数据库的查询

## (1)利用关系运算进行查询？

### 查询表达式

### 注意连接与积的差别

Student					
S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	21	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	18	04	980402
98040202	王四	男	21	04	980402
98050104	孙六	女	19	05	980501

SC		
S#	C#	Score
98030101	001	92.0
98030101	002	85.0
98030101	003	88.0
98040202	002	90.5
98040202	003	80.0
98040202	001	55.0
98050104	003	56.0
98030102	001	54.0
98030102	002	85.0
98030102	003	48.0

➤ 查询学习课程号为001的学生学号、姓名

$\pi_{S\#,Sname}(\sigma_{C\#="001"}(Student \bowtie SC))$

Course				
C#	Cname	Chours	Credit	T#
001	数据库	40	6	001
003	数据结构	40	6	003
004	编译原理	40	6	001
005	C 语言	30	4.5	003
002	高等数学	80	12	004

$\pi_{S\#,Sname}(\sigma_{C\#="001" \wedge Student.S\# = SC.S\#}(Student \times SC))$

连接条件

# 应用关系运算进行数据库的查询

## (2)由关系模型到结构化数据库语言SQL

### 关系运算式

$\Pi$  列名, ..., 列名 ( $\sigma$  检索条件 (表名1  $\times$  表名2  $\times$  ...))

$\Pi$  S#, Sname, Score ( $\sigma$  Cname="数据结构"  $\wedge$  Student.S#=SC.S#  $\wedge$  Course.C#=SC.C# (Student  $\times$  SC  $\times$  Course))

### 数据库语言SQL

**Select** 列名 [[, 列名] ... ]

**From** 表名1 [[, 表名2], ...]

[ **Where** 检索条件 ] ;

语义：将From后面的所有表串接起来，检索出满足“检索条件”的元组，并按给定的列名及顺序进行投影显示。

**Select** S#, Sname, Score

**From** Student, SC, Course

**Where** Cname= '数据结构' **and** Student.S#=SC.S# **and** Course.C#=SC.C#;



# 应用关系运算进行数据库的查询

## (3)小结?

### 查询表达式

### 组合各种运算

Student					
S#	Sname	Ssex	Sage	D#	Sclass
98030101	张三	男	20	03	980301
98030102	张四	女	21	03	980301
98030103	张五	男	19	03	980301
98040201	王三	男	18	04	980402
98040202	王四	女	19	04	980402
98050301	李一	男	20	05	980503

SC		
S#	C#	Score
98030101	001	92.0
98030101	002	85.0
98030101	003	88.0
98040202	002	90.5

关系

数据库

表

关系的基本操作:并/差/积/选择/投影/连接

数据库查询:关系基本操作的各种组合,构造查询表达式—即关系运算式

都是围绕表来进行:两个表串接起来(积和连接),从表中选出若干行(选择),从表中选出若干列(投影),两个表的合并(并)等

➤ 查询学习课程

➤ 查询学习课程号为001的学生姓名

$\pi_{S\#,Sname}(\sigma_{C\#=001}(Student \bowtie SC))$

➤ 查询学习课程名称为数据结构的学生的学号、姓名和这门课程的成绩

Student  $\bowtie$  SC  $\bowtie$  Course

$\sigma_{Cname='数据结构'}(Student \bowtie SC \bowtie Course)$

$\pi_{S\#,Sname,Score}(\sigma_{Cname='数据结构'}(Student \bowtie SC \bowtie Course))$

# SQL数据库的操作

# SQL语言

## ➤ SQL (Structured Query Language)

- ❑ 结构化查询语言，是关系数据库的标准
- ❑ 包括数据查询、数据库模式创建、数据库数据的增删改、数据库安全性和完整性定义与控制等



# SQL语言的特点

- 功能综合且风格统一
  - ❑ 集DDL、DML、DCL为一体
  - ❑ 实体和联系都是关系，因此每种操作只需一种操作符
- 数据操纵高度非过程化
- 面向集合的操作方式
- 以统一的语法结构提供两种使用方式（交互式 and 嵌入式）
- 语言简捷，易学易用
  - ❑ 数据定义 CREATE、DROP、ALTER
  - ❑ 数据查询 SELECT
  - ❑ 数据更新 INSERT、UPDATE、DELETE
  - ❑ 数据控制 GRANT、REVOKE

# SQL的基本概念

## ➤ 基本表

- ❑ 本身独立存在的表
- ❑ 关系数据库管理系统中一个关系就对应一个基本表
- ❑ 一个或多个基本表对应一个存储文件
- ❑ 一个表可以带若干索引

## ➤ 存储文件

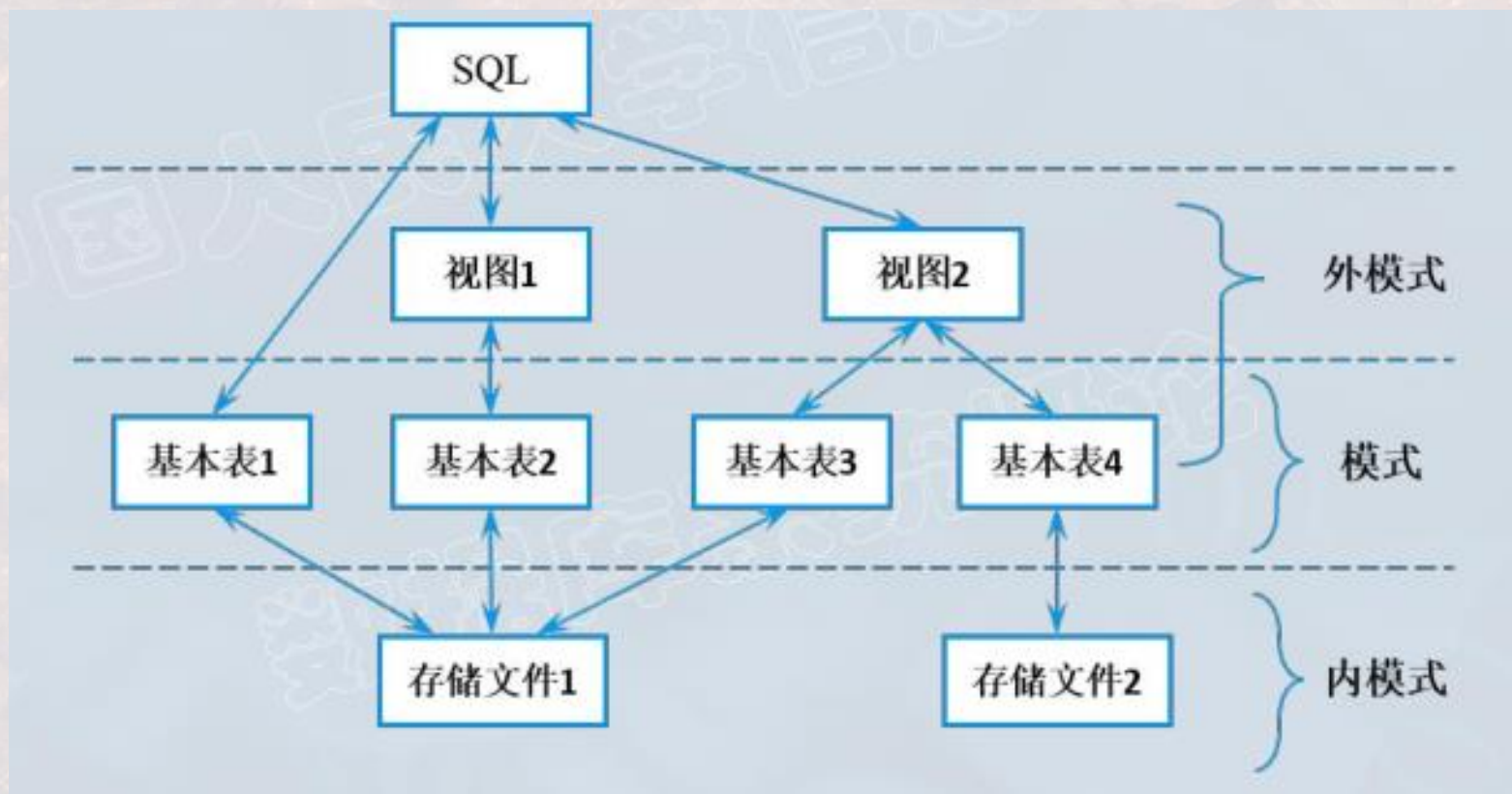
- ❑ 逻辑结构和物理结构组成了关系数据库的内模式
- ❑ 物理文件结构是由数据库管理系统设计确定的

## ➤ 视图

- ❑ 从基本表或其他视图中导出的表
- ❑ 视图是一个虚表，数据库中只存放视图的定义而不存放视图对应的数据

# SQL的基本概念

## ➤ SQL对关系数据库三级模式的支持





# SQL的数据定义

➤ 定义和修改基本表（定义模式中的关系）：

❑ CREATE TABLE

❑ DROP TABLE

❑ ALTER TABLE

➤ 定义视图（定义外模式）：

❑ CREATE VIEW

❑ DROP VIEW

➤ 定义索引（定义内模式）：

❑ CREATE INDEX

❑ DROP INDEX

# 基本表的定义

## ➤ 基本格式

CREATE TABLE <表名>

(<列名> <数据类型>[ <列级完整性约束条件> ]  
[, <列名> <数据类型>[ <列级完整性约束条件> ] ] ...  
[, <表级完整性约束条件> ] ) ;

<表名>: 所要定义的基本表的名字

<列名>: 组成该表的各个属性（列）

<列级完整性约束条件>: 涉及相应属性列的完整性约束条件

<表级完整性约束条件>: 涉及一个或多个属性列的完整性约束条件

# 基本表的定义

[例1] 建立一个“学生”表S，它由学号Sno、姓名Sname、性别Ssex、年龄Sage、所在系Sdept五个属性组成。其中学号不能为空，值是唯一的，并且姓名取值也唯一。

```
CREATE TABLE S
```

```
(Sno    CHAR(5) NOT NULL UNIQUE,  
 Sname  CHAR(20) UNIQUE,  
 Ssex    CHAR(1) ,  
 Sage    INT,  
 Sdept   CHAR(15));
```



# 基本表的定义

## ➤ 说明:

- ❑ SQL支持空值的概念。允许空值的列未输入数据时系统自动置为空值。
- ❑ SQL支持的数据类型随系统不同而有所差异，但一般都有：  
全字长整型、半字长整型、定点实型、浮点实型、  
CHAR(n)、VARCHAR(n)、TEXT、DATE



# 基本表的定义

## ➤ 常用完整性约束

- ❑ 主码约束: PRIMARY KEY
- ❑ 唯一性约束: UNIQUE
- ❑ 非空值约束: NOT NULL
- ❑ 参照完整性约束 FOREIGN KEY
- ❑ CHECK约束

### 思考: PRIMARY KEY与 UNIQUE的区别?

- ❑ PRIMARY KEY在建立的时候会默认地建立此field的索引, 且此PRIMARY KEY可以作为作为另外的表的FOREIGN KEY;
- ❑ 再者PRIMARY KEY跟UNIQUE的区别是PRIMARY KEY一定是NOT NULL, 而UNIQUE则没有此限制

# 基本表的定义

- [例2] 建立一个“课程”表C，它由课程号Cno，课程名称Cname，教师Teacher组成，其中Cno为主码。

```
CREATE TABLE C(  
    Cno CHAR(5) ,  
    Cname CHAR(10) ,  
    Teacher CHAR(10),  
    PRIMARY KEY(Cno));
```

或像书上写的将PRIMARY KEY直接定义在属性列后。

# 基本表的定义

- [例3] 建立一个“学生选课”表SC，它由学号Sno、课程号Cno，修课成绩Grade组成，其中(Sno, Cno)为主码。

```
CREATE TABLE SC(  
    Sno CHAR(5),  
    Cno CHAR(3),  
    Grade int,  
    PRIMARY KEY(Sno, Cno),  
    /*主码由两个属性构成，必须作为表级完整性进行定义*/  
    FOREIGN KEY(Sno) REFERENCES S(Sno),  
    /* 表级完整性约束，Sno 是外码，被参照表是S*/  
    FOREIGN KEY(Cno) REFERENCES C(Cno)  
    /* 表级完整性约束，Cno 是外码，被参照表是C*/  
);
```



# 基本表的删除

## ➤基本格式

**DROP TABLE** 表名 ;

## ➤示例

DROP TABLE S;



# 基本表的修改

## ➤ 基本格式

**ALTER TABLE <表名>**

**[ ADD <新列名> <数据类型> [ 完整性约束 ] ]**

**[ DROP <完整性约束名> ]**

**[ ALTER COLUMN <列名> <数据类型> ];**

**<表名>:** 要修改的基本表

**ADD子句:** 增加新列和新的完整性约束条件

**DROP [COLUMN]子句:** 删除表中的列

**DROP CONSTRAINT:** 删除指定的完整性约束条件

**ALTER COLUMN子句:** 用于修改列名和数据类型

# 基本表的修改

## ➤ 增加列基本格式

**ALTER TABLE** 表名 **ADD** 列名 类型;

❑ 示例 ALTER TABLE S ADD ADDRESS VARCHAR (30)

## ➤ 修改列基本格式

**ALTER TABLE** 表名 **ALTER COLUMN** 列名 类型;

❑ ALTER TABLE S ALTER COLUMN SA SMALLINT;

## ➤ 删除列基本格式

**ALTER TABLE** 表名 **DROP** 列名;

❑ ALTER TABLE S DROP UNIQUE(Sn);

# 例子 修改表

- 向student表中添加一列sno

ALTER TABLE student **ADD** sno CHAR(5) NULL

- 向student表添加主键约束

ALTER TABLE student **ADD PRIMARY KEY**(sno)

- 从student表删除列sex

ALTER TABLE student **DROP COLUMN** sex



# 例子 修改表

## ➤ 向study表中添加外键约束

**ALTER** TABLE study

**ADD CONSTRAINT** fkcno FOREIGN KEY(cno) REFERENCES course(cno)  
ON DELETE CASCADE ON UPDATE CASCADE

## ➤ 向study表中添加外键约束

**ALTER** TABLE study

**ADD CONSTRAINT** fksno FOREIGN KEY(sno) REFERENCES student(sno)  
ON DELETE CASCADE ON UPDATE CASCADE



# 视图的定义和修改

## ➤ 基本操作

- ❑ CREATE VIEW

- ❑ DROP VIEW

## ➤ 后面详细讨论



# 索引的建立和删除

- 建立索引是加快查询速度的有效手段
- 数据库常见索引：
  - 顺序文件上的索引
  - B+树索引
  - 哈希（hash）索引
  - 位图索引
- 特点：
  - B+ 树索引具有动态平衡的优点
  - 哈希索引具有查找速度快的特点

# 索引的建立和删除

## ➤ 谁可以建立与删除索引

- ❑ 数据库管理员或表的属主（即建立表的人）

## ➤ 谁维护索引

- ❑ 关系数据库管理系统自动完成

## ➤ 使用索引

- ❑ 关系数据库管理系统自动选择是否使用索引或使用哪个索引作为存取路径，用户不必也不能自主地选择索引。



# 索引的建立

## ➤ 索引建立的基本格式

一个索引项对应一个记录

**聚簇索引**

CREATE [ **UNIQUE** ] [ **CLUSTER** ] INDEX 索引名 **升序或降序**

ON 表名 ( 列名  $\begin{pmatrix} \text{ASC} \\ \text{DESC} \end{pmatrix}$  [, 列名  $\begin{pmatrix} \text{ASC} \\ \text{DESC} \end{pmatrix}$  ]... ) ;

**Unique**索引表明此索引的每个索引值只对应唯一的数据记录  
**Cluster**表示要建立的索引是聚簇索引。

# 索引的建立

例、为学生-课程数据库中的S，C，SC三个表建立索引。其中S表按学号升序建唯一索引，C表按课程号升序建唯一索引，SC表按学号升序和课程号降序建唯一索引(即先按学号升序，对同一个学号再按课程号降序)。

```
CREATE UNIQUE INDEX Ssno ON S(Sno);
```

/\* 保证了Sno取唯一值的约束\*/

```
CREATE UNIQUE INDEX Ccno ON C(Cno);
```

/\* 加上Cno取唯一值的约束\*/

```
CREATE UNIQUE INDEX SCno ON SC(Sno ASC, Cno  
DESC);
```

# 索引的建立

## ➤ 唯一值索引

- 对于已含重复值的属性列不能建UNIQUE索引
- 对某个列建立UNIQUE索引后，插入新记录时DBMS会自动检查新记录在该列上是否取了重复值。这相当于增加了一个UNIQUE约束

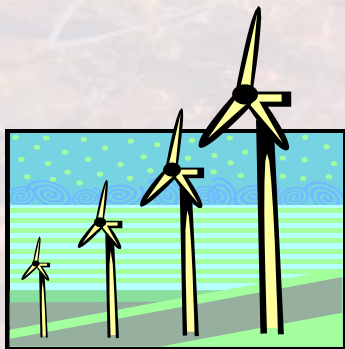


# 索引的删除

## ➤ 索引删除的基本格式

**DROP INDEX** 索引名

- ❑ 删除索引时，系统会从数据字典中删去有关该索引的描述。



# SQL的数据操纵

- SQL的数据查询（检索）

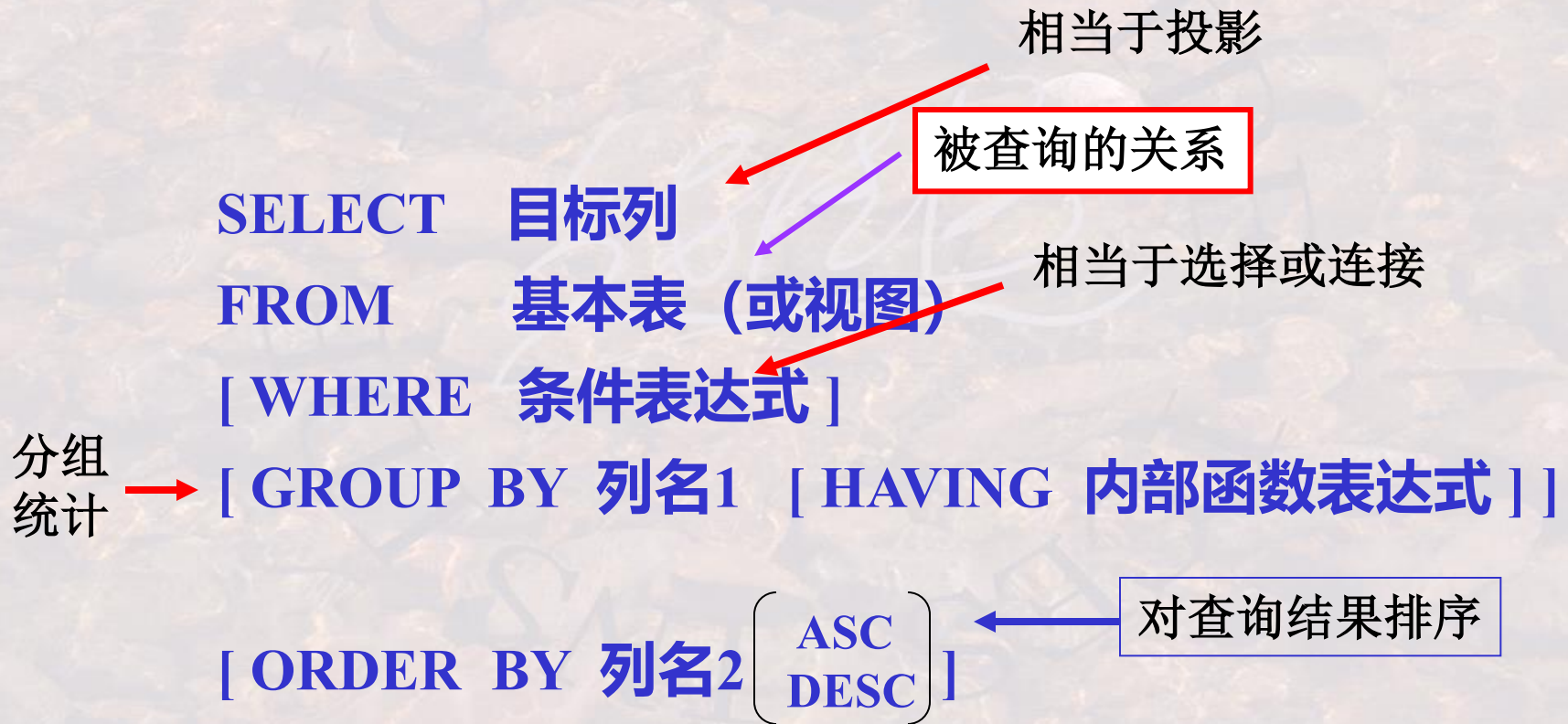
**SELECT**

- SQL的数据更新（增、删、改）

**INSERT, DELETE, UPDATE**

# SQL的数据查询

## ➤ 一般格式





# SQL的数据查询

## ■ 常用格式(SELECT- FROM- WHERE句型)

```
SELECT  A1,A2, ...,An  
FROM    R1,R2, ...,Rm  
WHERE   F
```

其中，条件表达式F可使用下列操作符：

- ➡ 算术比较运算符 (<, <=, >, >=, =, <>)
- ➡ 逻辑运算符 (AND, OR, NOT)
- ➡ 集合运算符 (UNION, INTERSECT, EXCEPT)
- ➡ 集合成员资格运算符 (IN, NOT IN)
- ➡ 谓词 (EXISTS, ALL, SOME, UNIQUE)
- ➡ 聚合函数 (AVG, MIN, MAX, SUM, COUNT)
- ➡ 嵌套的SELECT语句

# 示例数据库

## 学生-课程数据库

- 学生表: Student(Sno, Sname, Ssex, Sage, Sdept)
- 课程表: Course(Cno, Cname, Cpno, Ccredit)
- 学生选课表: SC(Sno, Cno, Grade)

# 单表查询

➤ 查询仅涉及一个表，是一种最简单的查询操作

□ 选择表中的若干列

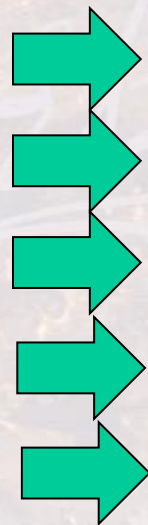
□ 选择表中的若干元组

□ 对查询结果排序

□ 使用聚集函数

□ 对查询结果分组

- 使用HAVING短语





# 查询指定列

[例1] 查询全体学生的学号与姓名。

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

[例2] 查询全体学生的姓名、学号、所在系。

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

[例3] 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

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

[例4] 查全体学生的姓名及其出生年份。

```
SELECT Sname, 2009-Sage  
FROM Student;
```

输出结果:

Sname	2009-Sage
李勇	1976
刘晨	1977
王名	1978
张立	1978

# 查询经过计算的值

[例5] 查询全体学生的姓名、出生年份和所有系，要求用小写字母表示所有系名。

```
SELECT Sname, 'Year of Birth: ', 2009-Sage,  
       ISLOWER(Sdept)
```

```
FROM Student;
```

输出结果:

Sname	'Year of Birth:'	2009-Sage	ISLOWER(Sdept)
李勇	Year of Birth:	1976	cs
刘晨	Year of Birth:	1977	is
王名	Year of Birth:	1978	ma
张立	Year of Birth:	1977	is



# 使用列别名改变查询结果的列标题

```
SELECT Sname NAME, 'Year of Birth: ' BIRTH,  
2009-Sage BIRTHDAY, ISLOWER(Sdept) DEPARTMENT  
FROM Student;
```

输出结果:

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1976	cs
刘晨	Year of Birth:	1977	is
王名	Year of Birth:	1978	ma
张立	Year of Birth:	1977	is

# 选择表中的若干元组

- 消除取值重复的行
- 查询满足条件的元组
  - ① 比较大小
  - ② 确定范围
  - ③ 确定集合
  - ④ 字符串匹配
  - ⑤ 涉及空值的查询
  - ⑥ 多重条件查询

# 消除取值重复的行

## ➤ 在SELECT子句中使用DISTINCT短语

假设SC表中有下列数据

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



# ALL 与 DISTINCT

[例6] 查询选修了课程的学生学号。

(1) SELECT Sno  
FROM SC;

等价于(默认 ALL)

SELECT ALL Sno  
FROM SC;

结果: Sno  
-----  
95001  
95001  
95001  
95002  
95002

(2) SELECT **DISTINCT** Sno  
FROM SC;

结果:

Sno  
-----  
95001  
95002

# ALL 与 DISTINCT

➤ 注意 **DISTINCT**短语的作用范围是所有目标列

例：查询选修课程的各种成绩

❑ 错误写法

```
SELECT DISTINCT Cno, DISTINCT Grade  
FROM SC;
```

❑ 正确写法

```
SELECT DISTINCT Cno, Grade  
FROM SC;
```

# 查询满足条件的元组

- ① 比较大小
- ② 确定范围
- ③ 确定集合
- ④ 字符串匹配
- ⑤ 涉及空值的查询
- ⑥ 多重条件查询



# ① 比较大小

在WHERE子句的<比较条件>中使用**比较运算符**

□=, >, <, >=, <=, != 或 <>, !>, !<,

□逻辑运算符NOT + 比较运算符

[例8] 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;
```

或

```
SELECT Sname, Sage
```

```
FROM Student
```

```
WHERE NOT Sage >= 20;
```

## ②确定范围

使用**谓词** BETWEEN ... AND ...

NOT BETWEEN ... AND ...

[例10] 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11] 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

使用谓词 **IN** <值表>, **NOT IN** <值表>

<值表>: 用逗号分隔的一组取值

[例12]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' );
```



## ④字符串匹配

➤ [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 ) : 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

▣ \_ ( 下横线 ) : 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 的任意字符

▣ 当用户要查询的字符串本身含有%或\_时, 要使用ESCAPE ' <换码字符>' 短语对通配符进行转义

## ④字符串匹配（续）

匹配模板为**固定字符串**

[例14] 查询学号为95001的学生的详细情况。

```
SELECT *  
FROM Student  
WHERE Sno LIKE '95001';
```

等价于：

```
SELECT *  
FROM Student  
WHERE Sno = '95001';
```

## ④字符串匹配（续）

匹配模板为**含通配符**的字符串

[例15] 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%';
```

[例16] 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17] 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%';
```



## ④字符串匹配（续）

使用换码字符将通配符转义为普通字符

[例19] 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 ' \'** 表示 \ 为换码字符
- **\_**前面有换码字符\， 被转义为普通的\_ 字符

## ⑤涉及空值的查询

- 使用谓词 **IS NULL** 或 **IS NOT NULL**
- “IS NULL” 不能用 “= NULL” 代替

[例21] 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22] 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

用逻辑运算符**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23] 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;
```



# 对查询结果排序

## 使用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

# 对查询结果排序

**[例24]** 查询选修了3号课程的学生们的学号及其成绩，  
查询结果按分数降序排列。

```
SELECT Sno, Grade
```

```
FROM SC
```

```
WHERE Cno= '3 '
```

```
ORDER BY Grade DESC;
```

Sno	Grade
-----	-----
95010	
95024	
95007	92
95003	82
95010	82
95009	75
95014	61
95002	55

# 使用聚集函数

## ➤ 统计元组个数

▣ COUNT(\*)

## ➤ 统计一列中值的个数

▣ COUNT([DISTINCT|ALL] < 列名>)

## ➤ 计算一列值的总和（此列必须为数值型）

▣ SUM([DISTINCT|ALL] < 列名>)

## ➤ 计算一列值的平均值（此列必须为数值型）

▣ AVG([DISTINCT|ALL] < 列名>)

## ➤ 求一列中的最大值和最小值

▣ MAX([DISTINCT|ALL] < 列名>)

▣ MIN([DISTINCT|ALL] < 列名>)



# 使用聚集函数

**[例25] 求选修了课程的学生人数。**

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



重复的只记一个

# 对查询结果分组

## ➤ 使用GROUP BY子句分组

- 按指定的一列或多列值分组，值相等的为一组

## ➤ GROUP BY + 聚集函数

- 聚集函数作用于每一个组

## ➤ 无GROUP BY 时使用聚集函数

- 聚集函数作用于整个查询结果

# 对查询结果分组

[例30] 求各个课程号及相应的选课人数。

```
SELECT Cno, COUNT(Sno)
FROM SC
GROUP BY Cno;
```

结果

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

对查询结果按Cno的值分组。  
所有具有相同Cno值的元素  
为一组，每组应用COUNT  
计算



# 使用HAVING短语

[例31] 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;
```

对查询结果按Sno的值分组。  
HAVING短语给出了选择组  
的条件：元组个数>3

# 使用HAVING短语

[例32] 查询有3门以上课程是90分以上的  
学生的学号及（90分以上的）课程数

```
SELECT Sno, COUNT(*)  
FROM SC  
WHERE Grade>=90  
GROUP BY Sno  
HAVING COUNT(*)>=3;
```

先处理WHERE子句;  
再分组,  
最后用HAVING条件中的聚集函数对每一组计数

# 使用HAVING短语

- 只有满足**HAVING**短语指定条件的组才输出
- **HAVING**短语与**WHERE**子句的区别：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



# 使用聚合函数对数据进行汇总和统计

- 聚合函数常见的几种用法：
- **SUM ( [ ALL | DISTINCT ] expression )** 返回表达式中所有值的和，或只返回DISTINCT值。  
SUM 只能用于数字列。空值将被忽略。
- **COUNT ( [ ALL | DISTINCT ] expression ] | \* )** 返回组中项目的数量。
- **COUNT(\*)** 返回组中项目的数量，这些项目包括 NULL 值和重复值。
- **COUNT(ALL expression)** 对组中的每一行都计算相应的expression的值，并返回非空值的expression的数量。
- **COUNT(DISTINCT expression)** 对组中的每一行都计算 expression并返回唯一的非空值的数量。

# 使用聚合函数对数据进行汇总和统计

- **AVG ( [ ALL | DISTINCT ] expression )** 返回组中值的平均值。空值将被忽略。
- **MAX ( [ ALL | DISTINCT ] expression )** 返回表达式的最大值
- **MIN ( [ ALL | DISTINCT ] expression )** 返回表达式的最小值。
- ❁ 聚合函数一般仅能出现在**SELECT**和**HAVING**子句中。不同的**DBMS**系统对聚合函数有不同的扩充。
- ❁ 使用聚合函数之后，**SELECT**子句中所出现的列名或列名表达式必须出现在**GROUP BY**子句中或者为聚合函数表达式。