

1.2 数据模型

1.2.1 两大类数据模型

1.2.2 数据模型的组成要素

1.2.3 概念模型

1.2.4 最常用的数据模型

1.2.5 层次模型

1.2.6 网状模型

1.2.7 关系模型



1.2.6 网状模型

- 在现实世界中事物之间的联系更多的是非层次关系的，用层次模型表示非树形结构不直接，网状模型则可以克服这一弊病。
- 网状数据模型的典型代表是DBTG系统。这是20世纪70年代数据系统语言研究会CODASYL（Conference On Data System Language）下属的数据库任务组（Data Base Task Group，简称DBTG）提出的一个系统方案。
- **DBTG系统虽然不是实际的软件系统，但是它提出的基本概念、方法和技术具有普遍意义。**它对于网状数据库系统的研制和发展起了重大的影响。后来不少的系统都采用DBTG模型或者简化的DBTG模型。例如，Cullient Software公司的IDMS、UniVac公司的DMS1100、Honeywell公司的IDS/2、HP公司的IMAGE等。

网状数据模型的数据结构

- 网状模型

满足下面两个条件的基本层次联系的集合：

1. 允许一个以上的结点无双亲；
2. 一个结点可以有多个的双亲。



网状数据模型的数据结构（续）

- 表示方法(与层次数据模型相同)

实体型：用记录类型描述

每个结点表示一个记录类型（实体）

属性：用字段描述

每个记录类型可包含若干个字段

联系：用结点之间的连线表示记录类型（实体）之间的一对多的父子联系



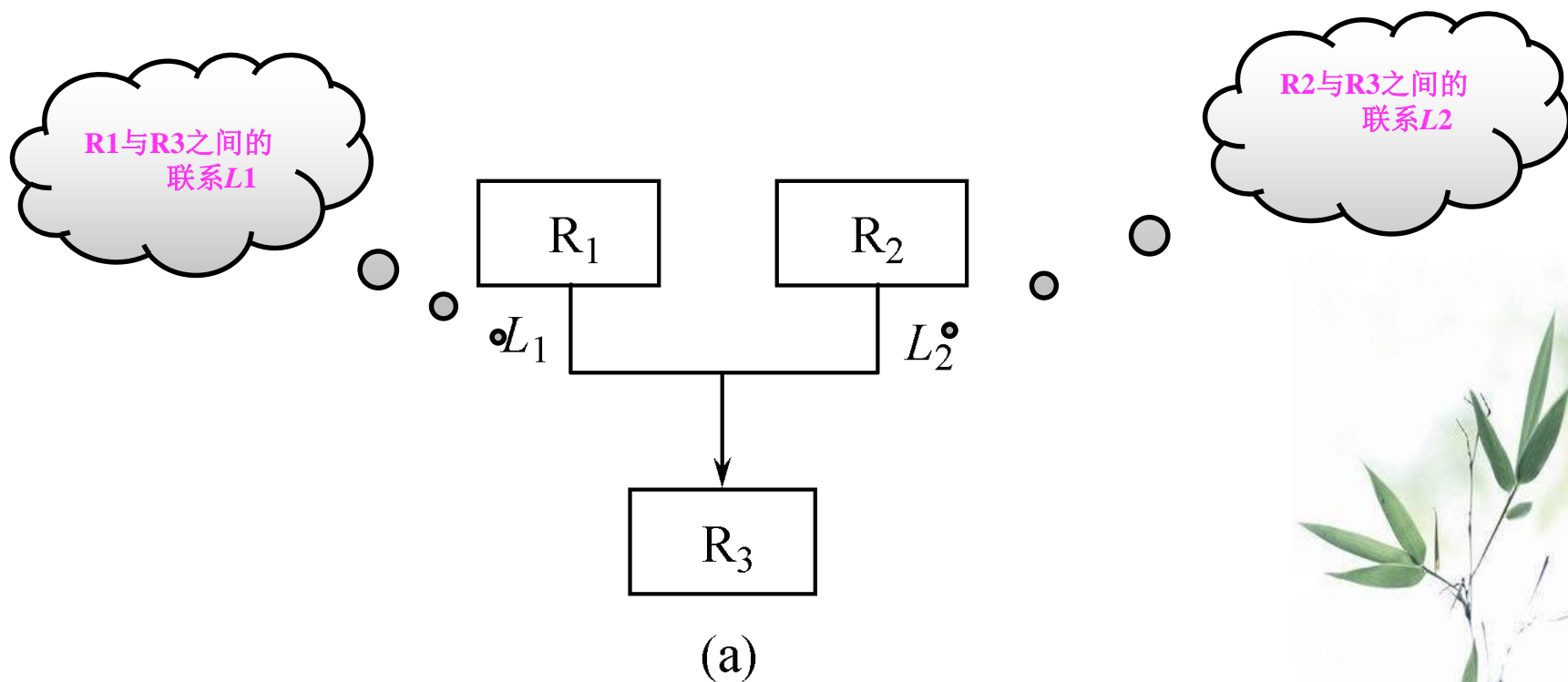
网状数据模型的数据结构（续）

- 网状模型与层次模型的区别
 - 网状模型允许多个结点没有双亲结点
 - 网状模型允许结点有多个双亲结点
 - 网状模型允许两个结点之间有多种联系（复合联系）
 - 网状模型可以更直接地去描述现实世界
 - 层次模型实际上是网状模型的一个特例

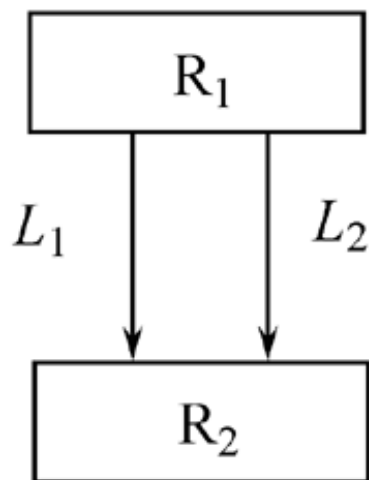


网状数据模型的数据结构（续）

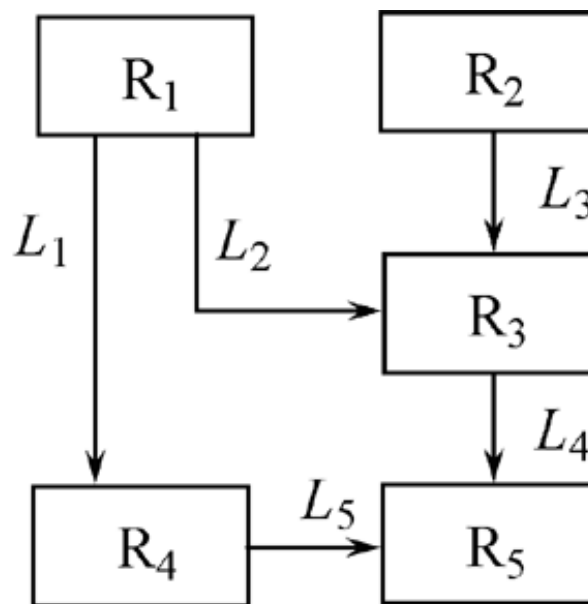
- ❖ 网状模型中子女结点与双亲结点的联系可以不唯一
要为每个联系命名，并指出与该联系有关的双亲记录和子女记录



网状数据模型的数据结构（续）



(b)

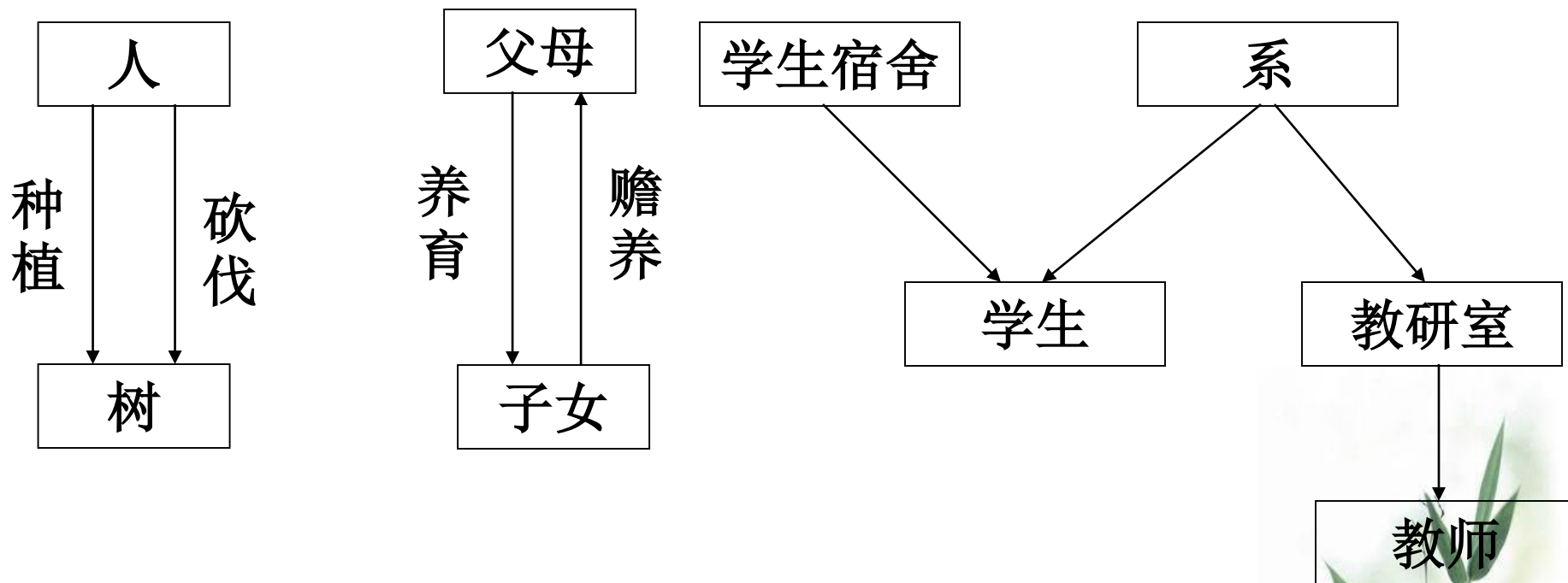


(c)

网状模型的例子



网状数据模型的数据结构



网状数据模型的数据结构（续）

实际的商品化网状数据库系统对网状数据结构都有不同的限制，例如HP公司的IMAGE3000数据库管理系统是一个网状数据库系统，但是限制网状结构的层次只有两层，是一个简化了的网状结构。

多对多联系在网状模型中的表示

- 用网状模型间接表示多对多联系
- 方法：

将多对多联系直接分解成一对多联系



网状数据模型的数据结构（续）

例如：一个学生可以选修若干门课程，某一课程可以被多个学生选修，学生与课程之间是多对多联系

- 引进一个学生选课的联系记录，由3个数据项组成,表示某个学生选修某一门课程及其成绩

- 学号
- 课程号
- 成绩



网状数据模型的数据结构（续）

每个学生可以选修多门课程，对学生记录中的一个值，选课记录中可以有多多个值与之联系，而选课记录中的一个值，只能与学生记录中的一个值联系。学生与选课之间的联系是一对多的联系，联系名为S-SC。

同样，课程与选课之间的联系是一对多的联系，联系名为C-SC。

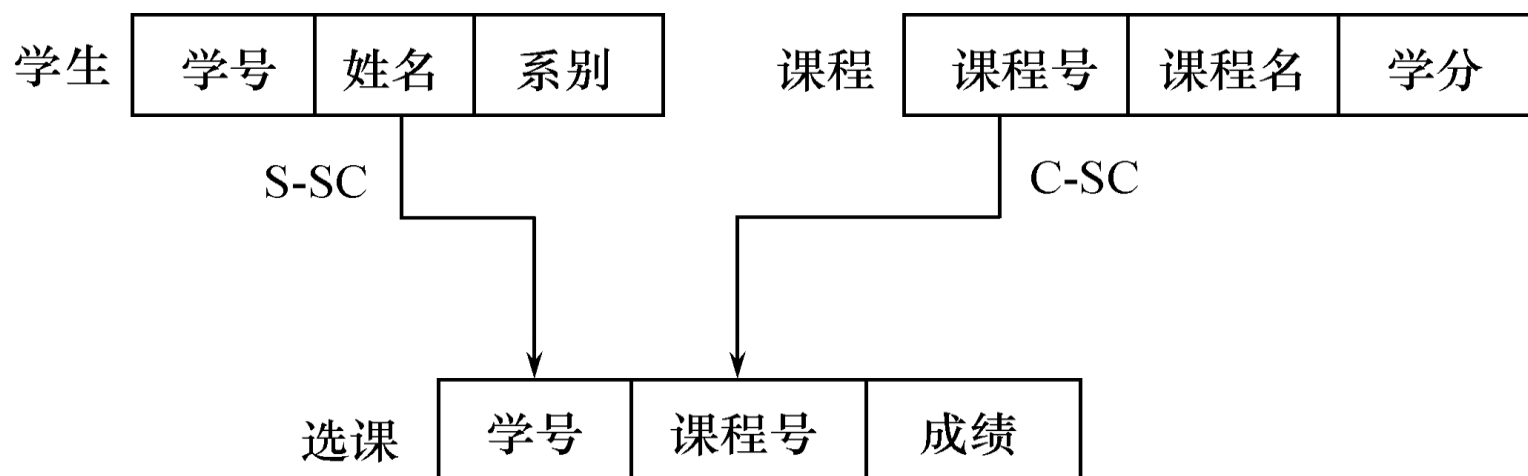


图 学生/选课/课程的网状数据模型

网状数据模型的优缺点

- 优点

- 能够更为直接地描述现实世界，如一个结点可以有多个双亲
- 具有良好的性能，存取效率较高

- 缺点

- 结构比较复杂，而且随着应用环境的扩大，数据库的结构就变得越来越复杂，不利于最终用户掌握
- DDL、DML语言复杂，用户不容易使用
- 由于记录之间的联系是通过存取路径实现的，应用程序在访问数据库时必须选择适当的存取路径，即用户需要了解系统结构的细节，加重了编写应用程序的负担。



1.2.7 关系模型

- 关系数据库系统采用关系模型作为数据的组织方式
- 计算机厂商新推出的数据库管理系统几乎都支持关系模型



一、关系数据模型的数据结构

- 在用户观点下，关系模型中数据的逻辑结构是一张二维表，由行和列组成。

学生登记表

学 号	姓 名	年 龄	性 别	系 名	年 级
2005004	王小明	19	女	社会学	2005
2005006	黄大鹏	20	男	商品学	2005
2005008	张文斌	18	女	法律	2005
...

属性

元组

关系数据模型的数据结构（续）

– 关系（**Relation**）

一个关系对应通常说的一张表

– 元组（**Tuple**）

表中的一行即为一个元组

– 属性（**Attribute**）

表中的一列即为一个属性，给每一个属性起一个名称即属性名



关系数据模型的数据结构（续）

- 主码 (**Key**)

表中的某个属性组，它可以唯一确定一个元组。

- 域 (**Domain**)

属性的取值范围。

- 分量

元组中的一个属性值。

- 关系模式

对关系的描述

关系名（属性1，属性2，...，属性n）

学生（学号，姓名，年龄，性别，系，年级）



关系数据模型的数据结构（续）

- 关系必须是规范化的，满足一定的规范条件

最基本的规范条件：关系的每一个分量必须是一个不可分的数据项，
不允许表中还有表

图中工资和扣除是可分的数据项，不符合关系模型要求

职工号	姓名	职 称	工 资			扣 除		实 发
			基 本	津 贴	职 务	房 租	水 电	
86051	陈 平	讲 师	1305	1200	50	160	112	2283

图 一个工资表(表中有表)实例

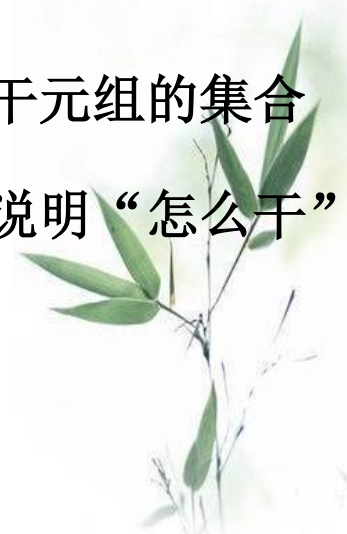
关系数据模型的数据结构（续）

术语对比

关系术语	一般表格的术语
关系名	表名
关系模式	表头（表格的描述）
关系	（一张）二维表
元组	记录或行
属性	列
属性名	列名
属性值	列值
分量	一条记录中的一个列值
非规范关系	表中有表（大表中嵌有小表）

二、关系数据模型的操纵与完整性约束

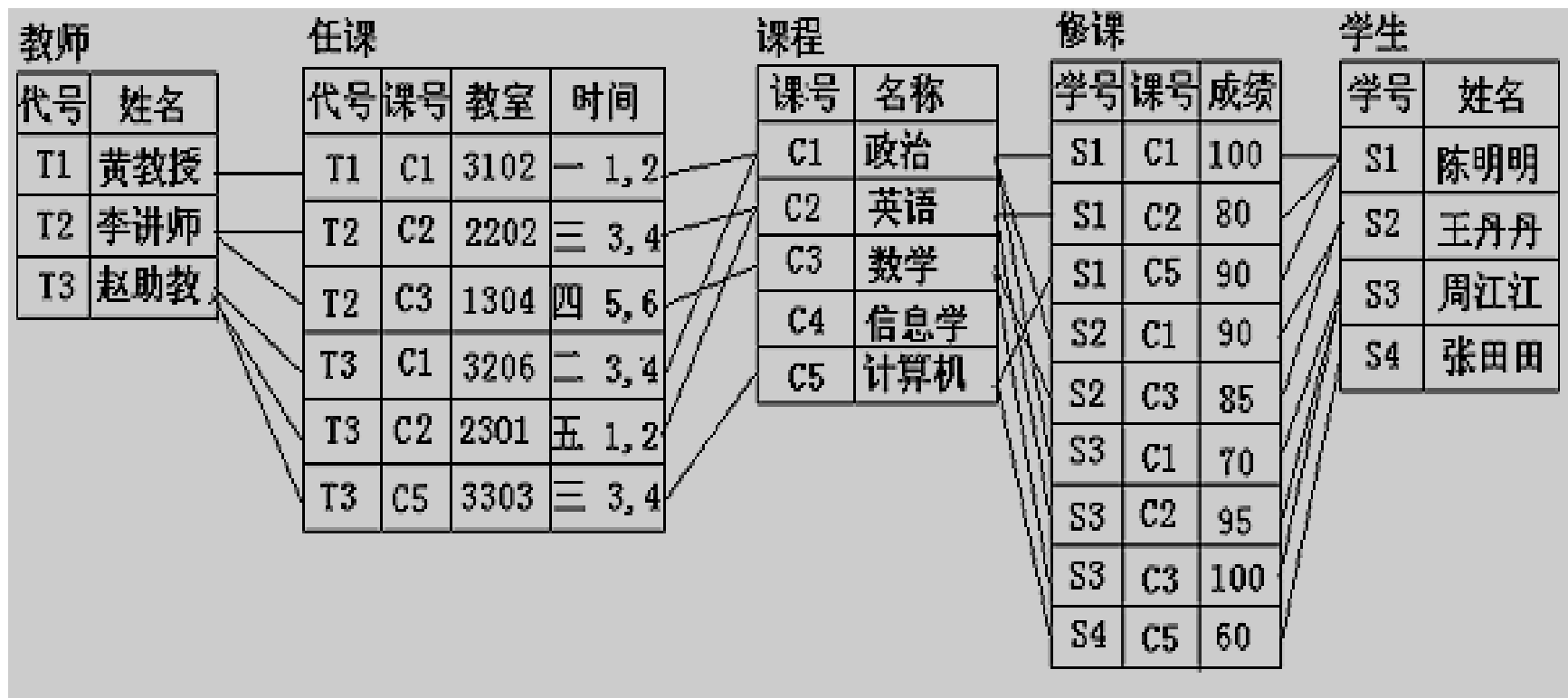
- 数据操作是集合操作，操作对象和操作结果都是关系
 - 查询
 - 插入
 - 删除
 - 更新
- 数据操作是集合操作，操作对象和操作结果都是关系，即若干元组的集合
- 存取路径对用户隐蔽，用户只要指出“干什么”，不必详细说明“怎么干”



关系数据模型的操纵与完整性约束（续）

- 关系的完整性约束条件
 - 实体完整性
 - 参照完整性
 - 用户定义的完整性





三、关系数据模型的存储结构

- 实体及实体间的联系都用表来表示
- 表以文件形式存储
 - 有的DBMS一个表对应一个操作系统文件
 - 有的DBMS自己设计文件结构



四、关系数据模型的优缺点

- 优点

- 建立在严格的数学概念的基础上

- 概念单一

- 实体和各类联系都用关系来表示

- 对数据的检索结果也是关系

- 关系模型的存取路径对用户透明

- 具有更高的数据独立性，更好的安全保密性

- 简化了程序员的工作和数据库开发建立的工作



关系数据模型的优缺点（续）

- 缺点

- 存取路径对用户透明导致查询效率往往不如非关系数据模型
- 为提高性能，必须对用户的查询请求进行优化增加了开发DBMS的难度



常用数据模型的比较

	层次模型	网状模型	关系模型	面向对象模型
创始	1968 IBM IMS系统	1969 CODASYL DBTG报告	1970 E.F.Codd 提出	20世纪80年代
数据结构	复杂 (树结构)	复杂 (有向图结构)	简单 (二维表)	复杂 (嵌套, 递归)
数据联系	通过指针	通过指针	通过表间的公 共属性	面向对象标识
查询语言	过程性语言	过程性语言	非过程性语言	面向对象语言
典型产品	IMS	IDS/II, IMAGE/ 3000, IDMS	Oracle, Sybase, DB2, SQL Server	ONTOS DB
盛行期	20世纪70年 代	20世纪70年代 到80年代中期	20世纪80年代 到现在	20世纪90年代到现 在

第一章 绪论

1.1 数据库系统概述

1.2 数据模型

1.3 数据库系统结构

1.4 数据库系统的组成

1.5 小结



1.3 数据库系统结构

- 从数据库**管理系统角度**看，数据库系统通常采用三级模式结构，是数据库系统内部的系统结构
- 从数据库**最终用户角度**看（数据库系统外部的体系结构）数据库系统的结构分为：
 - 单用户结构
 - 主从式结构
 - 分布式结构
 - 客户 / 服务器
 - 浏览器 / 应用服务器 / 数据库服务器多层结构等



数据库系统结构（续）

1.3.1 数据库系统模式的概念

1.3.2 数据库系统的三级模式结构

1.3.3 数据库的二级映像功能与数据独立性



1.3.1 数据库系统模式的概念

- “型” 和 “值” 的概念

- 型(Type)

- 对某一类数据的结构和属性的说明

- 值(Value)

- 是型的一个具体赋值

例如

学生记录型:

(学号, 姓名, 性别, 系别, 年龄, 籍贯)

一个记录值:

(900201, 李明, 男, 计算机, 22, 江苏)



数据库系统模式的概念（续）

- 模式（**Schema**）
 - 数据库逻辑结构和特征的描述
 - 是型的描述
 - 反映的是数据的结构及其联系
 - 模式是相对稳定的
- 实例（**Instance**）
 - 模式的一个具体值
 - 反映数据库某一时刻的状态
 - 同一个模式可以有很多实例
 - 实例随数据库中的数据更新而变动



模式和实例

两个实例



模式



- 学生表 (学号, 姓名, 年龄)
- 课程表 (课程号, 课程名, 学分)
- 选课表 (学号, 课程号, 成绩)

实际中的模式描述
比本例要详细得多

S001	张三	21
S002	李四	20

C001	数据库	4
C002	英语	6
C003	数学	6

S001	C001	90
S002	C001	80

S001	张三	21
S002	李四	20
S003	王五	22

C001	数据库	4
C002	英语	6
C003	数学	6

S001	C001	90
S002	C001	80
S003	C001	90
S003	C002	96
S003	C003	98

数据库系统结构（续）

1.3.1 数据库系统模式的概念

1.3.2 数据库系统的三级模式结构

1.3.3 数据库的二级映像功能与数据独立性



数据库管理系统中的数据抽象

- **数据库管理系统**的一个主要作用就是隐藏关于数据存储和维护的某些细节，为用户提供数据在不同层次上的抽象视图，这就是数据抽象。
- **数据库管理系统**通过如下三个层次的抽象来向用户屏蔽复杂性，简化系统的用户界面。

1、物理层抽象

2、逻辑层抽象

3、视图层（概念层）抽象



数据库管理系统中的数据抽象（续）

- 1、物理层抽象

最低层次的抽象，描述数据实际上是如何存储的。
（内模式）

- 2、逻辑层抽象

比物理层稍高层次的抽象，描述数据库中存储什么数据以及这些数据间存在的关系。（模式）



- 3、视图层（概念层）抽象

最高层次的抽象，但只描述整个数据库的某个部分。数据库系统的多数用户并不需要关心所有的信息，而只需要访问数据库的一部分。视图抽象层的定义正是为了使用户与系统的交互更简单。系统可以为同一数据库提供多个视图，而视图又保证了数据的安全性。（外模式）

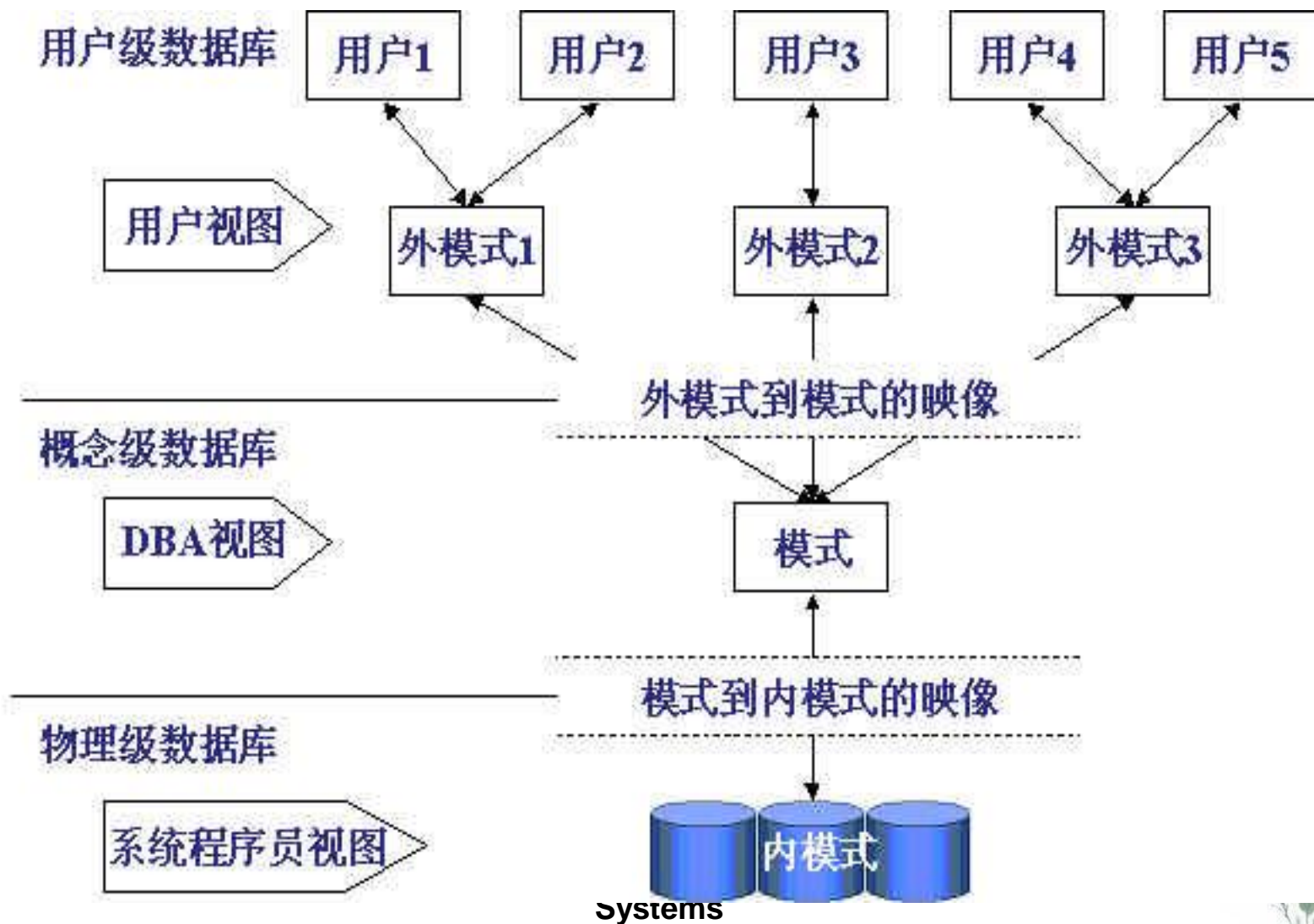


1.3.2 数据库系统的三级模式结构

- 模式（Schema）
- 外模式（External Schema）
- 内模式（Internal Schema）



数据库的三级模式二级映象

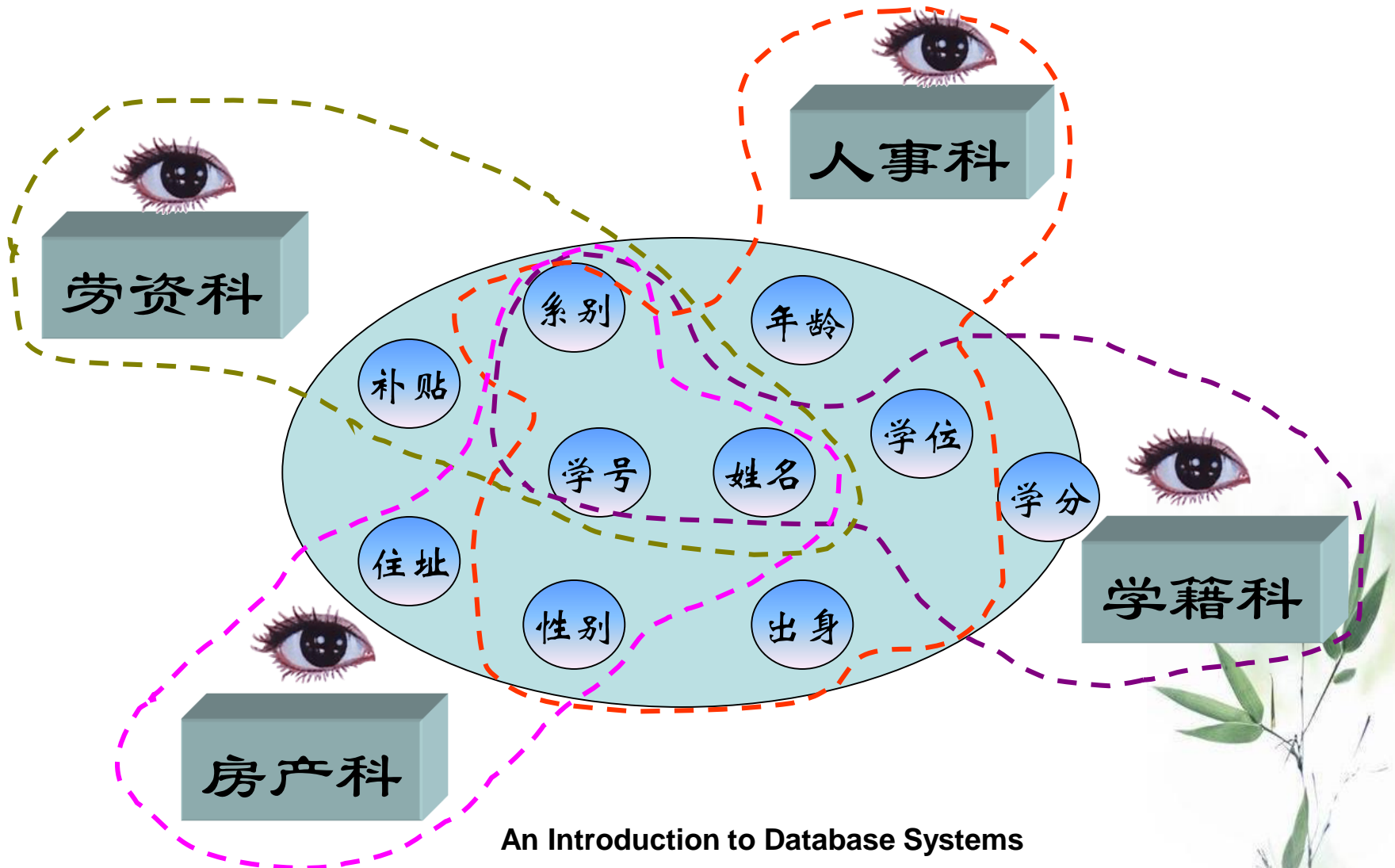


一、模式（Schema）

- 模式（也称逻辑模式）
 - 数据库中全体数据的逻辑结构和特征的描述
 - 所有用户的公共数据视图，综合了所有用户的需求
- 一个数据库只有一个模式
- 模式的地位：是数据库系统模式结构的中间层
 - 与数据的物理存储细节和硬件环境无关
 - 与具体的应用程序、开发工具及高级程序设计语言无关



—数据库系统阶段



模式（续）

- 模式的定义
 - 数据的逻辑结构（数据项的名字、类型、取值范围等）
 - 数据之间的联系
 - 数据有关的安全性、完整性要求



➤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	

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

- Create table sc

```
{ sno char(4)
  cno char(4)
  grade int
  primary key(sno,cno)
  foreign key (sno) reference student(sno)
  check ( 0=<grade<=100)
  ....
}
```



二、外模式（External Schema）

- 外模式（也称子模式或用户模式）
 - 数据库用户（包括应用程序员和最终用户）使用的局部数据的逻辑结构和特征的描述
 - 数据库用户的数据视图，是与某一应用有关的数据的逻辑表示



外模式（续）

- 外模式的地位：介于模式与应用之间
 - 模式与外模式的关系：一对多
 - 外模式通常是模式的子集
 - 一个数据库可以有多个外模式。反映了不同的用户的应用需求、看待数据的方式、对数据保密的要求
 - 外模式与应用的关系：一对多
 - 同一外模式也可以为某一用户的多个应用系统所使用
 - 但一个应用程序只能使用一个外模式



外模式（续）

- 外模式的用途
 - 保证数据库安全性的一个有力措施
 - 每个用户只能看见和访问所对应的外模式中的数据



示例1——子集

学生

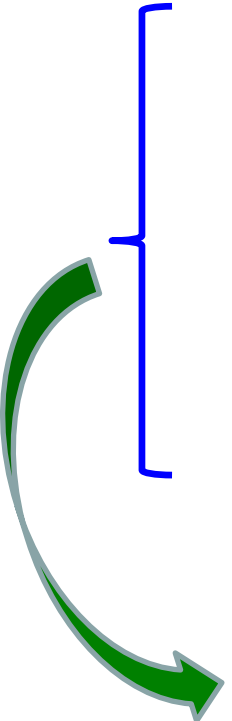
学 号	姓 名	年 龄	性 别	所 在 系
0611101	李勇	21	男	计算机系
0611102	刘晨	20	男	计算机系
0611103	王敏	20	女	计算机系
0621101	张立	20	男	信息管理系
0621102	吴宾	19	女	信息管理系

模式

学 号	姓 名	性 别
0611101	李勇	男
0611102	刘晨	男
0611103	王敏	女
0621101	张立	男
0621102	吴宾	女

外模式

示例2——重构

- 
- 学生（学号，**姓名**，性别，年龄，所在系）
 - 课程（课程号，**课程名**，学分）
 - 选课（学号，课程号，**成绩**）

学生（姓名，课程名，成绩）

示例3——安全性

职工表（职工号，姓名，所在部门，基本工资，职务工资，~~奖励工资~~）



职工信息（职工号，姓名，所在部门，
基本工资，职务工资）



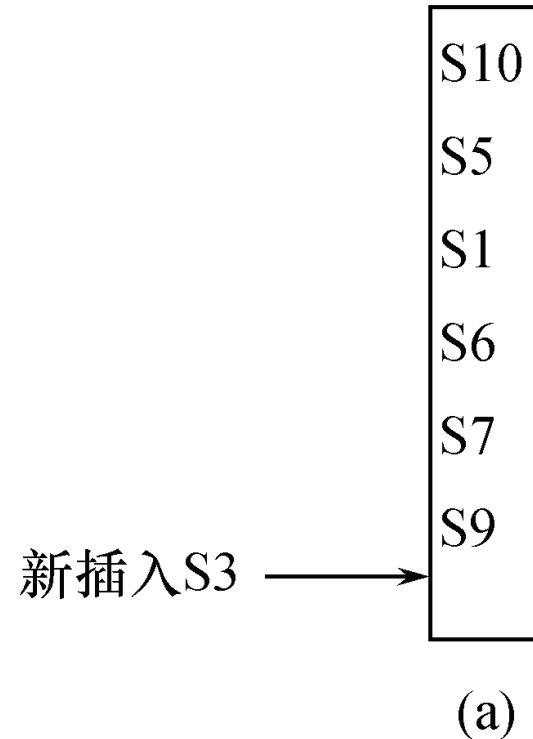
三、内模式（Internal Schema）

- 内模式（也称存储模式）
 - 是数据物理结构和存储方式的描述
 - 是数据在数据库内部的表示方式
 - 记录的存储方式（顺序存储，按照B+树结构存储，按hash方法存储）
 - 索引的组织方式
 - 数据是否压缩存储
 - 数据是否加密
 - 数据存储记录结构的规定
- 一个数据库只有一个内模式



内模式（续）

- 例如学生记录，如果按堆存储，则插入一条新记录总是放在学生记录存储的最后，如右图所示



内模式（续）

- 如果按学号升序存储，则插入一条记录就要找到它应在的位置插入，如图（b）所示
- 如果按照学生年龄聚簇存放，假如新插入的S3是16岁，则应插入的位置如图（c）所示

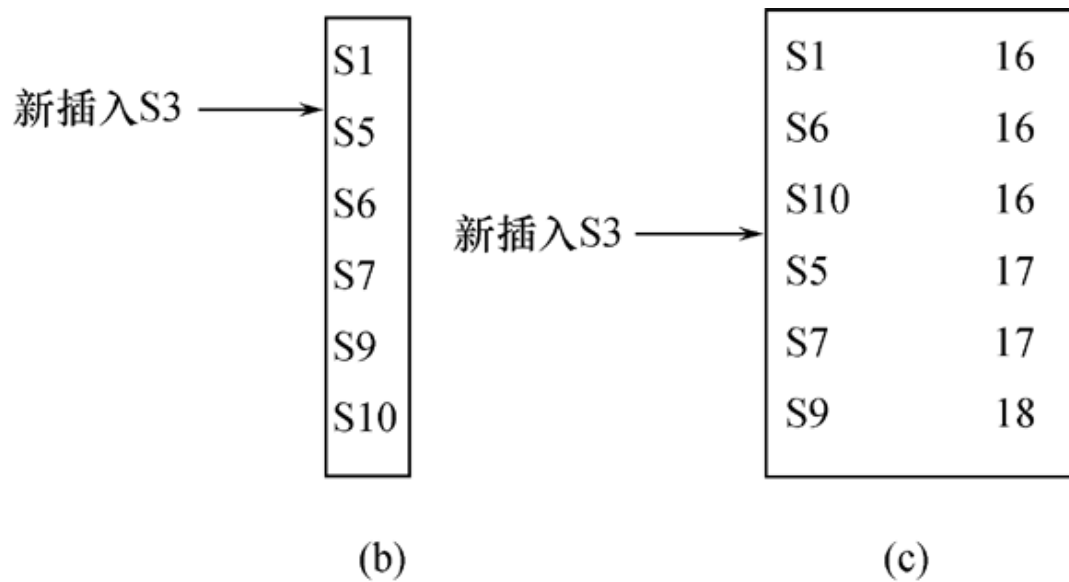
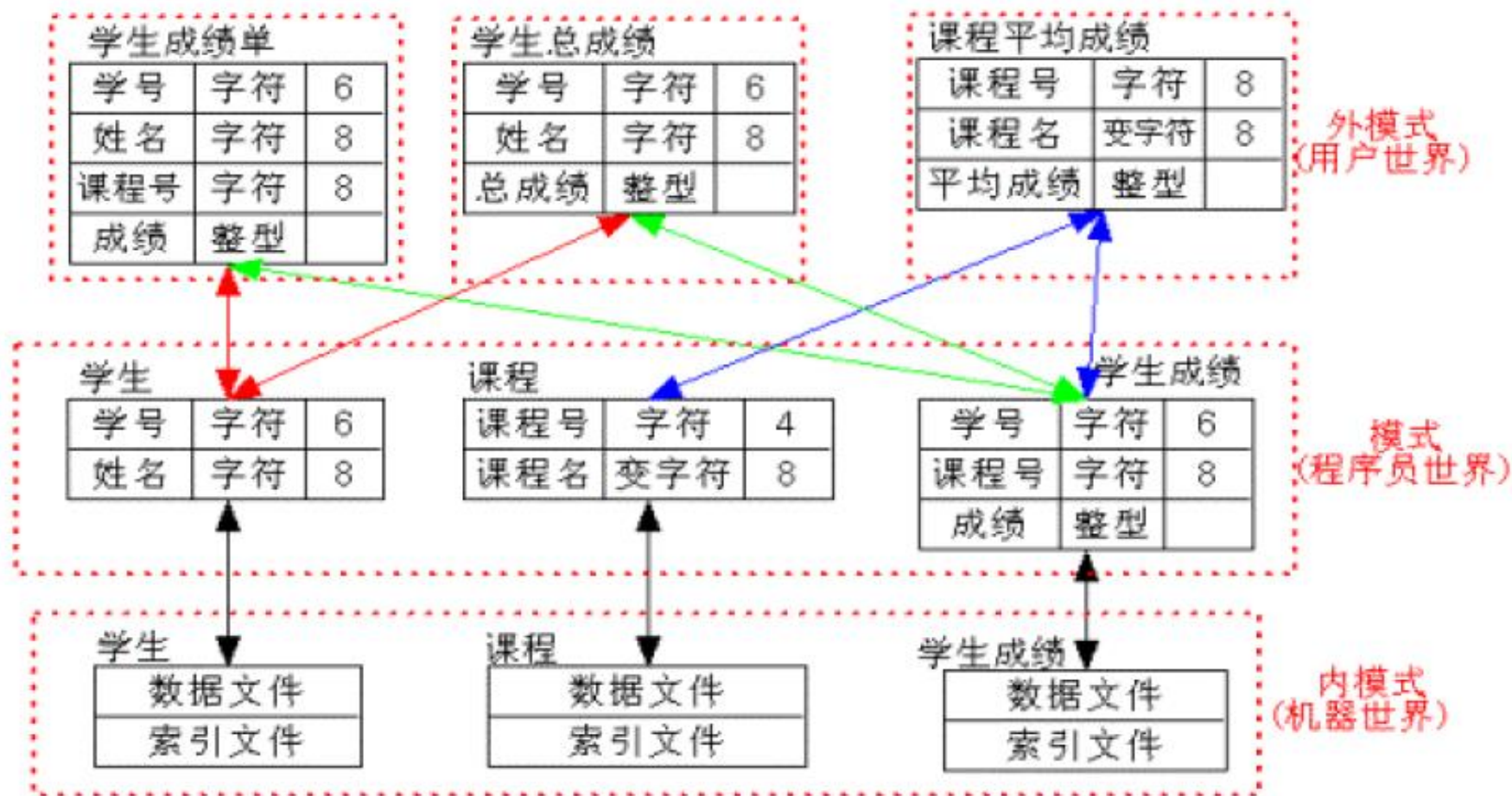


图 记录不同的存储方式示意



外模式/模式/内模式示例



数据库系统结构（续）

1.3.1 数据库系统模式的概念

1.3.2 数据库系统的三级模式结构

1.3.3 数据库的二级映像功能与数据独立性



1.3.3 数据库的二级映像功能与数据独立性

- 三级模式是对数据的三个抽象级别,它把数据的具体组织留给**DBMS**管理,使用户能逻辑地抽象地处理数据,而不必关心数据在计算机中的具体表示方式与存储方式.
- 二级映象在**DBMS**内部实现这三个抽象层次的联系和转换
 - 外模式 / 模式映像
 - 模式 / 内模式映像

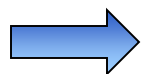


一、外模式 / 模式映象

- 模式：描述的是数据的全局逻辑结构
- 外模式：描述的是数据的局部逻辑结构
- 同一个模式可以有任意多个外模式
- 每一个外模式，数据库系统都有一个外模式 / 模式映象，定义外模式与模式之间的对应关系
- 映象定义通常包含在各自外模式的描述中



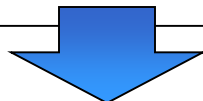
XH	XM	NL	XB
301	赵	21	男
302	钱	22	女



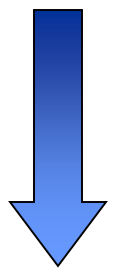
```

Select xh as 学号,
      xm as 姓名,
      xb as 性别,
      nl as 年龄,
From student

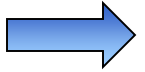
```



学号	姓名	性别	年龄
301	赵	男	21
302	钱	女	22



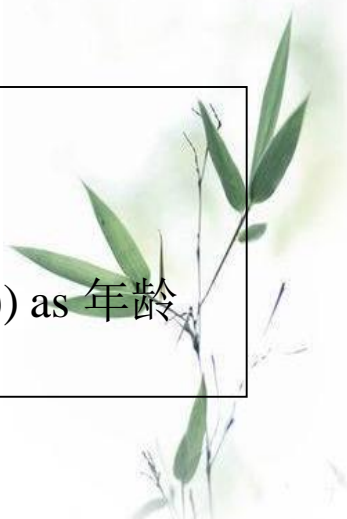
NO	XM	XB	CSRQ
301	赵	男	85.07.01
302	钱	女	84.03.07



```

Select No as 学号,
      xm as 姓名,
      xb as 性别,
      datediff(year,csrq,getdate()) as 年龄,
From student

```



外模式 / 模式映像（续）

保证数据的逻辑独立性

- 当模式改变时，数据库管理员修改有关的外模式 / 模式映像，使外模式保持不变
- 应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性



二、模式 / 内模式映象

- 模式 / 内模式映象定义了数据全局逻辑结构与存储结构之间的对应关系。
 - 例如，说明逻辑记录和字段在内部是如何表示的
- 数据库中模式 / 内模式映象是唯一的



模式 / 内模式映象（续）

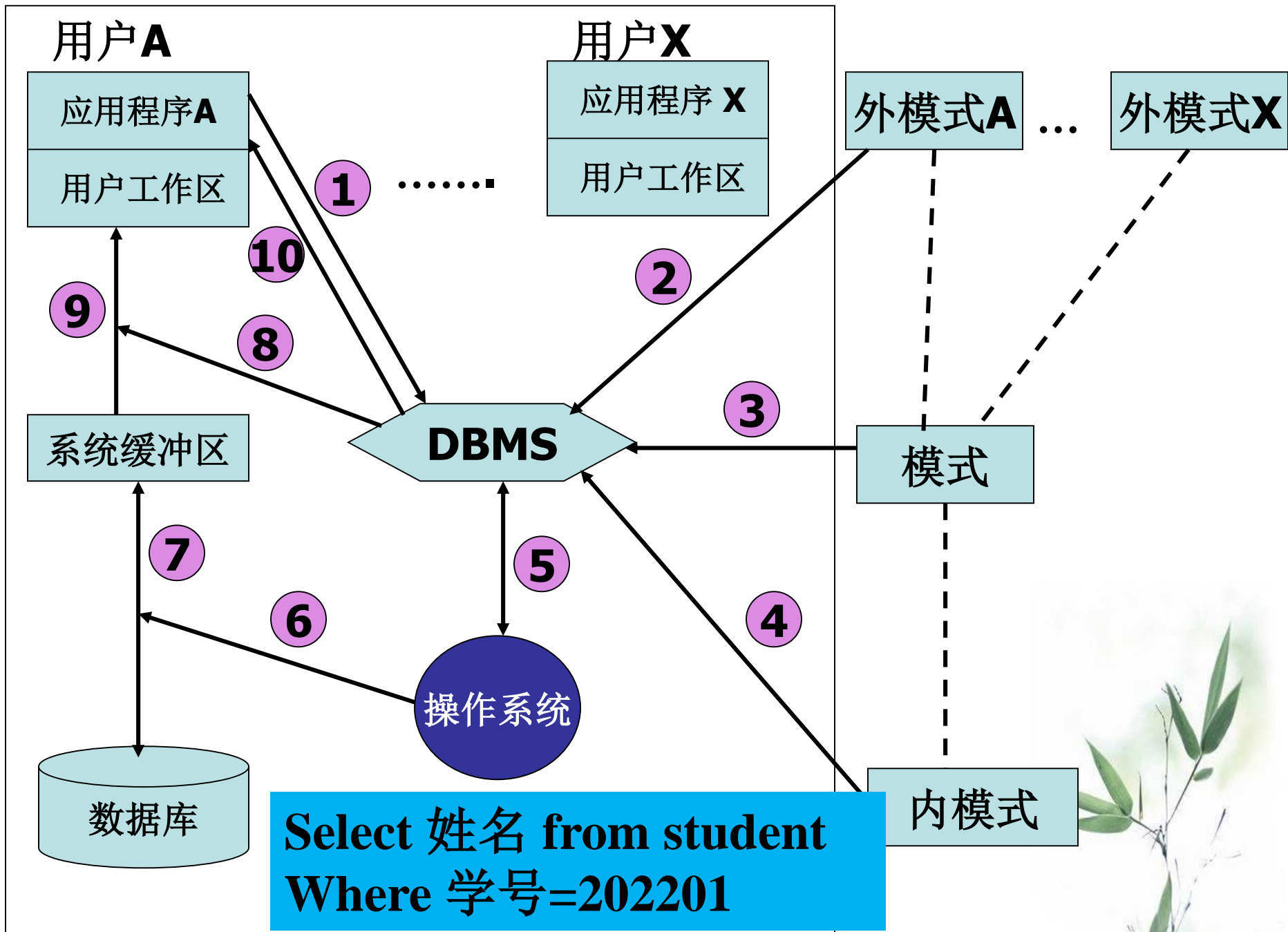
保证数据的物理独立性

- 当数据库的存储结构改变了（例如选用了另一种存储结构），数据库管理员修改模式 / 内模式映象，使模式保持不变
- 应用程序不受影响。保证了数据与程序的物理独立性，简称数据的物理独立性



- 数据与程序之间的独立性，使得数据的定义和描述可以从应用程序中分离出去
- 数据的存取由**DBMS**管理
 - 用户不必考虑存取路径等细节
 - 简化了应用程序的编制
 - 大大减少了应用程序的维护和修改





用户访问数据的过程

- 应用程序A通过DBMS读取数据库中记录的全过程
 - 1) 用户在应用程序A中安排一条读记录的DML语句
 - 该语句给出涉及的外模式中记录类型名
 - 执行该语句时，立即启动DBMS，并把读记录的命令传给DBMS
 - 2) DBMS检查读操作的合法性
 - 对读命令加以分析
 - 从DD中调出与程序A对应的外模式
 - 检查该操作是否合法，决定是否执行读命令



用户访问数据的过程

3) 决定执行A的命令，DBMS对模式操作

- 调出相应的模式
- 执行外模式/模式映象功能
 - 把外模式的外部记录格式映象成模式的记录格式
- 决定模式应读哪些记录

4) DBMS对内模式操作

- 调出相应的内模式
- 执行模式/内模式的映象功能，把模式记录格式映象成内模式的内部记录格式
- 确定应读入哪些物理记录以及相应的地址信息



用户访问数据的过程

- 5) **DBMS**向操作系统**OS**发出从指定地址读取物理记录的命令
- 6) **OS**执行读命令
 - 按指定地址从数据库中把记录读入**OS**的系统缓冲区
 - 随即读入数据库的系统缓冲区
 - 并在操作结束后向**DBMS**作出回答
- 7) **DBMS**收到**OS**读操作结束的回答后，将读入缓冲区中的数据转换成模式记录、外部记录

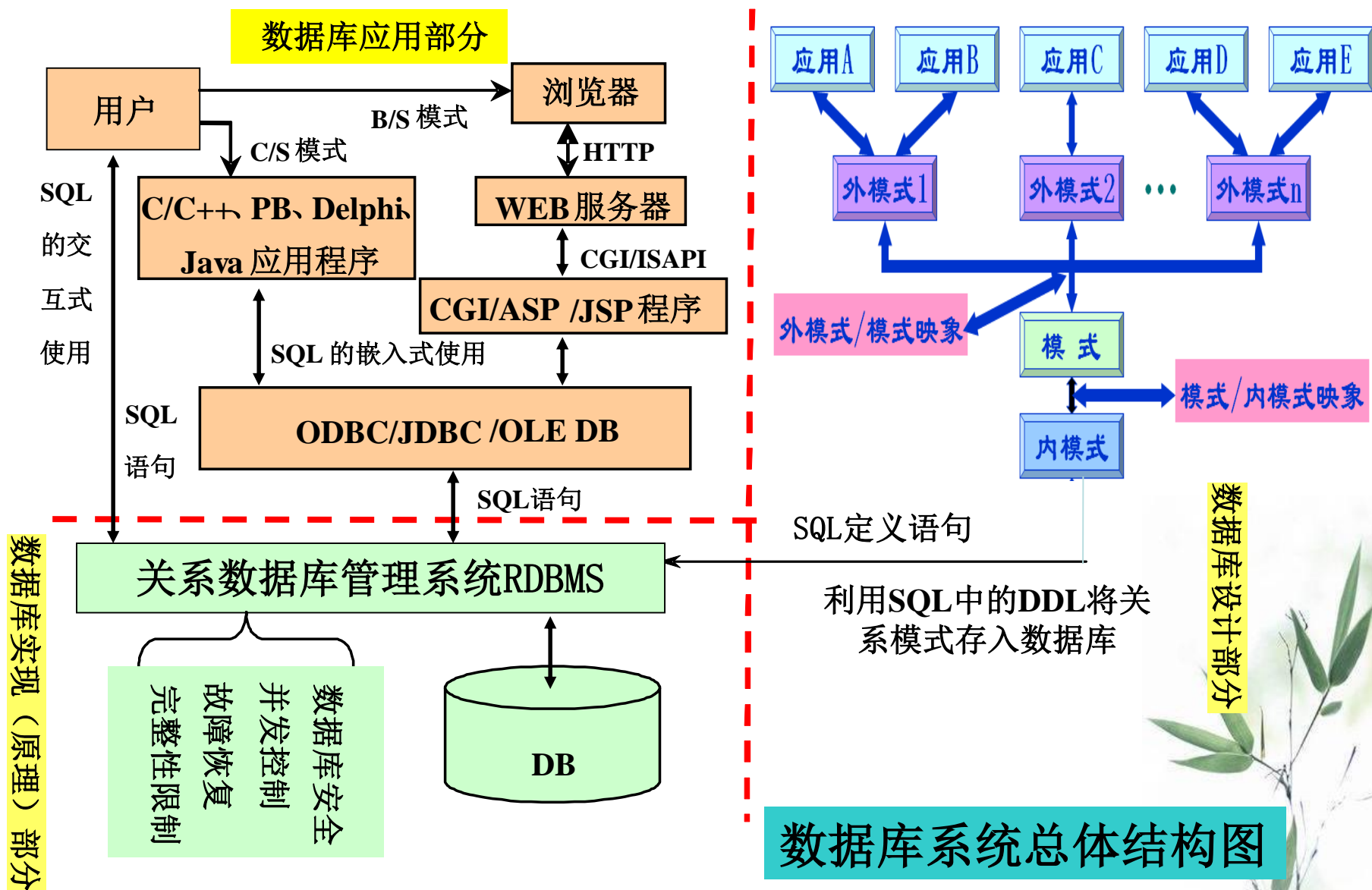


用户访问数据的过程

- 8) **DBMS**把导出的外部记录从系统缓冲区送到应用程序A的变量中
- 9) **DBMS**向运行日志数据库写入读一条记录的信息，以备以后查阅数据库的使用情况
- 10) **DBMS**将读记录操作的成功与否信息返回给应用程序A



数据库设计、原理与应用之间的联系



第一章 绪论

1.1 数据库系统概述

1.2 数据模型

1.3 数据库系统结构

1.4 数据库系统的组成

1.5 小结



1.4 数据库系统的组成

- 硬件平台及数据库
- 软件
- 人员



一、硬件平台及数据库

- 数据库系统对硬件资源的要求

(1) 足够大的内存

- 操作系统
- **DBMS**的核心模块
- 数据缓冲区
- 应用程序



硬件平台及数据库（续）

(2) 足够大的外存

- 磁盘或磁盘阵列

 - 数据库

- 光盘、磁带

 - 数据备份

(3) 较高的通道能力，提高数据传送率



二、软件

- DBMS
- 支持DBMS运行的操作系统
- 与数据库接口的高级语言及其编译系统
- 以DBMS为核心的应用开发工具
- 为特定应用环境开发的数据库应用系统

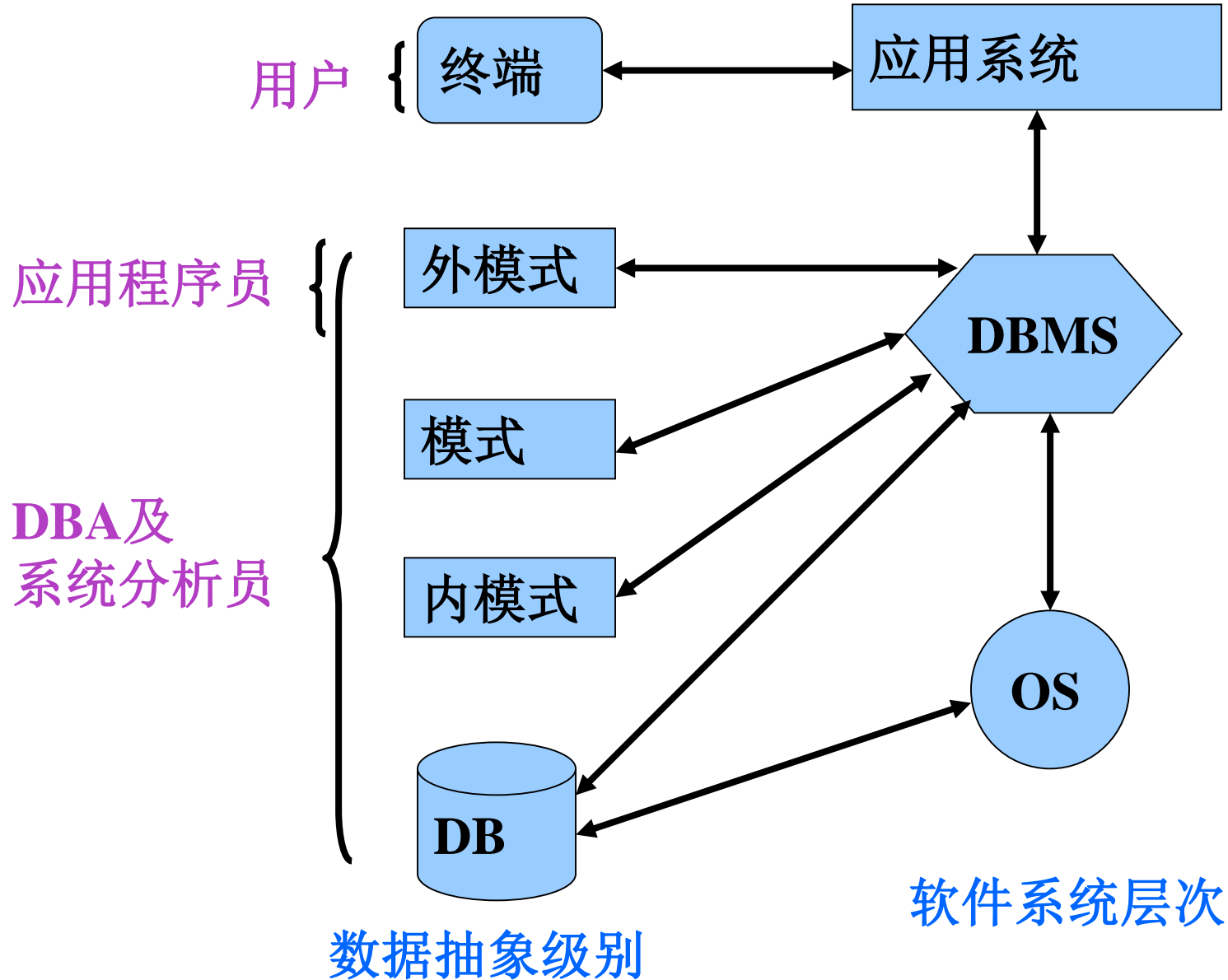


三、人员

- 数据库管理员
- 系统分析员和数据库设计人员
- 应用程序员
- 用户



各种人员的数据视图



1. 数据库管理员(DBA)

具体职责

- 1.决定数据库中的信息内容和结构
- 2.决定数据库的存储结构和存取策略
- 3.定义数据的安全性要求和完整性约束条件



数据库管理员(续)

- 4. 监控数据库的使用和运行
 - 周期性转储数据库
 - 数据文件
 - 日志文件
 - 系统故障恢复
 - 介质故障恢复
 - 监视审计文件



数据库管理员(续)

- 5. 数据库的改进和重组
 - 性能监控和调优
 - 定期对数据库进行重组，以提高系统的性能
 - 需求增加和改变时，数据库须需要重构造



2. 系统分析员和数据库设计人员

- 系统分析员

- 负责应用系统的需求分析和规范说明
- 与用户及DBA协商，确定系统的软硬件配置
- 参与数据库系统的概要设计

- 数据库设计人员

- 参加用户需求调查和系统分析
- 确定数据库中的数据
- 设计数据库各级模式



系统分析员和数据库设计人员（续）

- 数据库设计人员
 - 参加用户需求调查和系统分析
 - 确定数据库中的数据
 - 设计数据库各级模式



3. 应用程序员

- 设计和编写应用系统的程序模块
- 进行调试和安装



4. 用户

用户是指最终用户（**End User**）。最终用户通过应用系统的用户接口使用数据库。

- 1. 偶然用户
 - 不经常访问数据库，但每次访问数据库时往往需要不同的数据库信息
 - 企业或组织机构的高中级管理人员



用户（续）

- 2. 简单用户
 - 主要工作是查询和更新数据库
 - 银行的职员、机票预定人员、旅馆总台服务员
- 3. 复杂用户
 - 工程师、科学家、经济学家、科技工作者等
 - 直接使用数据库语言访问数据库，甚至能够基于数据库管理系统的API编制自己的应用程序

