



2025年春季学期

51

数据库系统概论

An Introduction to Database Systems

第一章 绪论

中国科学技术大学
人工智能与数据科学学院

黄振亚, huangzhy@ustc.edu.cn



1.2 数据模型

114

1.2.1 数据建模

1.2.2 概念模型

1.2.3 数据模型组成的三要素

1.2.4 层次模型

1.2.5 网状模型

1.2.6 关系模型

1.2.7* 数据库领域中不断涌现的数据模型



1.2.4 层次模型

115

- 层次模型是数据库系统中最早出现的数据模型
- 层次数据库系统的典型代表是IBM公司的IMS
(Information Management System) 数据库管理系统
- 层次模型用**树形结构**来表示各类实体以及实体间的联系



层次数据模型的数据结构(续)

116

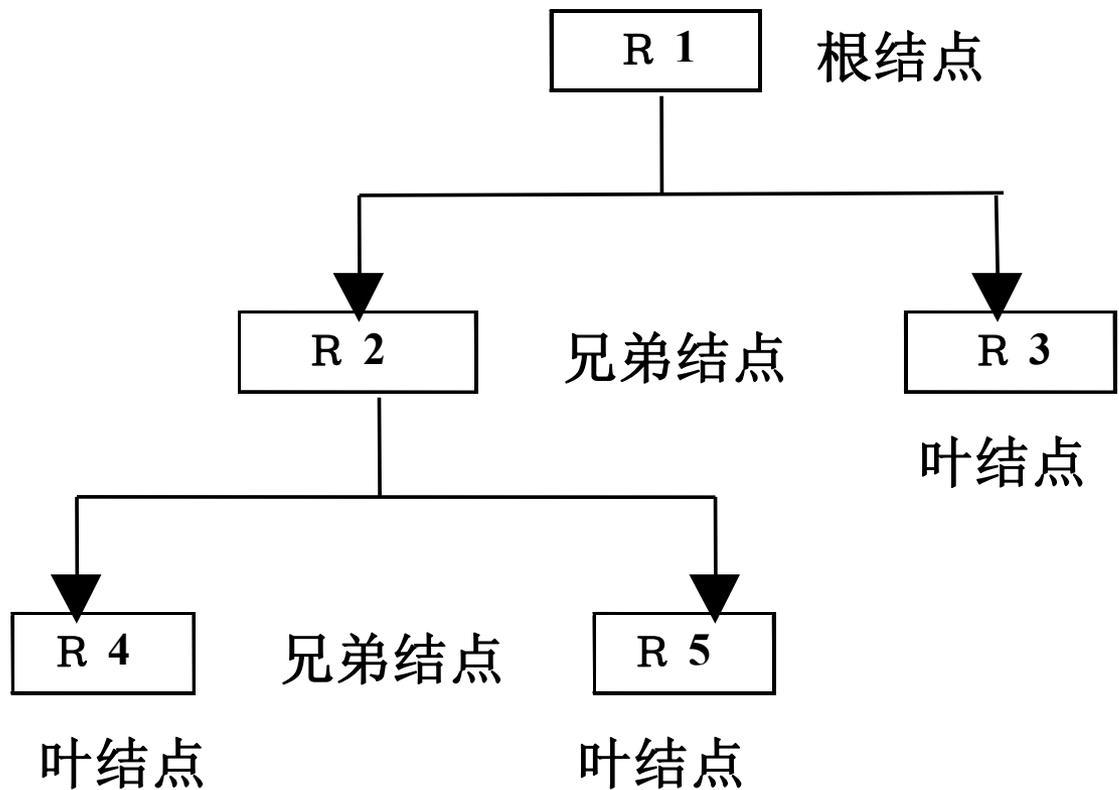


图1.10 一个层次模型的示例



层次数据模型的数据结构

117

□ 表示方法

□ 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



一、层次数据模型的数据结构

□ 基本层次联系：**两个实体集之间的一对多(含一对一)联系**

□ 层次模型

层次模型的完整性约束条件

满足下面两个条件的基本层次联系的集合为层次模型

1. 有且只有一个结点没有双亲结点，这个结点称为根结点

2. 根以外的其它结点有且只有一个双亲结点

□ 层次模型中的几个术语

□ 根结点，双亲结点，兄弟结点，叶结点

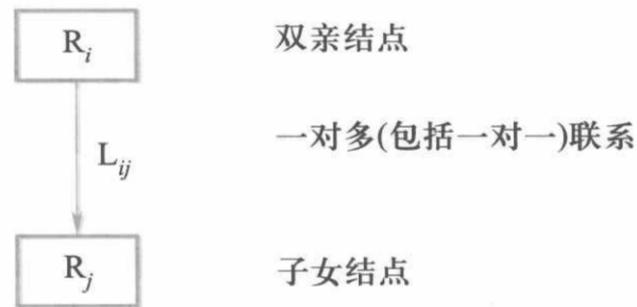
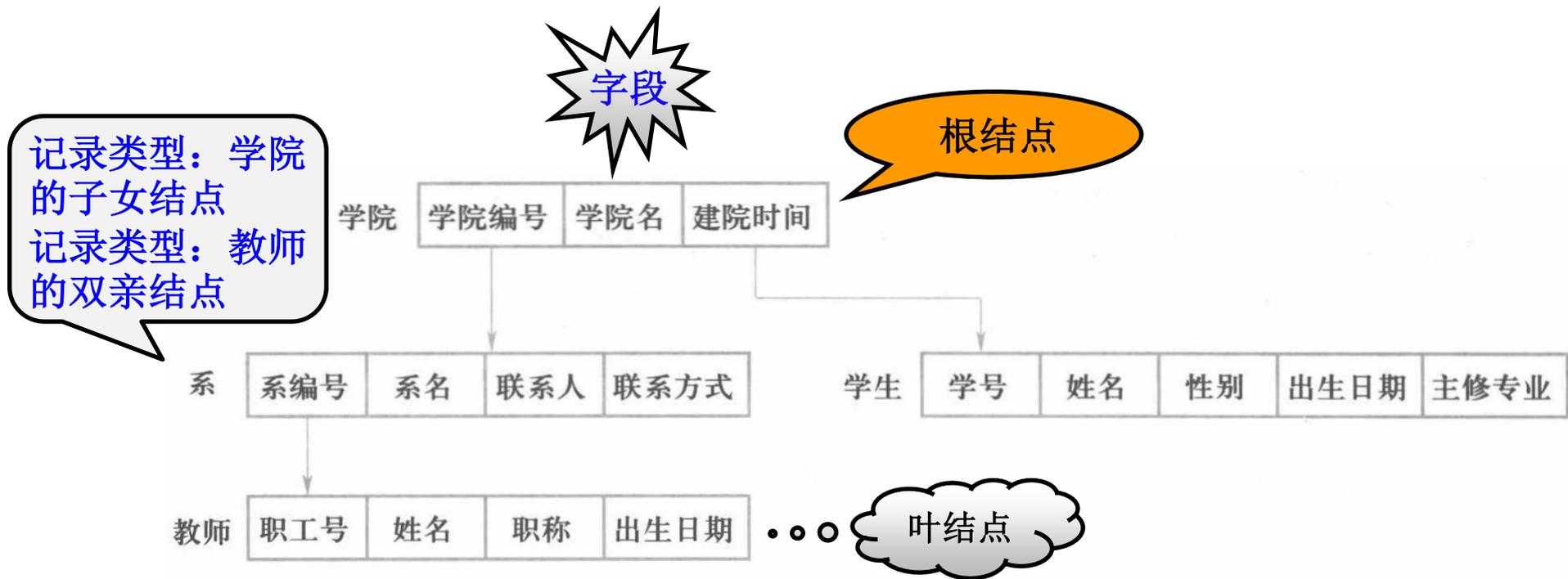


图 1.9 基本层次联系



层次数据模型的数据结构(续)

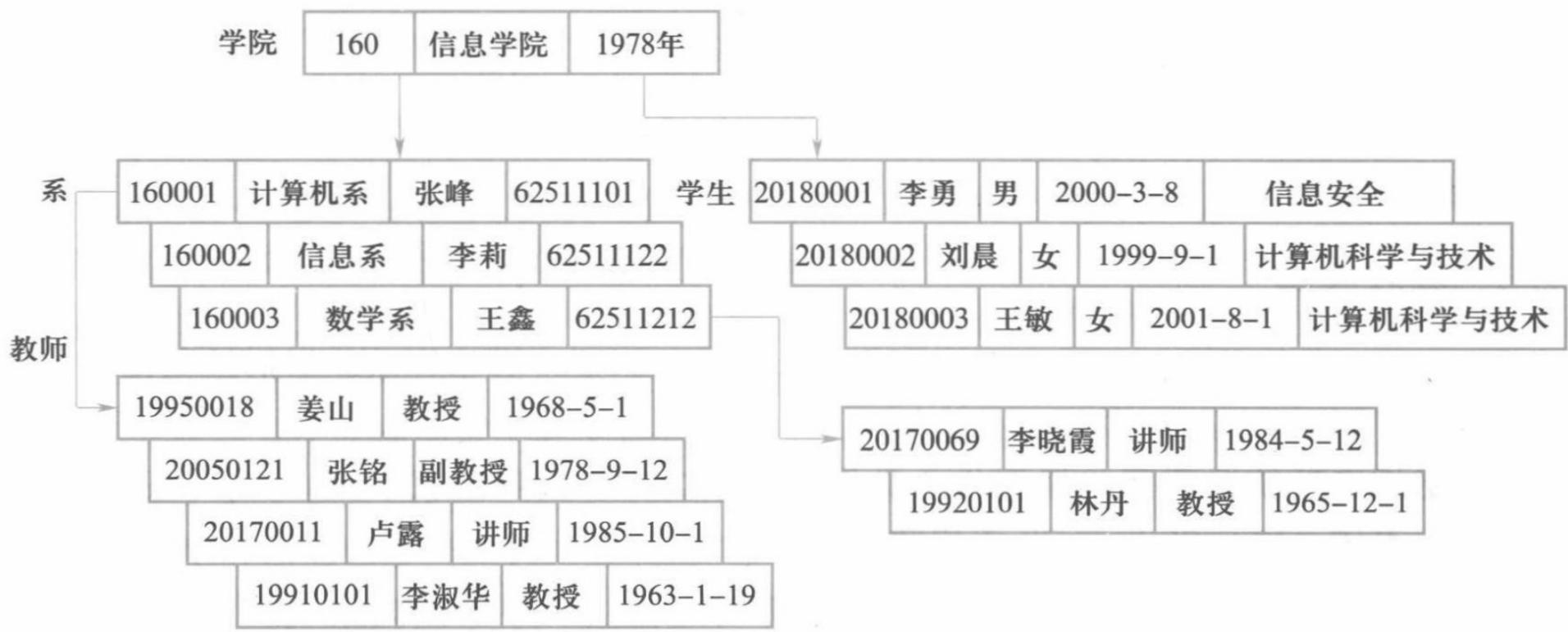
图1.11 “学生学籍管理”子系统的层次模型示意图





层次数据模型的数据结构(续)

图1.12 “学生学籍管理” 子系统层次模型的一个值





层次数据模型的数据结构(续)

121

- 层次模型的特点：
 - 结点的双亲是唯一的
 - 只能直接处理一对多的实体联系
 - 每个记录类型可以定义一个排序字段，也称码字段
 - 任何记录值 **只有按其路径查看**时，才能显出它的全部意义
 - 没有一个子女记录值能够脱离双亲记录值而独立存在



层次模型的数据操纵与完整性约束

122

- 层次模型的数据操纵
 - 查询
 - 插入
 - 删除
 - 更新



层次模型的数据操纵与完整性约束（续）

123

- 层次模型的完整性约束条件
 - 导航式查询方式
 - 无相应的双亲结点值就不能**插入**子女结点值
 - 如果**删除**双亲结点值，则相应的子女结点值也被同时删除
 - **更新**操作时，应更新所有相应记录，以保证数据的一致性



层次模型的优缺点

124

□ 优点

- 层次模型的数据结构比较简单清晰
- 查询效率高，性能优于关系模型，不低于网状模型
- 层次数据模型提供了良好的完整性支持

□ 缺点

- 节点之间的**多对多联系**表示不自然
- 对插入和删除操作的限制多，应用程序的编写比较复杂
- 查询子女结点必须通过双亲结点
- 由于结构严密，层次命令趋于程序化



1.2 数据模型

125

1.2.1 数据建模

1.2.2 概念模型

1.2.3 数据模型组成的三要素

1.2.4 层次模型

1.2.5 网状模型

1.2.6 关系模型

1.2.7* 数据库领域中不断涌现的数据模型



1.2.5 网状模型

126

□ 网状数据库系统采用**网状模型**作为数据的组织方式

□ 典型代表是DBTG系统

□ 亦称CODASYL系统

■ 美国数据系统语言委员会CODASYL

□ 70年代由DBTG提出的一个系统方案

■ CODASYL下属的数据库任务组DBTG

□ 奠定了数据库系统的基本概念、方法和技术

□ 实际系统

□ Cullinet Software公司的IDMS

□ Univac公司的 DMS1100

□ Honeywell公司的IDS/2

□ HP公司的IMAGE



查尔斯·巴赫曼

第一个没有博士学位的图灵奖获得者

第一个工程学背景而不是科学背景的图灵奖

第一个因将计算机应用于工商管理而赢得图灵奖

第一个因一个特定的软件而赢得图灵奖

第一个在职业生涯完全在企业中度过的图灵奖获得者

他的主要贡献不是在学术界任教研工作，而是在工业界开发实际的产品。



网状数据模型的数据结构

127

- 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

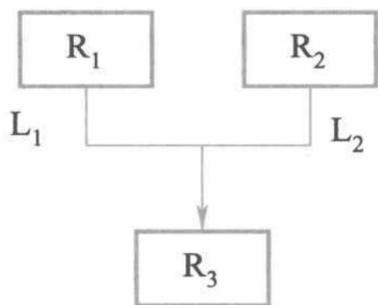


1. 网状数据模型的数据结构

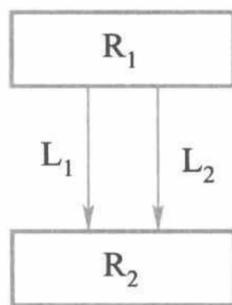
□ 网状模型

满足下面两个条件的基本层次联系的集合：

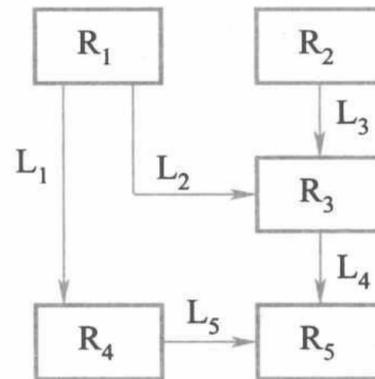
1. 允许一个以上的结点无双亲；
2. 一个结点可以有多个的双亲。



(a)



(b)



(c)

图 1.13 网状模型示例



网状数据模型的数据结构（续）

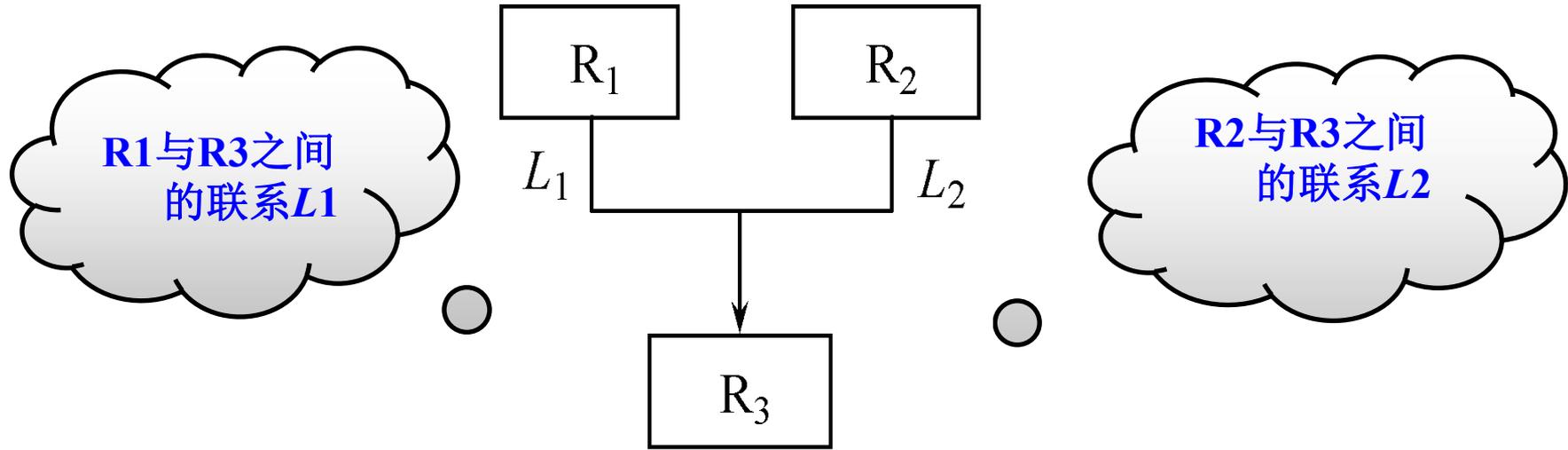
129

- 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



网状模型的例子 (a)



网状数据模型的数据结构（续）

131

例如：一个学生可以选修多门课程，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连接记录，由**5**个数据项组成
 - 学号
 - 课程号
 - 成绩
 - 开课学期
 - 教学班

- 表示某个学生在某个学期的某个教学班选修了某一门课程及其成绩



网状数据模型的数据结构（续）

图1.13 学生/选课/课程的网状数据模型





2. 网状数据模型的操纵与完整性约束

133

- 网状数据库系统（如DBTG）对数据操纵加了一些限制，提供了一定的完整性约束
 - 码：唯一标识记录的数据项的集合
 - 一个联系中双亲记录与子女记录之间是一对多联系
 - 支持双亲记录和子女记录之间某些约束条件
 - 允许插入尚未确定双亲结点值的子女节点值
 - 允许只删除双亲节点值
 - 但也可以具体由用户实现方式做改变（如DBTG属籍类别）



3. 网状数据模型的优缺点

134

- 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
 - 编写应用程序负担比较大



1.2 数据模型

135

1.2.1 数据建模

1.2.2 概念模型

1.2.3 数据模型组成的三要素

1.2.4 层次模型

1.2.5 网状模型

1.2.6 关系模型

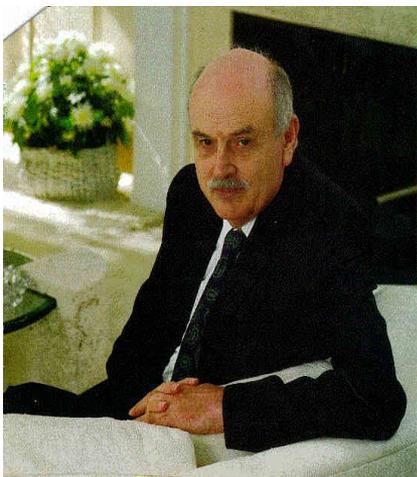
1.2.7* 数据库领域中不断涌现的数据模型



1.2.6 关系模型

136

- 关系数据库系统采用关系模型作为数据的组织方式
- 1970年美国IBM公司San Jose研究室的研究员E.F.Codd首次提出了数据库系统的关系模型
- 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



E.F.Codd

Database Systems

3/5/2025



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，它由行和列组成

图1.14 学生登记表

学号	姓名	性别	出生日期	主修专业
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
...

属性 (Attribute) label pointing to the first column (学号)

元组 (Tuple) label pointing to the first row (20180001, 李勇, 男, 2000-3-8, 信息安全)



关系数据模型的数据结构（续）

138

□ 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

□ 主码（Key）

- 表中的某个属性组，它可以唯一确定一个元组。



关系数据模型的数据结构（续）

139

- **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）

例：学生（学号，姓名，性别，出生日期，主修专业）



关系数据模型的数据结构

140

- 表示方法
 - 实体型：用关系表表示
 - 属性：用属性名表示
 - 联系
 - 一对一：隐含在实体对应关系中
 - 一对多：隐含在实体对应关系中
 - 多对多：用新的关系表表示



关系数据模型的数据结构（续）

141

例1

学生、系，系与学生之间的一对多联系：

学生（学号，姓名，年龄，性别，系号，年级）

系（系号，系名，办公地点）

例2

系、系主任，系与系主任间的一对一联系

职工（工号，姓名，年龄，性别）

系（系号，系名，办公地点，系主任工号）



关系数据模型的数据结构（续）

142

例3

学生、课程，学生与课程之间的多对多联系：

学生（学号，姓名，年龄，性别，系号，年级）

课程（课程号，课程名，学分）

选修（学号，课程号，成绩）—多对多联系



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**

表1.3 非规范化的表示例：表中有表

学号	姓名	性别	出生日期	主修专业	联系方式		
					手机号	Email	微信号
20180001	李勇	男	2000-3-8	信息安全	18301200745	liyong@qq.com	liyong@ruc
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

“联系方式”是可分的数据项, **不符合关系模型要求**



关系数据模型的数据结构（续）

144

表1.4 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

145

- 数据操作是集合操作，操作对象和操作结果都是关系（若干元组的集合）
 - 查询
 - 插入
 - 删除
 - 更新
- 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）

146

- 关系的完整性约束条件（含义将在第二章介绍）
 - 实体完整性
 - 参照完整性
 - 用户定义的完整性



三、关系数据模型的优缺点

147

- 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）

148

□ 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



知识扩展: NoSQL

□ NoSQL (Not Only SQL) , 泛指非关系型的数据库

分类	举例	典型应用场景	数据模型	优点	缺点
文档型数据库	MongoDb CouchDB	Web应用 (与Key-Value类似, Value是结构化的, 不同的是数据库能够了解Value的内容)	文档形式存储, 类似JSON的KV存储 Value: 简单类型 (字符串), 复杂类型 (表格, 对象)	数据结构要求不严格, 表结构可变, 不需要像关系型数据库一样需要预先定义表结构	查询性能不高, 而且缺乏统一的查询语法。
图形 (Graph) 数据库	Neo4J, InfoGrid, Infinite Graph	社交网络, 推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址, N度关系查找等	经常需对整个图做计算才能得出需要的信息, 且这种结构不太好做分布式集群方案
键值 (key-value)	Redis, Cabinet/Tyrant, Voldemort, Oracle BDB	内容缓存, 用户信息等, 如会话, 日志等。主要用于处理大量数据的高访问负载	Key 指向 Value 的键值对, 通常用 hash table 来实现	查找速度快, 可以通过 key 快速查询到其 value。一般来说, 存储不管 value 的格式, 照单全收。	数据无结构化, 通常只被当作字符串或者二进制数据
列存储数据库	HBase, Cassandra	日志, 分布式的文件系统, 时空数据等	以列簇式存储, 将同一列数据存在一起	查找速度快, 可扩展性强, 方便做数据压缩, 对针对某一列或者某几列的查询有IO优势。	功能相对局限 缺乏统一查询语言

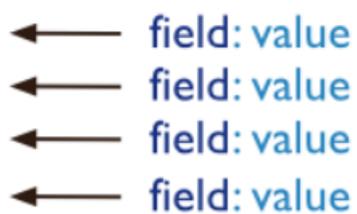


NoSQL: 文档数据存储-MongoDB

□ MongoDB简介

- 基于**分布式**文件存储，由 C++ 语言编写，旨在为 WEB 应用提供**可扩展的高性能**数据存储解决方案。
- 介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。
- 将数据存储为一个**文档**，数据结构由键值(**key=>value**)对组成，**类似于 JSON 对象**。Value可以包含其他文档，数组及文档数组

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```



能，还支持索引



与sql不同，各个字段不需要预先定义好数据类型，所以mongodb很灵活



NoSQL: 图(Graph)数据存储

如何表示图数据？

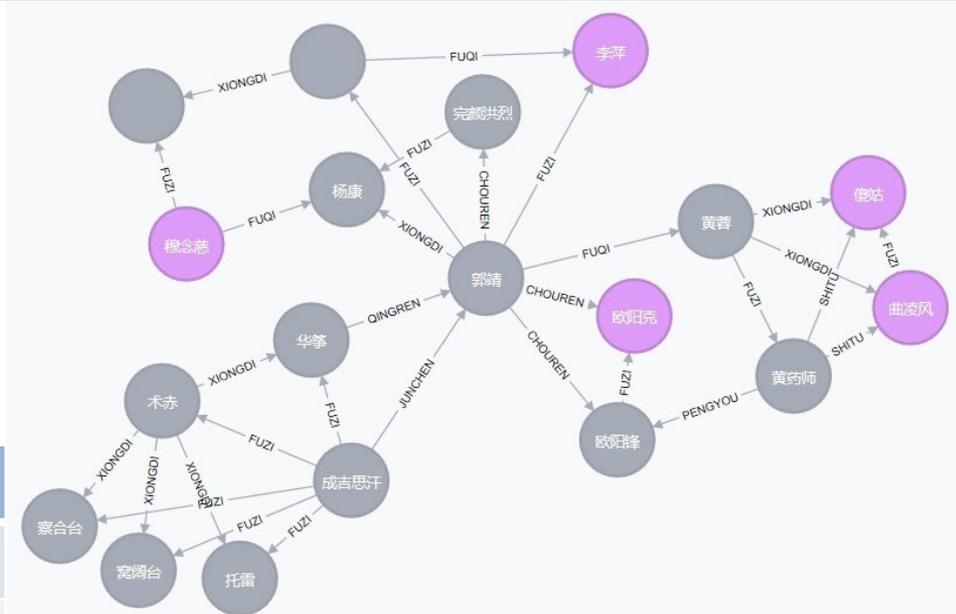
SQL

- 冗余数据
- 大量空白

名字	子女	兄弟	父亲	年龄
郭靖	破虏	杨康		33
杨康	\	郭靖	完颜洪烈	
破虏	\	\	郭靖	10

Neo4J

- 以“图形”的形式存储数据，符合图数据特性
- 方便删改



```

Create(n:person{name:"郭靖",age:"33" })
Create(m:person{name:"破虏",age:"10" })
create (n)-[:R{type:"父子"}]->(m)

```

cypher是neo4j官网提供的声明式查询语言



NoSQL: 图(Graph)数据存储

□ 删去郭靖

SQL

名字	子女	兄弟	父亲	年龄
郭靖	破虏	杨康		33
杨康	\	郭靖	完颜洪烈	
破虏	\	\	郭靖	10

Neo4j

`match(n:person{name:"郭靖"}) delete n`

□ 杨康和破虏是什么关系?

名字	子女	兄弟	父亲	年龄
郭靖	破虏	杨康		33
杨康	\	郭靖	完颜洪烈	
破虏	\	\	郭靖	10

```

match (n:Person{name:"杨康"}),
(m:Person{name:"破虏"}),
r=(n)-[]-(m),
return n,m,r

```



NoSQL: 图(Graph)数据存储

- Neo4J vs MySQL性能对比
 - 具有关系的数据
 - 例：在一个社交网络中找到深度为5的朋友，数据集约为100万人，平均每人50个朋友

~

实验结果如下：

深度	MySQL执行时间(s)	Neo4J执行时间(s)	返回记录数
2	0.016	0.01	~2500
3	30.267	0.168	~110 000
4	1543.505	1.359	~600 000
5	未完成	2.132	~800 000



NoSQL: 键值数据库

Redis (REmote DIctionary Server)

key-value 存储系统

应用场景

发布订阅，地图信息，计数器等

电商“秒杀”：

■ 短时间内极大量访问

■ 避免“超抢”、“超卖”



```
HMGET id Total Booked
HINCRBY id Booked 1
```

库存总量
订单总量

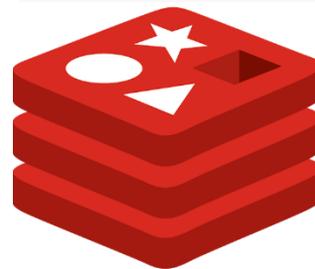
```
"goodsId" : {
  "Total": 100
  "Booked": 100
}
```



下单



秒杀成功/失败





第一章 绪论

163

1.1 数据库系统概述

1.2 数据模型

1.3 数据库系统的三级模式结构

1.4 数据库系统的组成

*1.5 数据库系统的体系结构

1.6 小结



1.3 数据库系统的三级模式结构

164

- 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



数据库系统的三级模式结构

165

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

166

- 数据模型：“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

学生记录类型：

(学号，姓名，性别，出生日期，主修专业)

一个记录值：

(**20180003**，王敏，女，**2001-8-1**，计算机科学与技术)



数据库系统模式的概念（续）

167

□ 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以及数据之间的联系
- 模式是相对稳定的

Schema与Database

□ 实例（Instance）

- 模式的一个具体值
- 反映数据库在某一时刻的状态（值）
- 同一个模式可以有很多实例
- 实例随数据库中的数据更新操作而变动



数据库系统模式的概念（续）

168

例如：学生选课数据库模式，

包含学生关系、课程关系和学生选课关系

- 学生（学号，姓名，性别，出生日期，主修专业）
- 课程（课程号，课程名，学分，先修课）
- 学生选课（学号，课程号，成绩，开课学期，教学班）



数据库系统模式的概念（续）

169

在学生选课**数据库模式**中，包含学生记录、课程记录和学生选课记录

- 2018级的一个学生**数据库实例**，包含：
 - 学校2018级所有学生的记录
 - 学校开设的所有课程的记录
 - 所有学生选课的记录
- 2017级学生数据库模式对应的实例与2018级学生数据库模式对应的**实例是不同的**



数据库系统模式的概念（续）

170

Sno	Sname	Ssex	Sbirthdate	Smajor
20180001	李勇	男	2000-3-8	IS
20180002	刘晨	女	1999-9-1	CS
20180003	王敏	女	2001-8-1	CS

Cno	Cname	Ccredit	Cpno
81001	程序设计基础与C语言	4	
81002	数据结构	4	81001
81003	数据库系统概论	4	81002

Sno	Cno	Grade	Semester	Teachingclass
2018001	81001	85	20192	81001-01
2018001	81002	96	20201	81002-01
2018001	81003	87	20202	81003-01
2018002	81001	80	20192	81001-02
2018002	81002	98	20201	81002-01



数据库系统的三级模式结构

171

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



1.3.2 数据库系统的三级模式结构

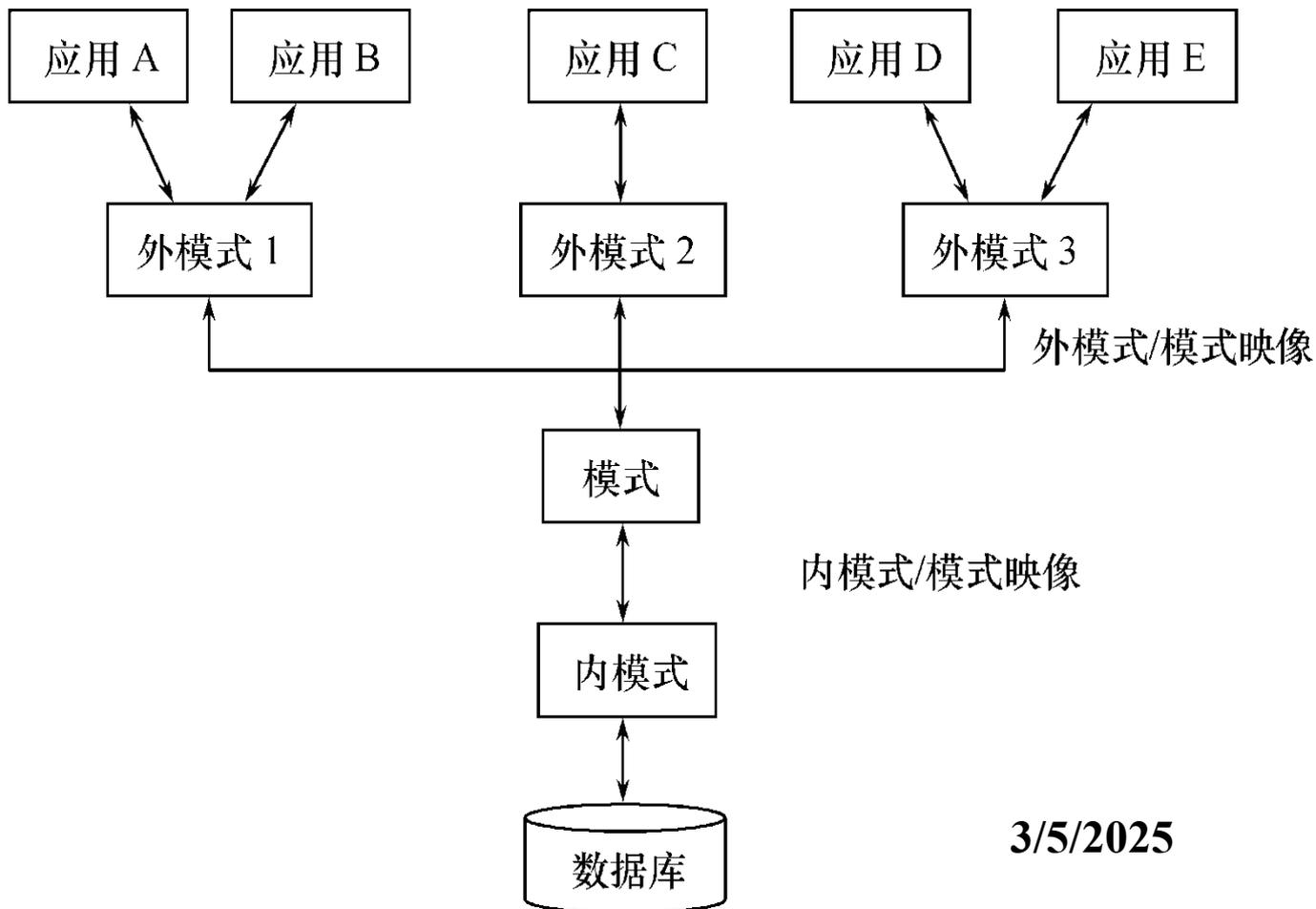
172

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



数据库系统的三级模式结构（续）

图1.15 数据库系统的三级模式结构





一、模式（Schema）

174

- 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
 - 是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
 - Python, C, JAVA



模式（续）

175

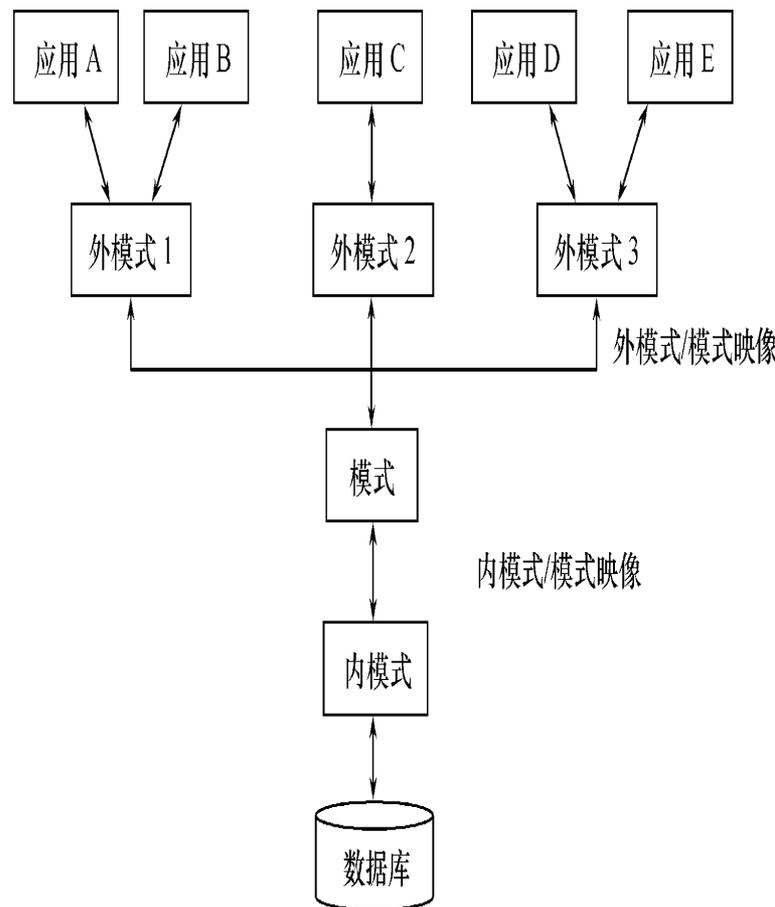
□ 模式的定义

- 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（与文件系统区别）
- 数据有关的安全性、完整性要求（共享带来的需求）
- DBMS用数据定义语言DDL严格定义模式（第三章）



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, 是与某一部局部应用有关的数据的逻辑表示
 - 例如, 数据库课程的成绩





外模式（续）

177

- 外模式的地位：介于模式与应用程序之间
 - 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
 - 对模式中同一数据，在外模式中的结构、类型、长度、保密级别等都可以不同
 - 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）

178

□ 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其所对应的外模式中的数据（不应该看到的数据不在外模式中）
- 保证数据独立性的措施
 - 映射



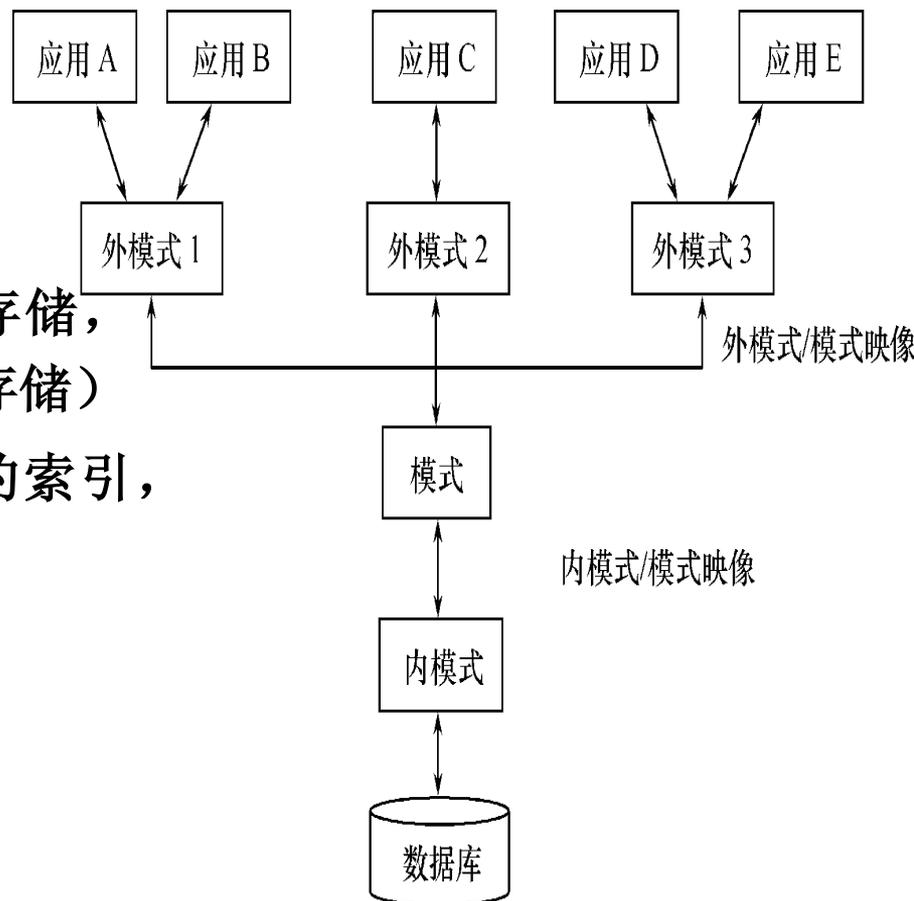
三、内模式 (Internal Schema)

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 (例如, 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 (例如, B+树的索引, Hash的索引)
- 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
- DBMS定义与管理

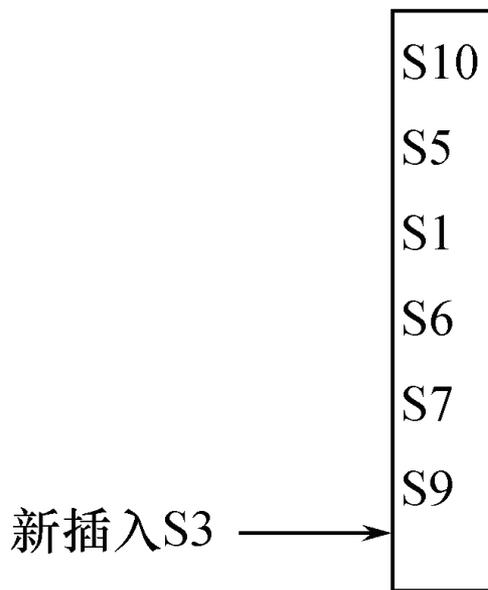




内模式（补充）

180

- 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

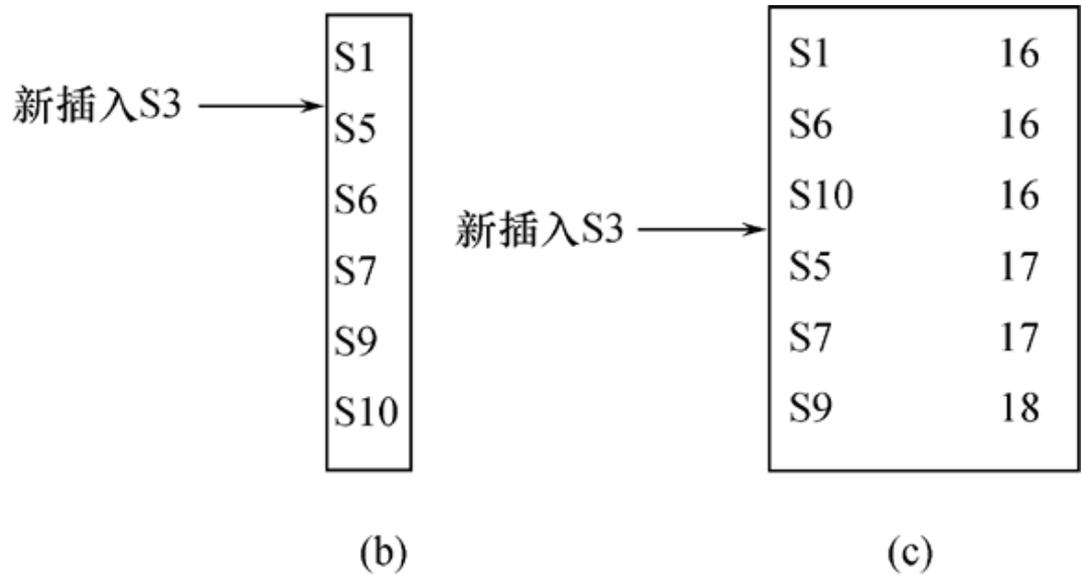


(a)



内模式（补充）

- 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





数据库系统的三级模式结构

182

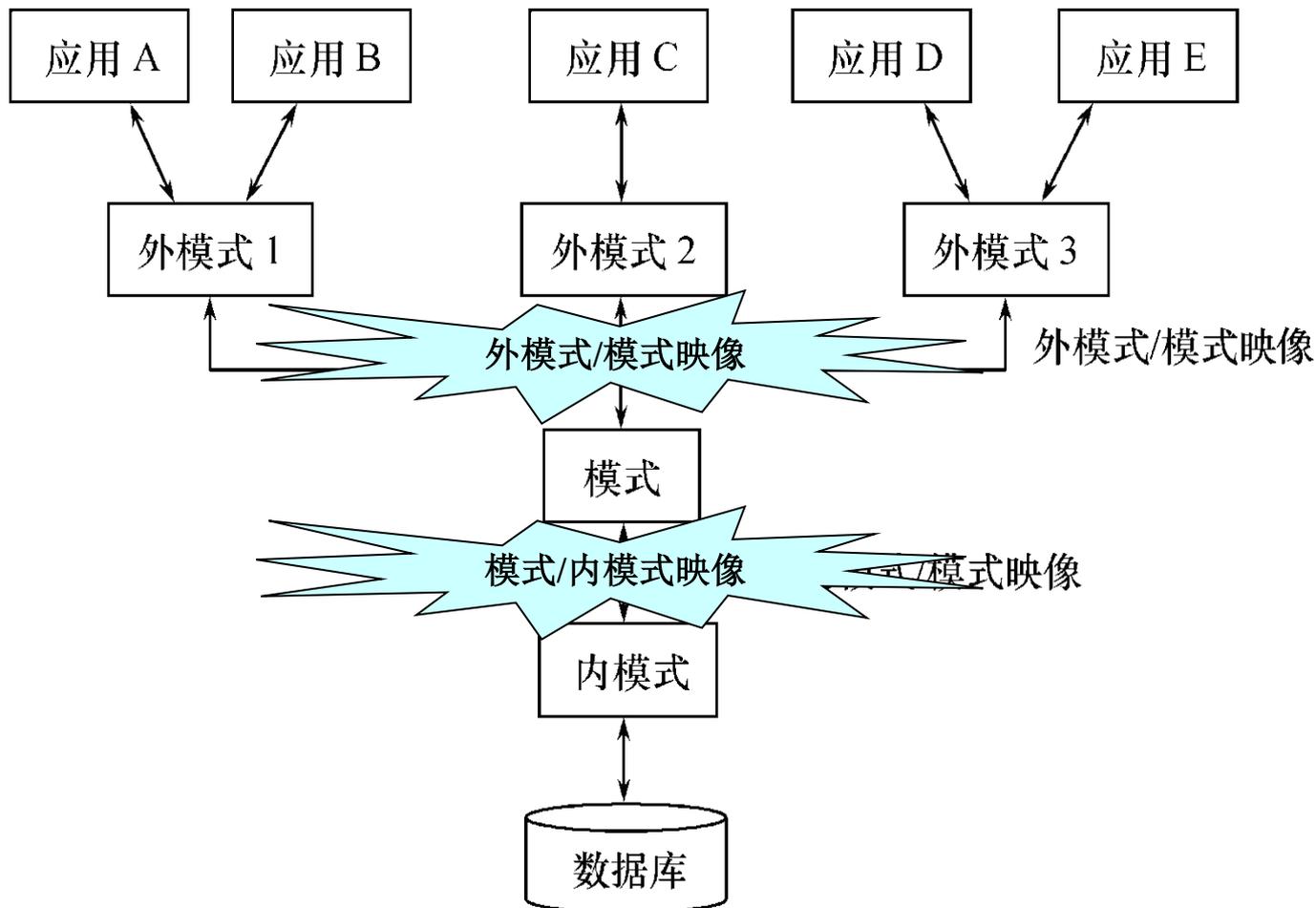
1.3.1 数据库系统模式的概念

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





1.3.3 数据库的二级映像功能与数据独立性

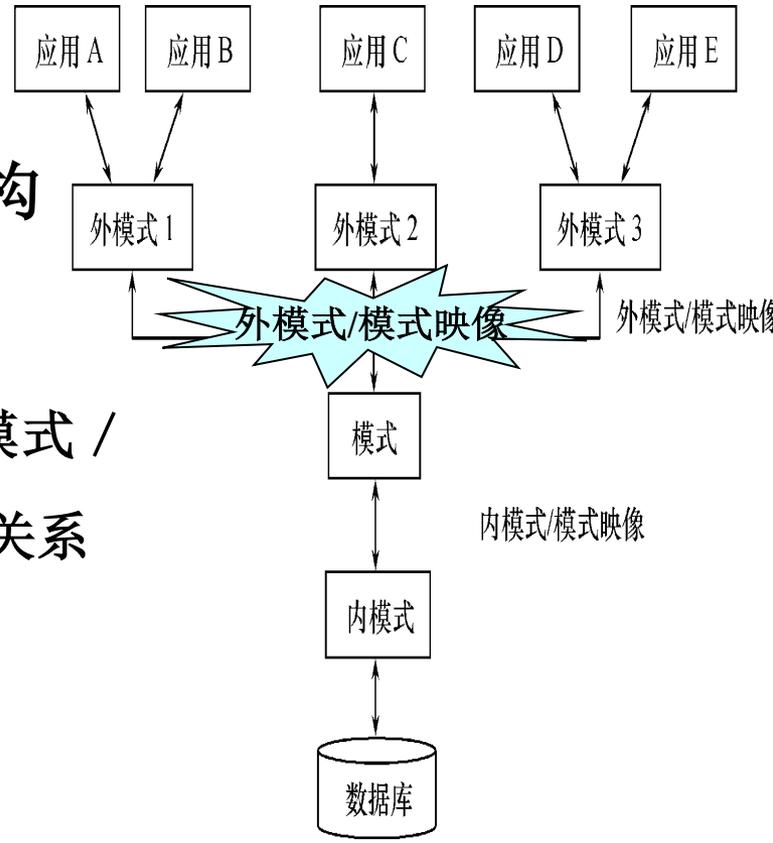
184

- 三级模式是对数据的三个抽象级别
- 数据与程序的独立性
 - 数据的逻辑或物理结构发生变化，应用程序不需改变
- 二级映象在DBMS内部实现这三个抽象层次的联系和转换，保证数据与程序的独立性
 - 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





外模式 / 模式映象（续）

186

□ 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 / 模式映象，可以使外模式保持不变
- **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
 - 修改 外模式/模式 的映射（**create view**）
- 外模式 / 模式映象可以保证获得一定程度的数据的逻辑独立性（非完全独立）

学生（**学号**，姓名，年龄，院系号）

本科生，硕士生，博士生



二、模式 / 内模式映像

187

- 模式 / 内模式映像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 / 内模式映像是**唯一**的
- 该映像定义通常包含在模式描述中



模式 / 内模式映象（续）

188

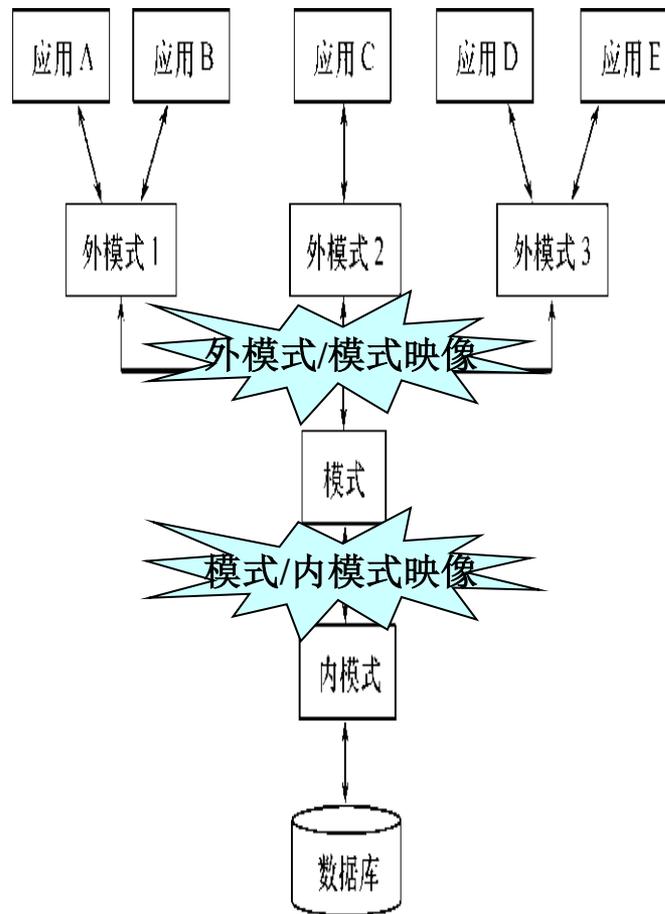
- 保证数据的物理独立性
 - 当数据库的存储结构改变了，数据库管理员修改模式 / 内模式映象，使模式保持不变
 - 例如，存储结构：由堆存储—hash存储
 - 从而应用程序不受影响。保证了数据与程序的物理独立性，简称数据的物理独立性
 - Create index
 - 模式 / 内模式映象保证获得完全的数据物理独立性



数据库三级小结

□ 数据库模式

- 即全局逻辑结构是数据库的中心与关键
- 独立于数据库的其他层次
- 设计数据库模式结构时应首先确定数据库的逻辑模式

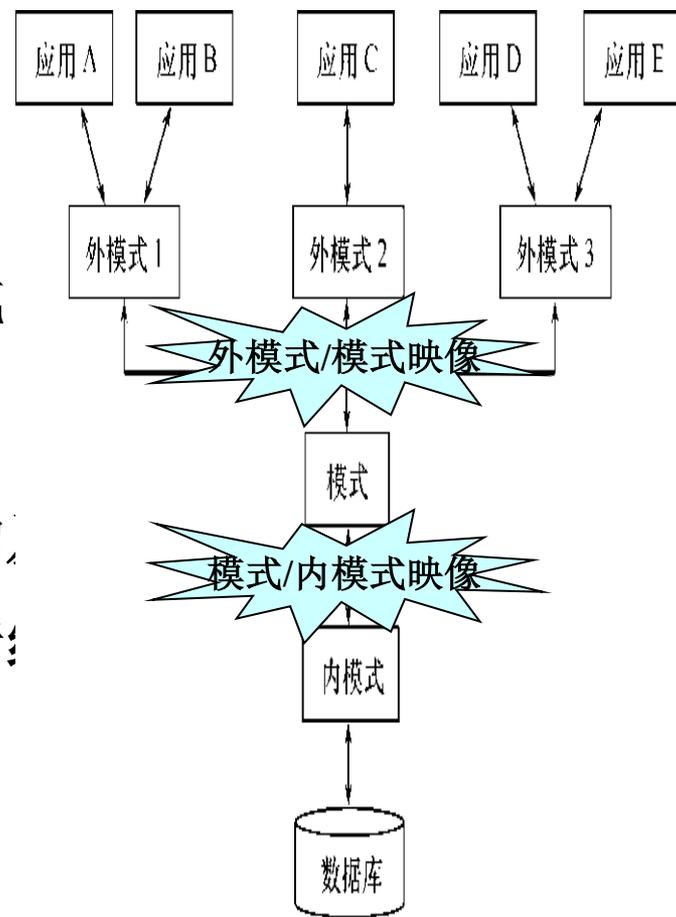




数据库三级模式小结

数据库的内模式

- 依赖于它的全局逻辑结构
- 独立于数据库的用户视图，即外模式
- 独立于具体的存储设备
- 将全局逻辑结构中所定义的数据结构，其联系按照一定的物理存储策略进行组织，以达到较好的时间与空间效率

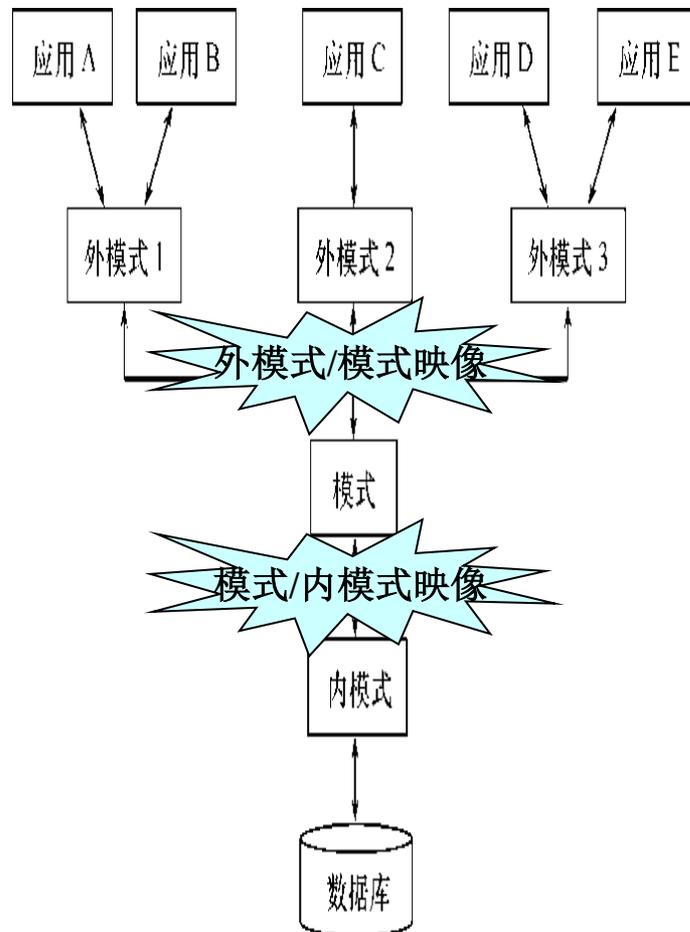




数据库三级模式小结

数据库的外模式

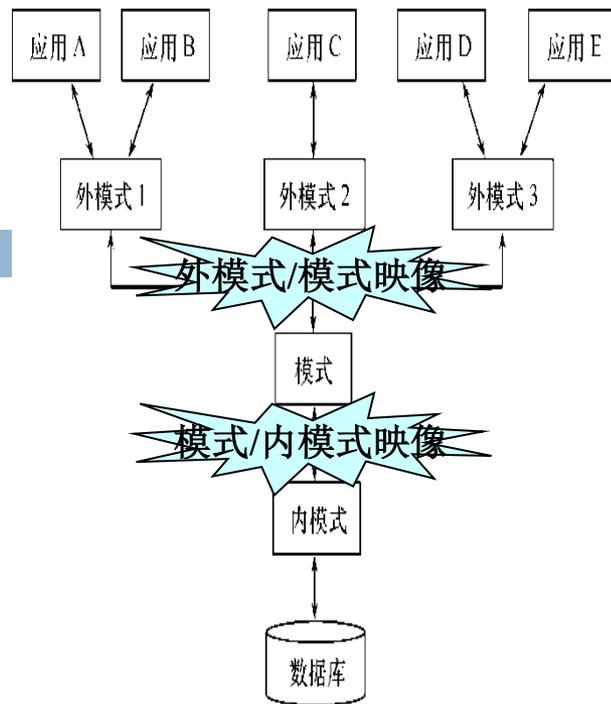
- 面向具体的应用程序
- 定义在逻辑模式之上
- 独立于存储模式和存储设备
- 当应用需求发生较大变化，相应外模式不能满足其视图要求时，该外模式就得做相应改动
- 设计外模式时应充分考虑到应用的扩充性





数据库三级模式小结

- 特定的应用程序
 - 在外模式描述的数据结构上编制的
 - 依赖于特定的外模式
 - 与数据库的模式和存储结构独立
 - 不同的应用程序有时可以共用同一个外模式
- 数据库的二级映像
 - 保证了数据库外模式的稳定性
 - 从底层保证了应用程序的稳定性，除非应用需求本身发生变化，否则应用程序一般不需要修改





数据库三级模式小结

193

- 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



第一章 绪论

194

1.1 数据库系统概述

1.2 数据模型

1.3 数据库系统的三级模式结构

1.4 数据库系统的组成

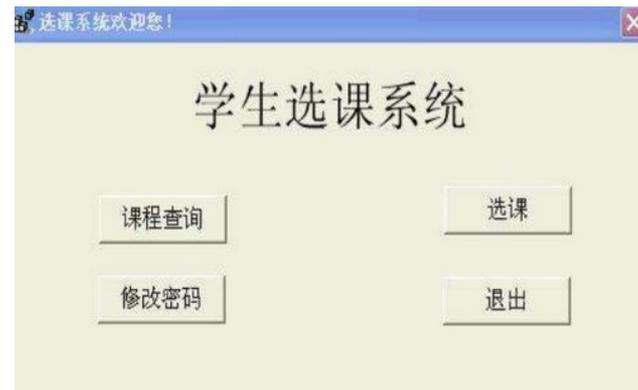
*1.5 数据库系统的体系结构

1.6 小结



1.4 数据库系统的组成

- 数据库
- 数据库管理系统（及其开发工具）
- 应用系统
- 数据库管理员





数据库系统的组成（续）

196

- 硬件平台及数据库
- 软件
- 人员



一、硬件平台及数据库

197

□ 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

(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

198

- **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

199

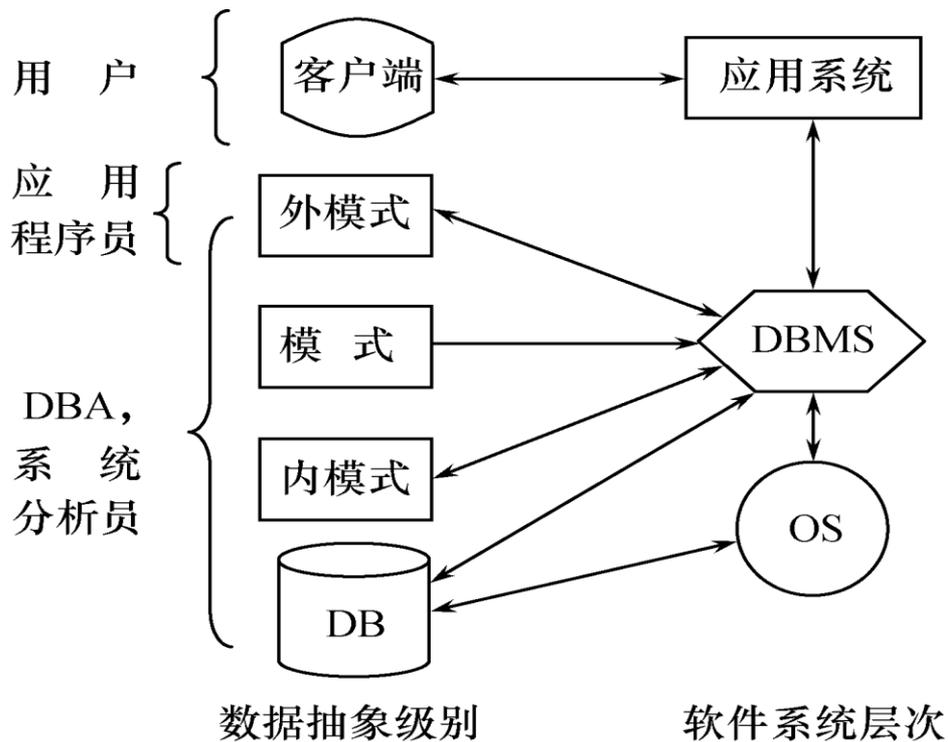
- 数据库管理员
- 系统分析员和数据库设计人员
- 应用程序员
- 用户



人员 (续)

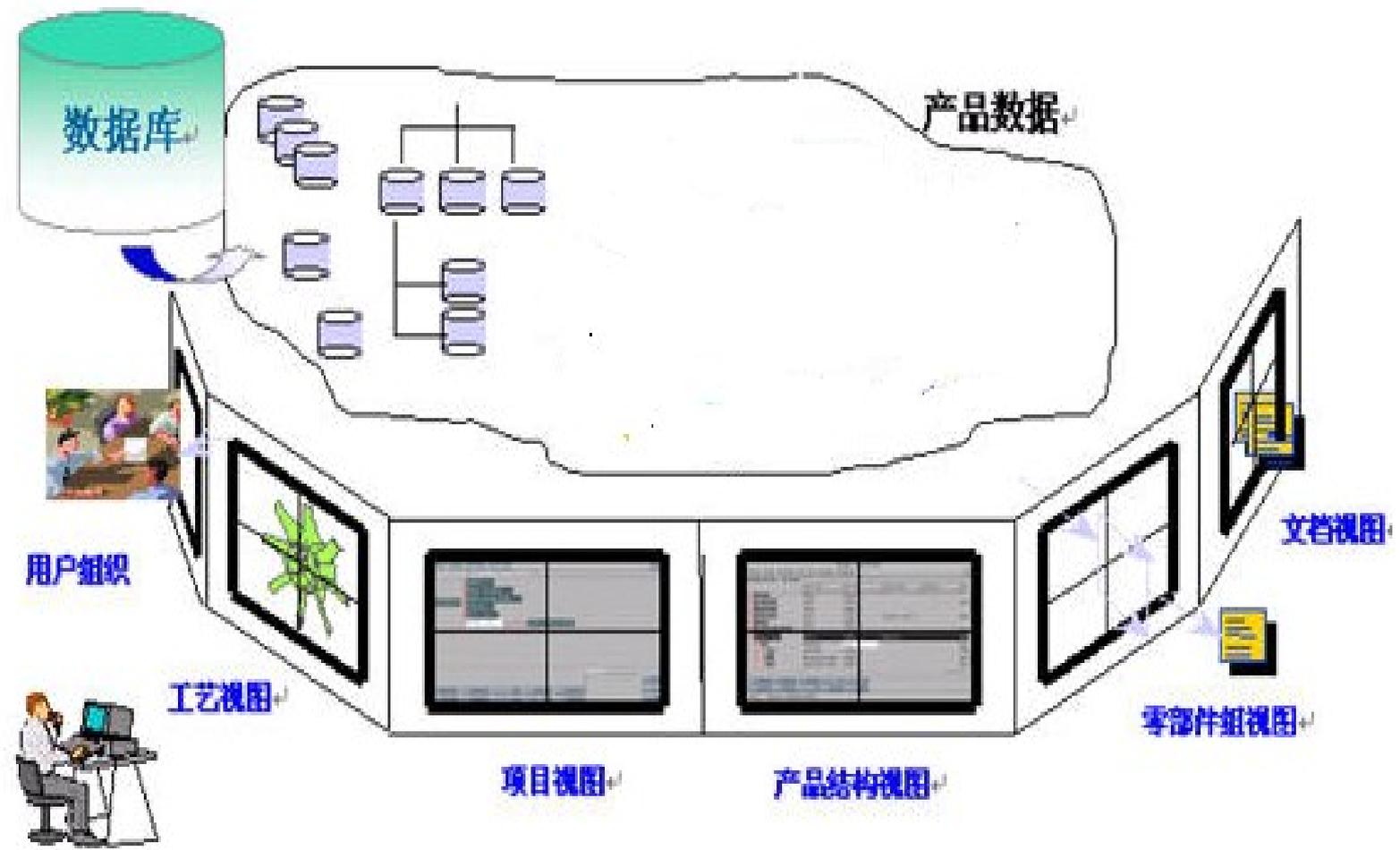
- 不同的人员涉及不同的数据抽象级别，具有不同的数据视图，如下图所示

图1.16 各种人员的数据视图





人员 (续)





1. 数据库管理员(DBA)

202

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
- 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
- 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

203

□ 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，确定系统的硬软件配置
- 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



3. 应用程序员

204

- 设计和编写应用系统的程序模块
- 进行调试和安装



4. 用户

205

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）

206

□ 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



第一章 绪论

208

1.1 数据库系统概述

1.2 数据模型

1.3 数据库系统的三级模式结构

1.4 数据库系统的组成

*1.5 数据库系统的体系结构

1.6 小结



1.6 小结

- 数据库系统概述
 - 数据库的基本概念
 - 数据管理的发展过程
- 数据模型
 - 数据模型的三要素
 - 数据结构、数据操作、数据的完整性约束条件
 - 概念模型， E-R 模型
 - 三种主要数据库模型
 - 层次模型、网状模型、关系模型



小结(续)

210

- 数据库系统的结构
 - 数据库系统三级模式结构
 - 数据库系统两层映像系统结构
- 数据库系统的组成



冲浪在数据潮头的实干家

211

课外阅读：数据库界的四位图灵奖得主

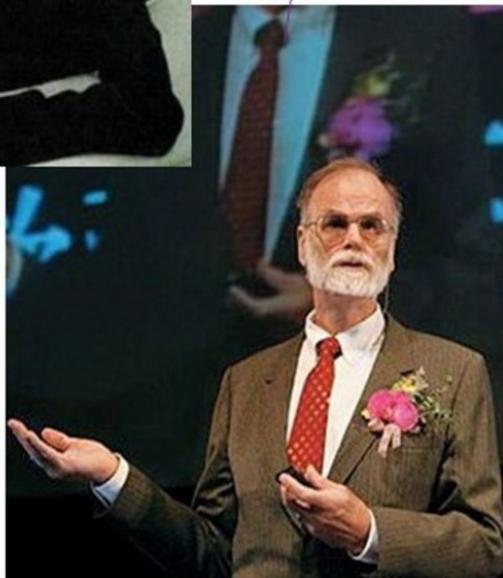


Charles W. Bachman
网状数据库，奠基者兼实践者



Jim Gray
事务处理，数据库系统实现

Michael Stonebraker
现代数据库概念与实践



E. F. Codd
关系数据库



数据库界的四位图灵奖得主



下课了。。。。



休息一会儿。。。。

