2024年春季学期

数据库系统概论

An Introduction to Database Systems

第一章　绪论

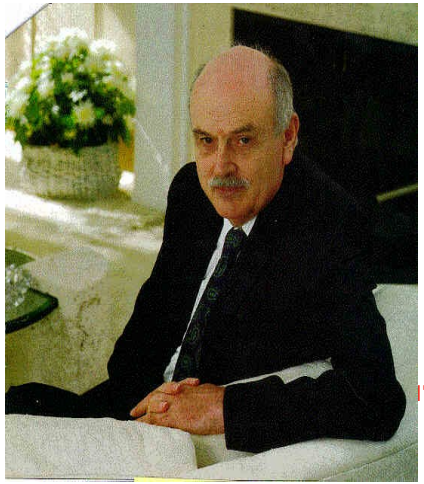
中国科学技术大学 大数据学院

黄振亚，huangzhy@ustc.edu.cn

1.2 数据模型

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
- 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

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	年 龄	性 别	系 名	年 级
2013004	王小明	19	女	社会学	2013
2013006	黄大鹏	20	男	商品学	2013
2013008	张文斌	18	女	法律	2013
…	…	…	…	…	…

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
- **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）

- **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，年龄，性别，系，年级）

关系数据模型的数据结构

- 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）

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）

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

图1.15 一个工资表(表中有表)实例

职工号	姓名	职 称	应发工 资			扣 除		实发工资
			基 本	津 贴	职务	房 租	水 电	
86051	陈 平	讲 师	1305	1200	50	160	112	2283
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

工资和扣除是可分的数据项，不符合关系模型要求

关系数据模型的数据结构（续）

表1.5 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（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）

- 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

- 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

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

知识扩展：NoSQL

- NoSQL适用于
  - 数据模型比较简单；
  - 需要灵活性更强的IT系统；
  - 对数据库性能要求较高；
  - 不需要高度的数据一致性；
  - 对于给定key，比较容易映射复杂值的环境
- 比较
  - NoSQL数据库的产生是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题
  - NoSQL结构比较简单，逻辑控制相对较少，同等存量下数据量超过关系型数据库，但是处理能力不一定高
  - 对于数据间有固定模式且紧密联系的，还是建议选择关系型数据库。

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",          <-- field: value
  age: 26,              <-- field: value
  status: "A",          <-- field: value
  groups: [ "news", "sports" ]   <-- field: value
}
```

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：什么时候使用MongoDB?

- 一般情况
  - Sql能存储的一般都可以用mongodb存储（事务除外）
- 非常适合日志、博客等比较杂乱的系统的存储
  - 种类较多、范围较大、内容也比较杂乱
  - 原因：在数据集合collection中，document对字段没有强约束
- 大文件存储
  - 二进制存储，可以用pickle等工具包对其进行压缩
  - GridFS：是MongoDB规范，用于存储和检索图片、音频、视频等大文件，可以存储超过16M的文件

NoSQL：图像(Image)存储

- 如何存储图形图像数据？**两种存储方式**

存储图片路径

将图片存在本地，比如windows系统中，在数据库中写入图片的路径，用来索引图片

id	path
1	/image/apple.jpg
2	/image/car.jpg
3	/image/cat.jpg

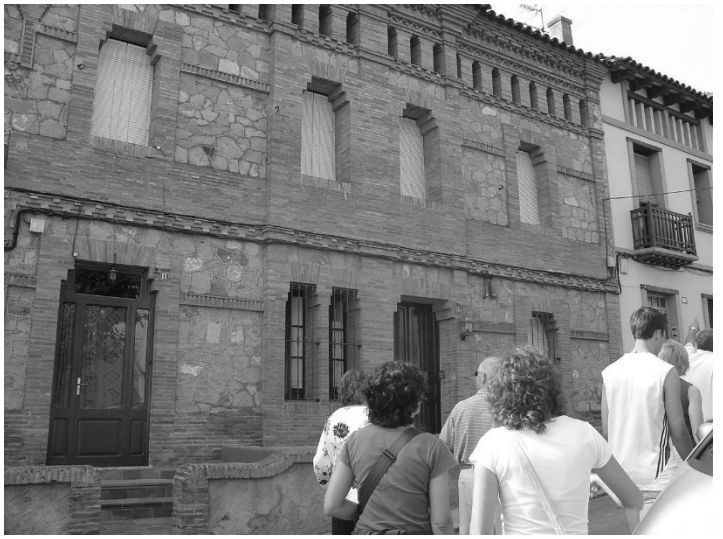
存储图像数据

直接将图像数据存入数据库系统中。可以直接提取图像的像素值，存为numpy, json等格式数据，写入数据库系统中。也可以将其以二进制文件的格式写入。

id	data
1	[[137 124 143 111 62 248 253] [133 116 160 125 133 153 85] [102 122 123 137 142 128 130] [116 52 121 121 56 124 98] [99 116 118 36 127 134 169] [67 119 89 253 158 222 204] [100 54 110 62 202 213 184]]

NoSQL：图像(Image)存储

height: 960
width: 1280



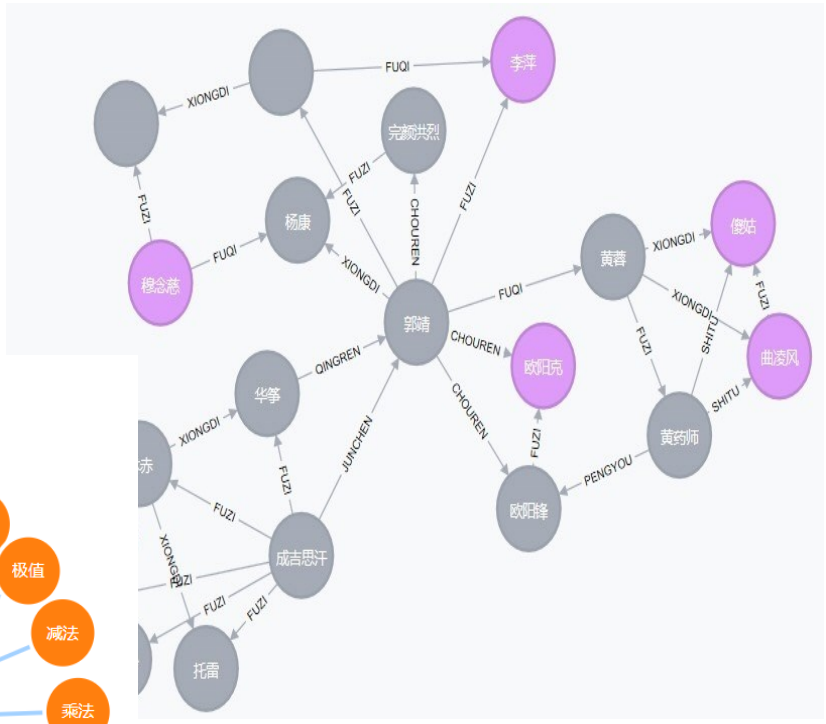
彩色图

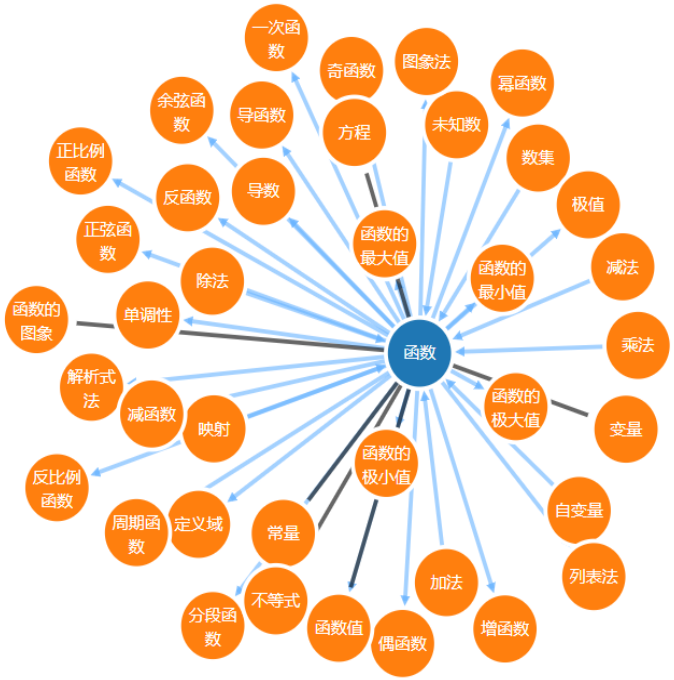
灰度图

当计算机看到一张图像时，它看到的是一堆像素值。
对于左边彩色图，它将看到一个960*1280 *3的数组，3指代的是RGB三个通道；
对于右边灰度图，它将看到一个960*1280的数组。
其中，每个数字的值从0到255不等，其描述了对应那一点的像素灰度。
所以，计算机对图像做处理时，实际上就是对这些数组中的像素值做处理。

NoSQL：图(Graph)数据存储

- 图数据
  - 社交网络
  - 知识图谱
  - 推荐系统
  - 欺诈检测
  - …

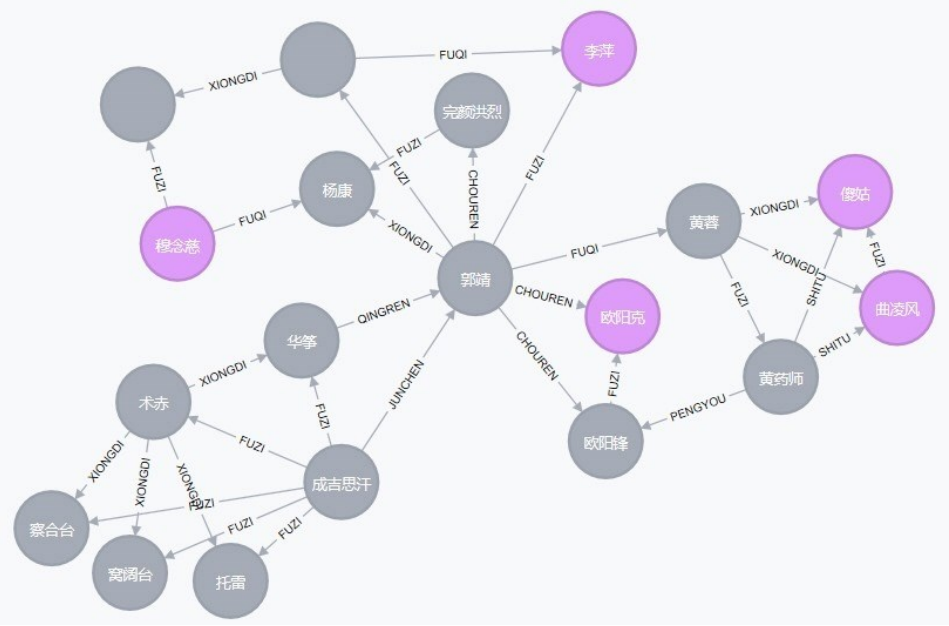




NoSQL：图(Graph)数据存储

- 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

- 键值数据库：Redis
  - 数据必须存储在某个key下
  - 数据可以是：
    - 整数
    - 字符串
    - 列表
    - 哈希表
    - 二进制
  - 只提供数据的基本操作
- 因为简单，所以快！

Key Value

```
SET counter 10
INCR counter => 11
GET counter => 11

RPUSH names “taylor”
RPUSH names “swift”
LLEN names => 2
LPOP names => “taylor”

HSET user:1000 name "John"
HSET user:1000 password "s3cret”
HGET user:1000 name => "John"
```

第一部分是存储数字（设置，自增，获取）
第二部分是存储队列（入队，获取长度，出队）
第三部分是存储字典（设置值、获取值）

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
}
```

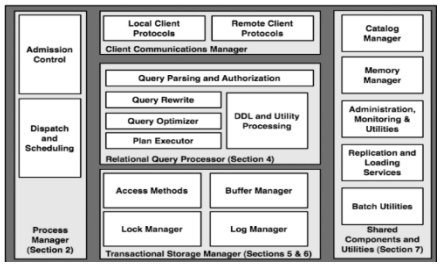
下单

秒杀成功/失败

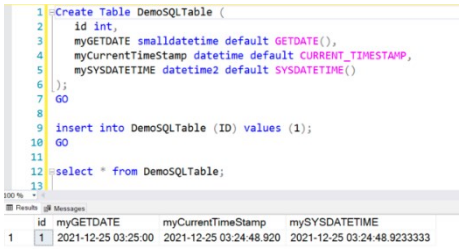
大模型与数据库

- 大模型与数据库

数据库 (1970s) → 大数据分析 (2010s) → 智能大模型 (2020s)



数据库系统



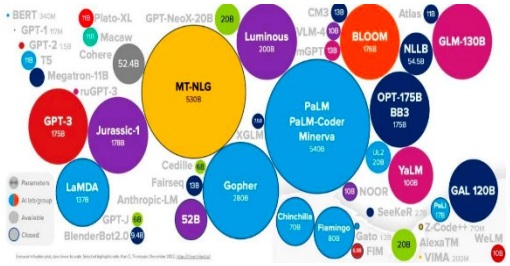
```
Create Table DemoSQLTable (
    id int,
    myGETDATE smalldatetime default GETDATE(),
    myCurrentTimeStamp datetime default CURRENT_TIMESTAMP,
    mySYSDATETIME datetime2 default SYSDATETIME()
);
GO

insert into DemoSQLTable (ID) values (1);
GO

select * from DemoSQLTable;
```

	id	myGETDATE	myCurrentTimeStamp	mySYSDATETIME
1	1	2021-12-25 03:25:00	2021-12-25 03:24:48.920	2021-12-25 03:24:48.9233333

SQL

互联网

搜索引擎



大模型

生成试问答

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

- 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以及数据之间的联系
  - 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的更新操作而变动

Schema与Database

数据库系统模式的概念（续）

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，年龄，院系号）
- 课程（课程号，课程名，学分）
- 选修（学号，课程号，成绩）

数据库系统模式的概念（续）

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

- 2013年的一个学生数据库实例，包含：
  - 2013年学校中所有学生的记录
  - 学校开设的所有课程的记录
  - 所有学生选课的记录
- 2012年度学生数据库模式对应的实例与2013年度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）

Sno	Sname	Ssex	Sage	Sdept
001	李勇	男	20	CS
002	刘晨	女	19	CS

Cno	Cname	Cpno	Ccredit
1	数据库	3	3.5
2	数学		4
3	数据结构		4

Sno	Cno	Grade
001	1	92
001	2	85
001	3	88
002	2	90
002	3	80

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

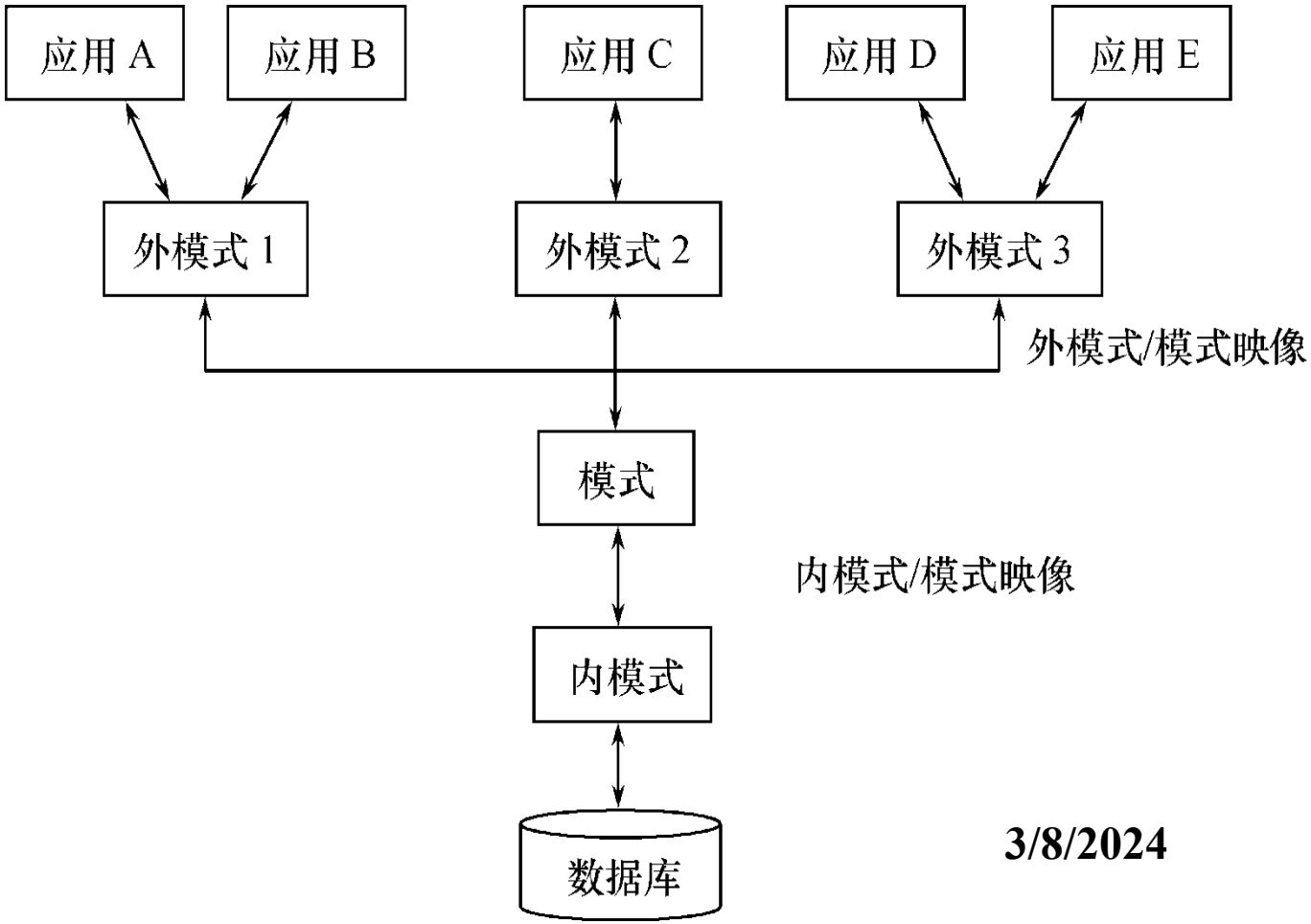
1.3.3 数据库的二级映像功能与数据独立性

1.3.2 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）

数据库系统的三级模式结构（续）

图1.16 数据库系统的三级模式结构



一、模式（Schema）

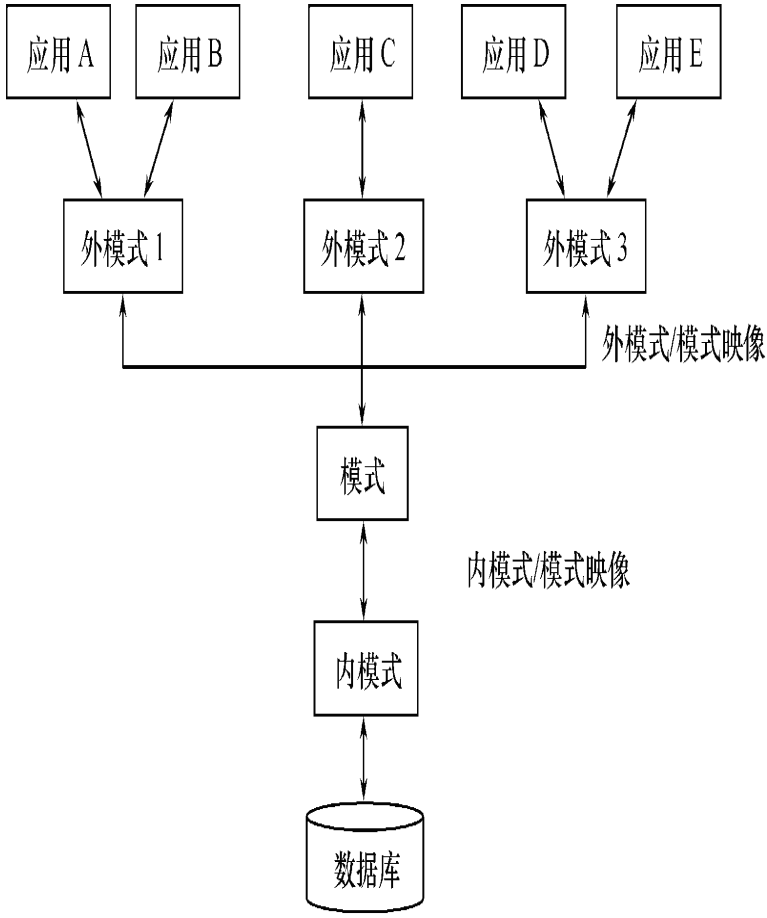
- 模式（也称逻辑模式）
  - 数据库中全体数据的逻辑结构和特征的描述
  - 是所有用户的公共数据视图，综合了所有用户的需求
- 一个数据库只有一个模式
- 模式的地位：是数据库系统模式结构的中间层
  - 与数据的物理存储细节和硬件环境无关
  - 与具体的应用程序、开发工具及高级程序设计语言无关
    - Python，C，JAVA

模式（续）

- **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）

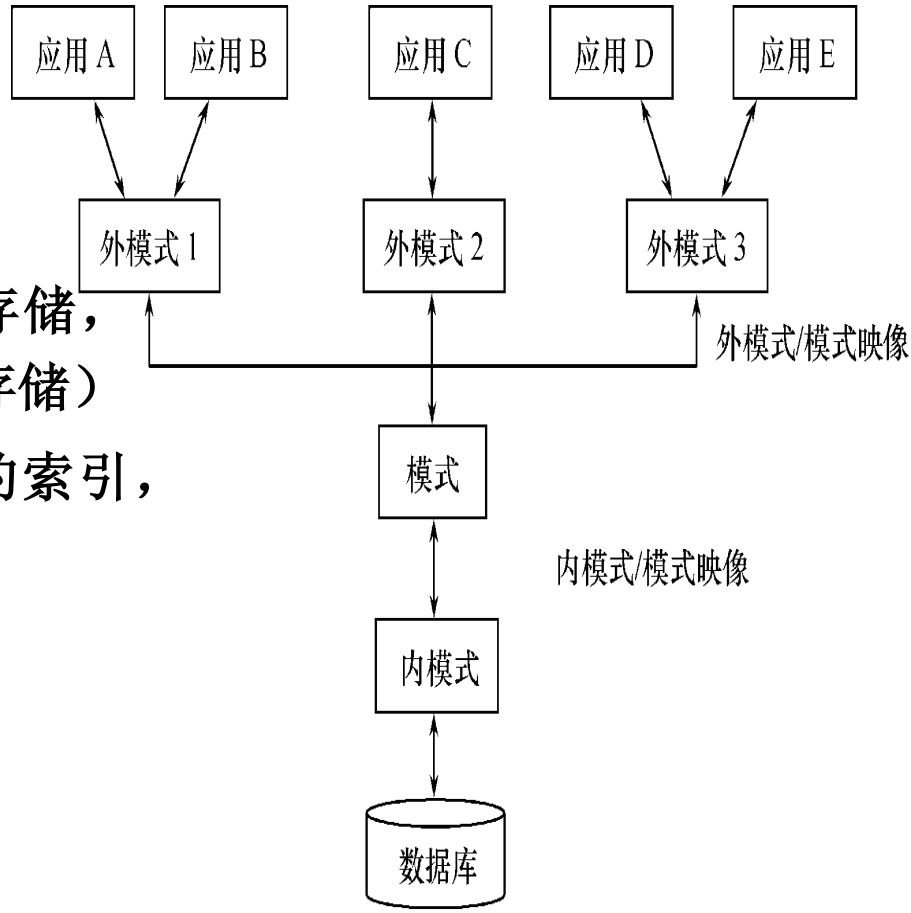
- 外模式的地位：介于模式与应用程序之间
  - 模式与外模式的关系：一对多
    - 外模式通常是模式的子集
    - 一个数据库可以有多个外模式。反映了不同的用户的应用需求、看待数据的方式、对数据保密的要求
    - 对模式中同一数据，在外模式中的结构、类型、长度、保密级别等都可以不同
  - 外模式与应用的关系：一对多
    - 同一外模式也可以为某一用户的多个应用系统所使用
    - 但一个应用程序只能使用一个外模式

外模式（续）

- 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

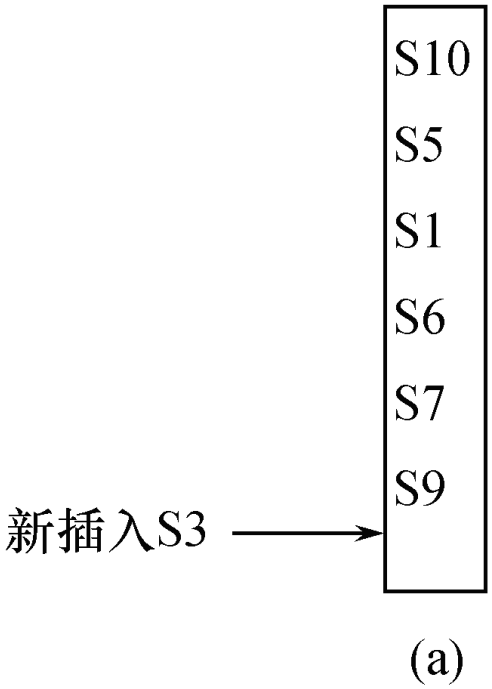
三、内模式（Internal Schema）

- 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）

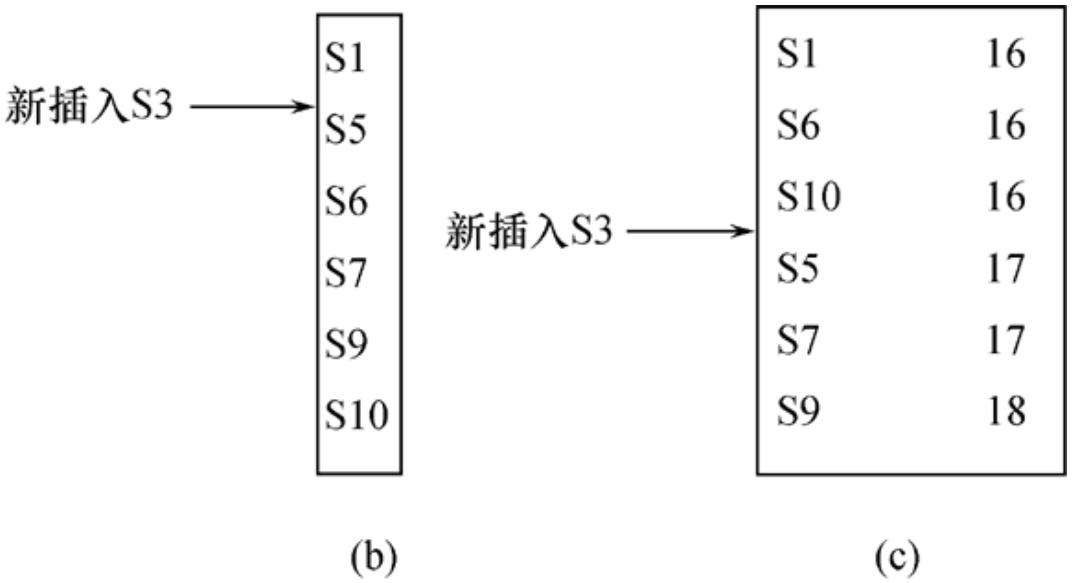
- 例如，学生记录，如果按**堆**存储，则插入一条新记录总是放在学生记录存储的**最后**



(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

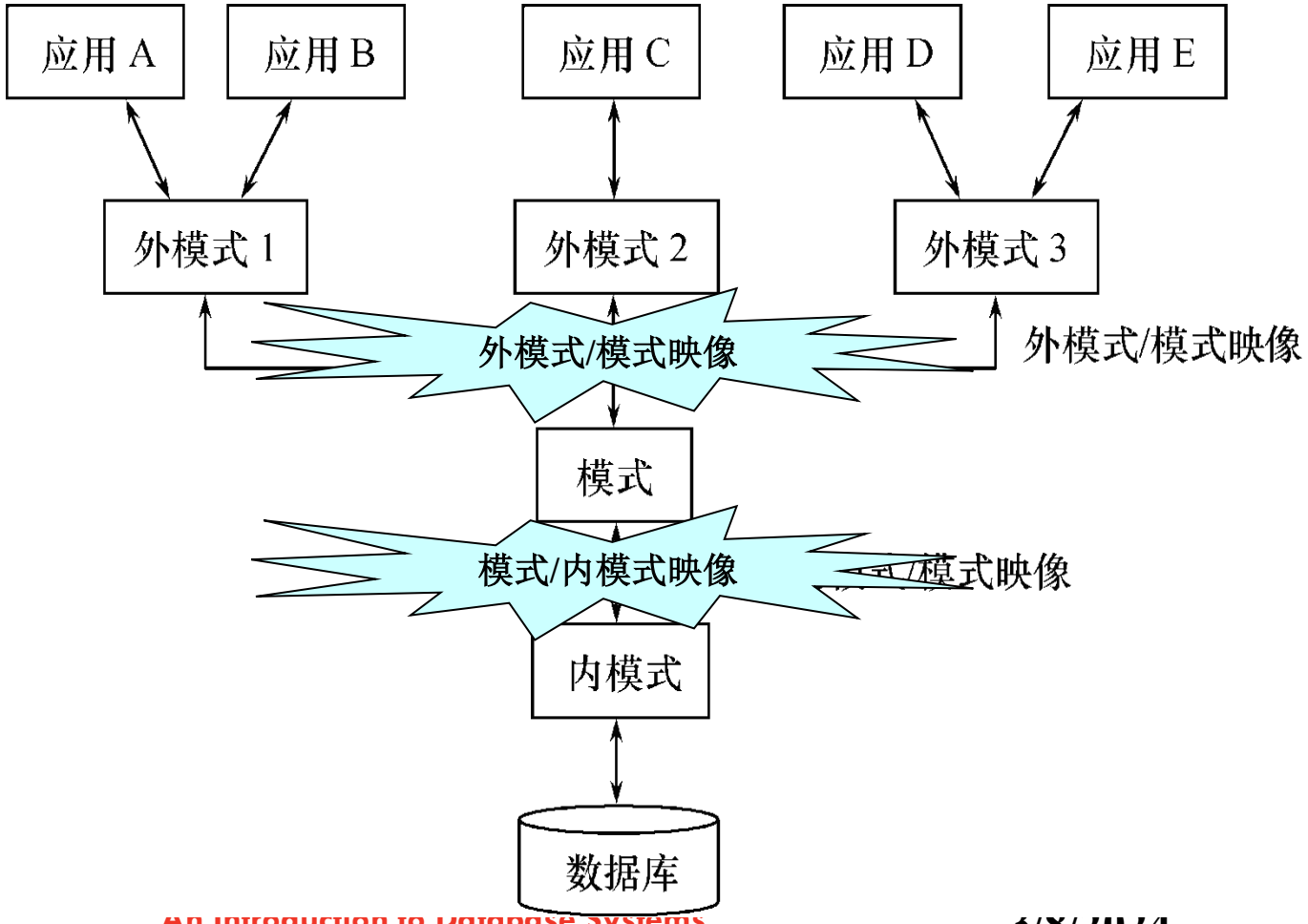
1.3.2 数据库系统的三级模式结构

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

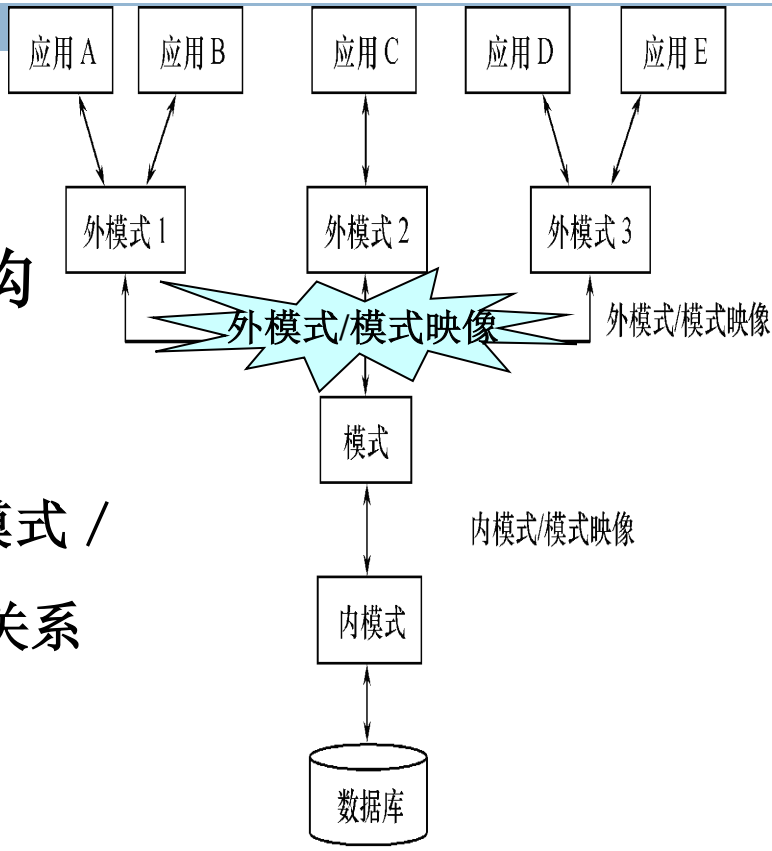
- **三级模式是对数据的三个抽象级别**
- **数据与程序的独立性**
  - 数据的逻辑或物理结构发生变化，应用程序不需改变
- **二级映象在DBMS内部实现这三个抽象层次的联系和转换，保证数据与程序的独立性**
  - 外模式 / 模式映像
  - 模式 / 内模式映像

1.3.3 数据库的二级映像功能与数据独立性



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）

- **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

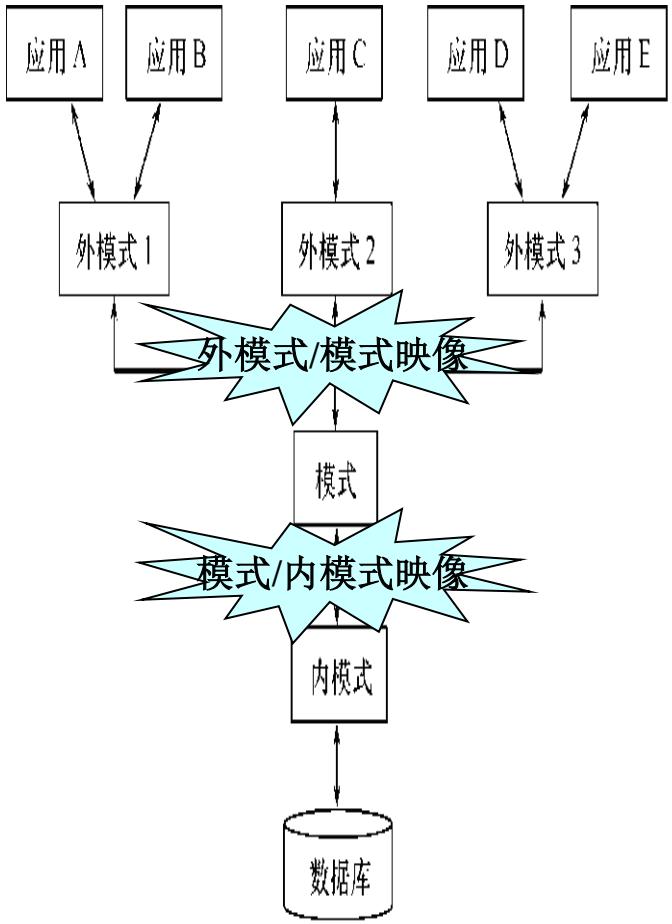
- 模式／内模式映象定义了数据全局逻辑结构与存储结构之间的对应关系
  - 例如，说明逻辑记录和字段在内部是如何表示的
- 数据库中模式／内模式映象是唯一的
- 该映象定义通常包含在模式描述中

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，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

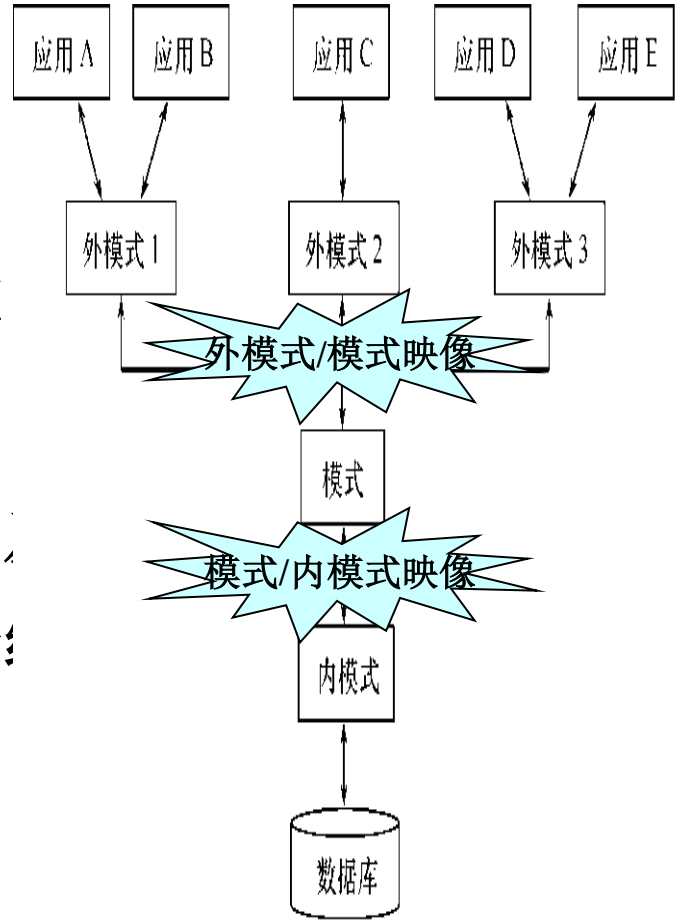
数据库三级小结

- 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



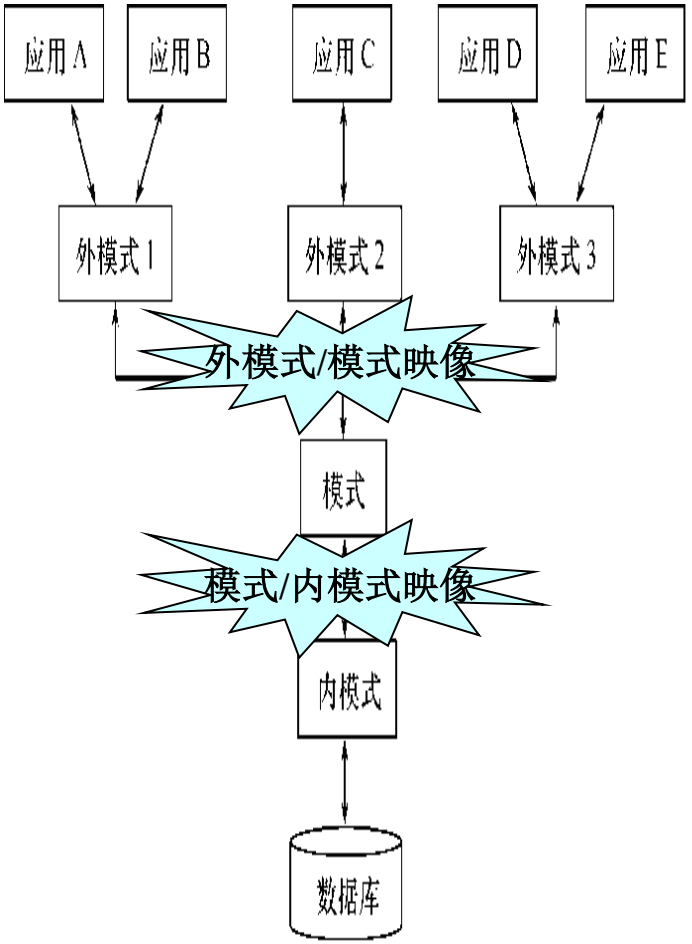
数据库三级模式小结

- 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
    织，以达到较好的时间与空间效率

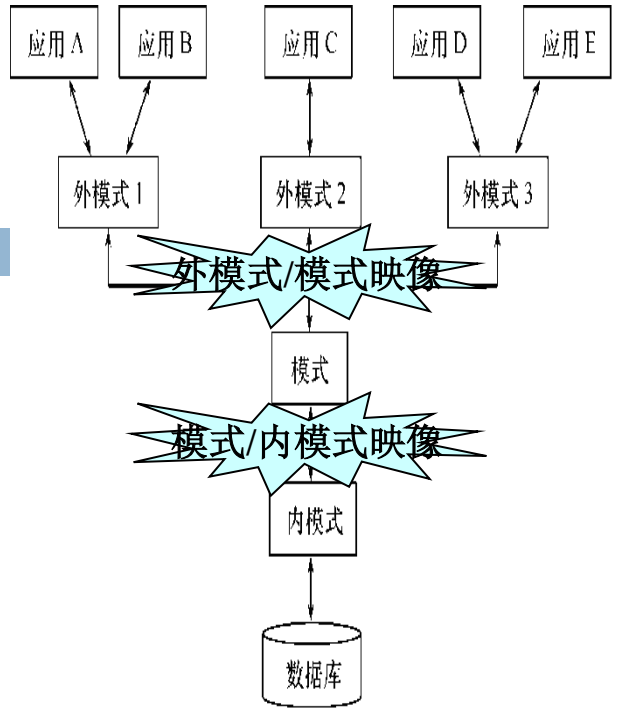


数据库三级模式小结

- 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

- 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

1.1 数据库系统概述

1.2 数据模型

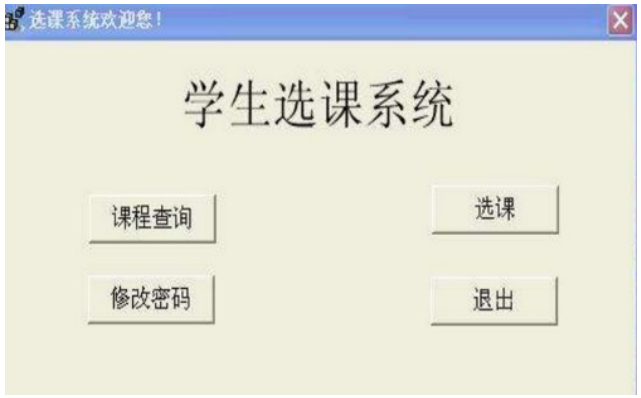
1.3 数据库系统结构

1.4 数据库系统的组成

1.5 小结

1.4 数据库系统的组成

- 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）

- 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
  - (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

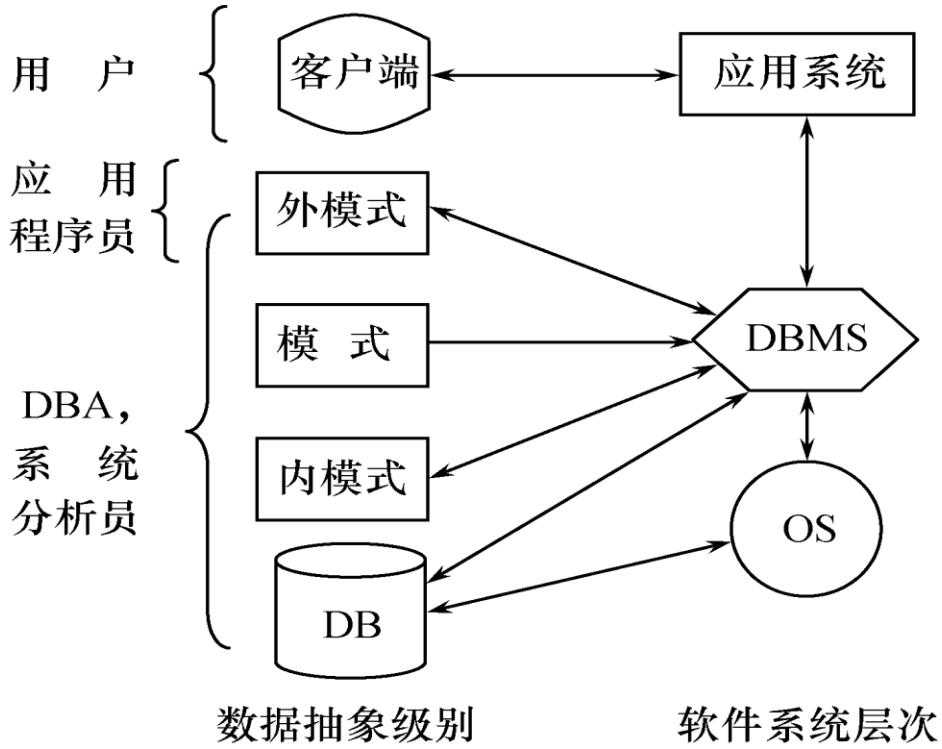
三、人员

- 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

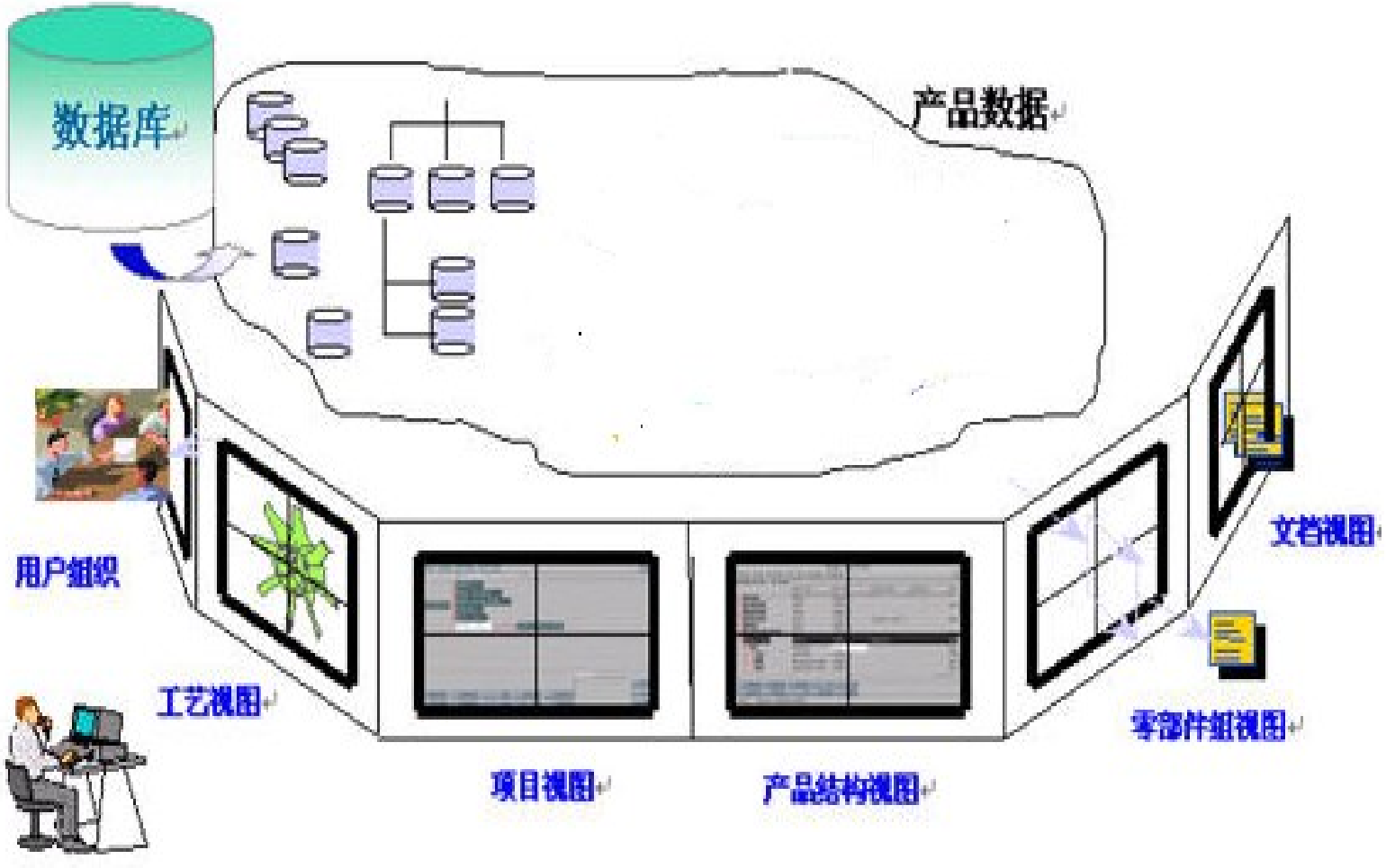
人员（续）

- 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1.17 各种人员的数据视图



人员（续）



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
- 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织，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

- 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

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

1.5 小结

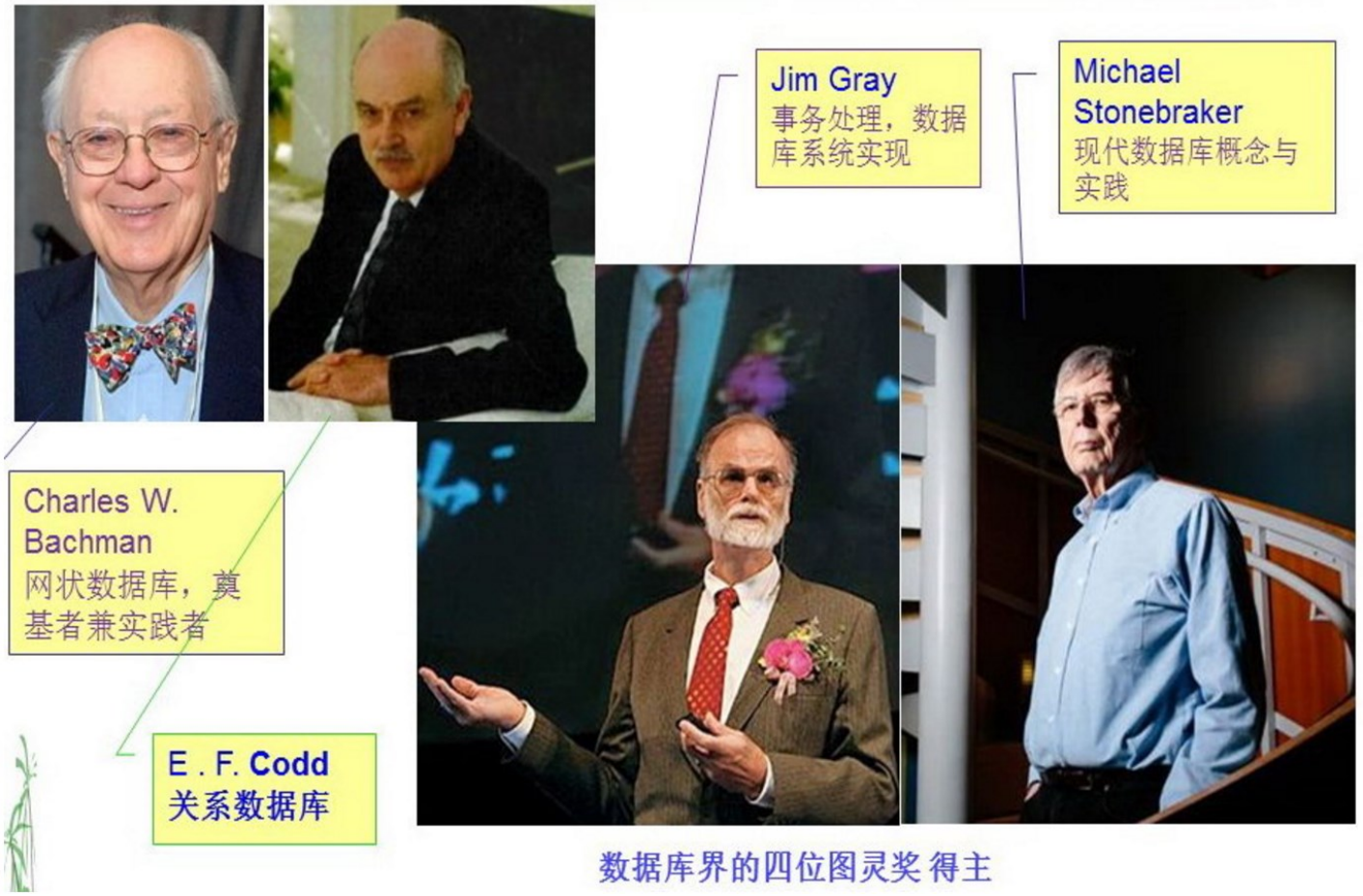
- 数据库系统概述
  - 数据库的基本概念
  - 数据管理的发展过程
- 数据模型
  - 数据模型的三要素
    - 数据结构、数据操作、数据的完整性约束条件
  - 概念模型， E-R 模型
  - 三种主要数据库模型
    - 层次模型、网状模型、关系模型

小结(续)

- 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

课外阅读：数据库界的四位图灵奖得主



数据库界的四位图灵奖 得主

作业（P34页）

第五版教材，提交方式（书面作业，请使用作业本）

1. 试述数据、数据库、数据库管理系统，数据库系统的概念

9. 试述数据模型的概念、数据模型的作用和数据模型的3个要素

14. 试述关系数据库的特点

15. 试述数据库系统的三级模式结构，并说明这种结构的优点是什么？

17. 什么叫数据与程序的物理独立性？什么是数据与程序的逻辑独立性？为什么数据库系统具有数据与程序的独立性？