第十三章 数据库技术新发展

本章内容

- 数据库技术发展的阶段
- 数据库系统发展的特点
  I. 数据模型：对象关系模型，XML模型等
  II. 与相关技术的结合：分布式数据库，对象关系数据库等
- 数据管理技术的发展趋势

数据库技术的发展

数据模型及其发展历程-- 软件学报2019年30（1）

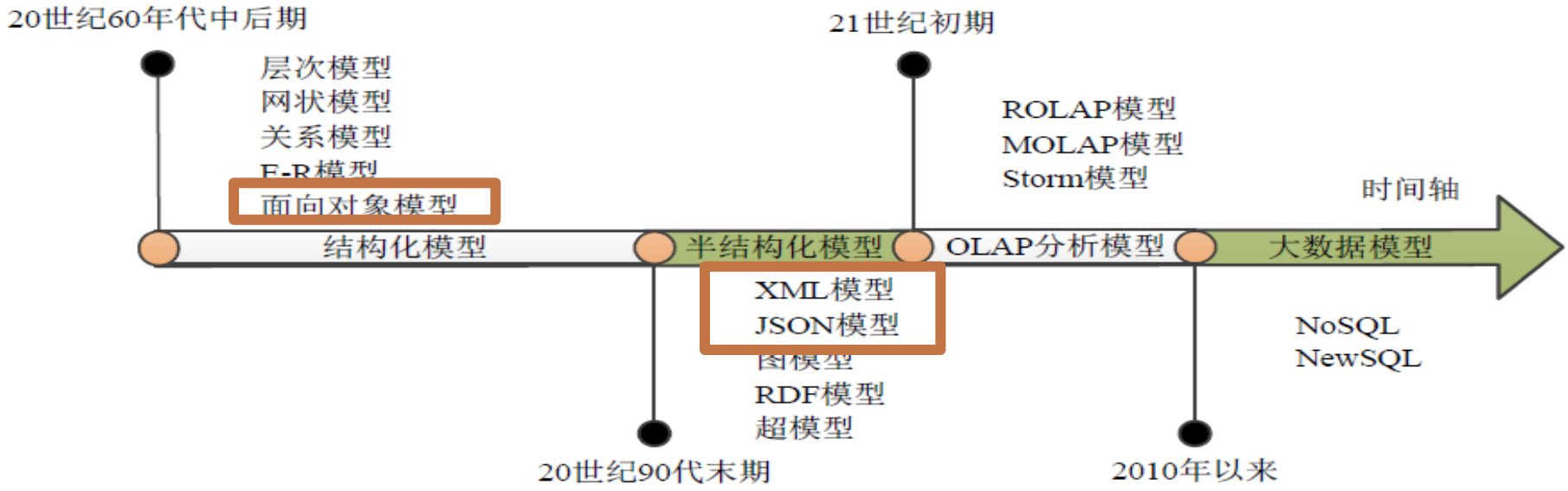
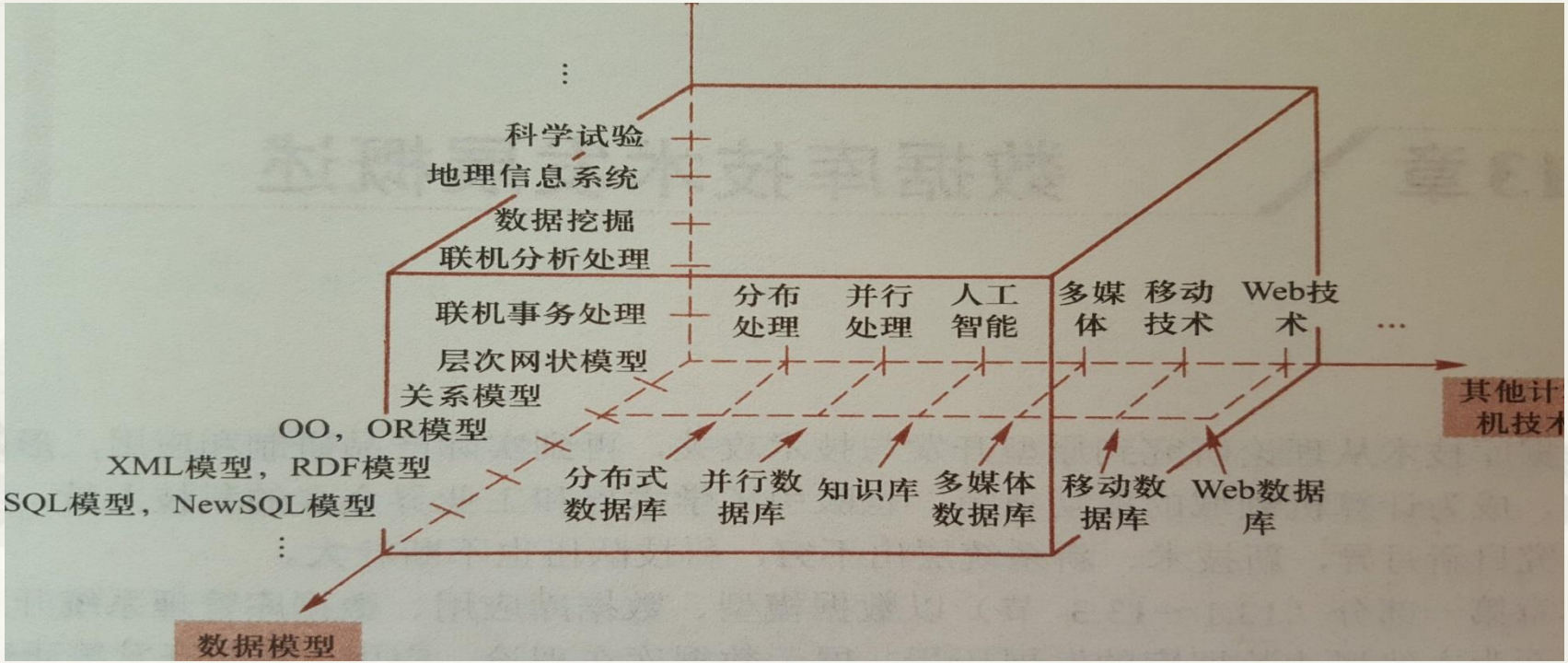


Fig.1 Timeline for the development of data models

图 1 数据模型的发展时间轴

数据库系统发展图


科学试验
地理信息系统
数据挖掘
联机分析处理
联机事务处理
层次网状模型
关系模型
OO，OR模型
XML模型，RDF模型
SQL模型，NewSQL模型
数据模型
分布处理
并行处理
人工智能
多媒体
移动技术
Web技术
分布式数据库
并行数据库
知识库
多媒体数据库
移动数据库
Web数据库

数据库技术的发展——第一代

* 格式化数据模型
  * 层次IBM（IMS）、网状数据库系统(DBTG)
  * 数据库系统的先驱

缺点：

编程繁琐

可移植性差

数据逻辑独立性差

数据库技术的发展——第一代

- 共同的特点
  - 支持三级模式的体系结构
    - 外模式、模式、内模式
    - 具有物理独立性以及一定的逻辑独立性
  - 用存取路径来表示数据之间的联系
  - 独立的数据定义语言
    - 模式的修改很不方便
  - 导航的数据操纵语言
    - 要什么，怎么做
    - 嵌入高级语言

数据库技术的发展——第二代

- 关系数据库系统（IBM公司San Jose 研究员E.F.Codd提出的）
  - 以关系数据模型为基础
  - 关系操作、SQL语言
    - 交互、嵌入式形式
    - 非过程化
  - 数据完整性管理

数据库技术的发展——第二代

* 特点
  * 数据结构单一
    * 扁平的二维表
    * 用关系表达数据、数据间的联系
  * 以关系代数为基础的操作
  * 数据独立性强，物理存储和存取路径的透明性
  * 非过程化的数据库语言，语意化描述

数据库技术的发展——第三代

- 新的应用领域的需求，对关系模型提出了强有力的挑战，推动数据库技术的研究与发展
- 新的应用→新的数据类型
- 新的技术→新的数据库系统

数据库技术的发展——第三代

- 新的需求
  - 非格式化（非结构化）数据的支持
  - 存储和处理复杂对象
  - 支持复杂数据类型
  - 对象管理
  - 大对象的存取和处理
  - 数据库语言与程序设计语言的一体化
  - 长事务和嵌套事务的支持

数据库技术的发展——第三代

* 传统数据库系统的局限
  * 面向机器的语法数据模型
    * 强调数据的高度结构化，语义表示能力较差
    * 无法表示客观世界的复杂对象以及内在联系
  * 数据类型的简单、固定
  * 数据结构与行为分离
  * 被动响应
  * ……

数据库技术的发展——第三代

- 定位
  - 关系数据库系统较适于OLTP的应用
- 新一代的数据库系统应该适应新的需求
  - 支持数据管理、对象管理、知识管理
  - 保持、继承第二代数据库的技术
  - 必须是开放型的系统

数据库系统发展的特点
--数据模型的发展

* 面向对象数据模型
* 对象关系数据模型：关系数据库与对象数据库的结合。

对象关系数据库—简介

- 复杂类型的引入
  - 域可以是原子的，也可以是关系的
  - 关系中包含关系
  - 按照应用的需求直接描述对象
  - 违背 1NF
- 考察具有多值依赖的关系

flat_doc(title,author,pub-name,pub-branch, keyword)

对象关系数据库——举例

title	*author*	*pub-name*	*pub-branch*	*keyword*
Compilers	Smith	McGraw-Hill	New York	parsing
Compilers	Jones	McGraw-Hill	New York	parsing
Compilers	Smith	McGraw-Hill	New York	analysis
Compilers	Jones	McGraw-Hill	New York	analysis
Networks	Jones	Oxford	London	Internet
Networks	Frick	Oxford	London	Internet
Networks	Jones	Oxford	London	Web
Networks	Frick	Oxford	London	Web

* title→→author
* title→→pub-name,pub-branch
* title→→keyword

对象关系数据库——关系数据模型的做法

* R1(title, author)
* R2(title, keyword)
* R3(title, pub-name, pub-branch)

属于4NF

问题

* 需要通过join才能获取信息
* 无法直接表达属性间的关系
* 具有大量的冗余数据

对象关系数据库——嵌套关系

- 以嵌套关系的形式，更能表示元组与实体之间的关系

title	*author-set*	*publisher*	*keyword-set*
		(*name*, *branch*)	
Compilers	{Smith, Jones}	(McGraw-Hill, New York)	{parsing, analysis}
Networks	{Jones, Frick}	(Oxford, London)	{Internet, Web}

- 复合属性（非原子）
  - author-set、publisher、keyword-set

对象关系数据库--复杂类型和面向对象

- 对SQL的扩展主要体现在对SQL在复杂数据类型的支持上
  - 集合体类型和大对象类型
    - 嵌套关系
  - 结构类型
    - 嵌套记录、组合记录
  - 面向对象
    - 对象表识、引用
  - 继承

对象关系数据库 复杂类型——大对象类型

- clob: Character large objects

  book-review clob(10KB)

  存贮 长字符串数据

- blob: binary large objects

  image *blob(10MB)*

  movie *blob (2GB)*

  存贮音频，图像数据

对象关系数据库—自定义类型

- UDT（User Defined Type）
- 允许用户定义带有自身行为说明和内部结构的用户定义类型.

对象关系数据库—自定义类型

* 例如：出版商类型，书类型

```
create type Publisher as
        (name          varchar(20),
         branch        varchar(20))
        method publisherTel () returns char(8);
create type Book as
        (title         varchar(20),
         author-array  varchar(20) array [10],
         pub-date      date,
         publisher     Publisher,
         keyword-set   setof(varchar(20)))
```

```
Create method
publisherTel () returns
char(8) for Publisher
Begin
 ….
End;
```

对象关系数据库—创建自定义类型的关系表

* 创建教材关系表textbook

Create table textbook of book (primary key (title, pub-date))

* Text*book*：单列关系

和类型同名的构造函数 constructor产生。

对象关系数据库– 自定义类型表的结构

- Select * from textbook
- 输出元组如下：

Book (‘Compilers’,Array [`Smith’,`Jones’], date ‘2002-11-20’,Publisher (`McGraw-Hill’,`New York’), Set (`parsing’,`analysis’))

这里的book 是一种类型

对象关系数据库 复杂类型——继承

* 假设已创建person的类型

  create type *Person* (*name* varchar(20), *address* varchar(20))

* 用继承的方法定义学生和教师

  create type *Student* under *Person*

  (*degree* varchar(20), *department* varchar(20))

  create type *Teacher* under *Person*

  (*salary* integer, *department* varchar(20))

对象关系数据库– 无名属性类型

```
create table books
        (title varchar(20),
         author-array varchar(20) array[10],
         pub-date date,
         publisher row (name varchar(20), branch
  varchar(20)),
        keyword-list setof(varchar(20)))
```

注意：这里的Book 是用户创建的表，而非类型。

对象关系数据库 -- 复杂类型的查询

- 具有复杂类型的查询，是对SQL语言的一个扩展，SQL99支持这一扩展
- 找出每本书的书名及出版社的名称

```
select  title, publisher.name
from  books
```

访问结构的属性

对象关系数据库--复杂类型的查询

* 找出某一本书的作者(假定已知有三名作者)

```
select author-array[1], author-array[2], author-
array[3]
from books
where title = `Database System Concepts'
```

对象关系数据库-- 插入元组

- 向关系*books*中插入元组

```
insert into books values
        (`Compilers’,
        array[`Smith’,`Jones’],
        Publisher(‘McGraw Hill’,`New York’ ),
        set(`parsing’,`analysis’))
```

- 这里的book是用create table创建的表

总结对象关系数据库

* 引入了面向对象的特点：继承，指针，。。。
* 复杂数据类型，用户自定义类型，。。。
* 保持关系基础，扩展模型能力
* 向上兼容：与关系模型数据库兼容

关系模型与对象关系模型

MySQL 和 PostgreSQL 对比

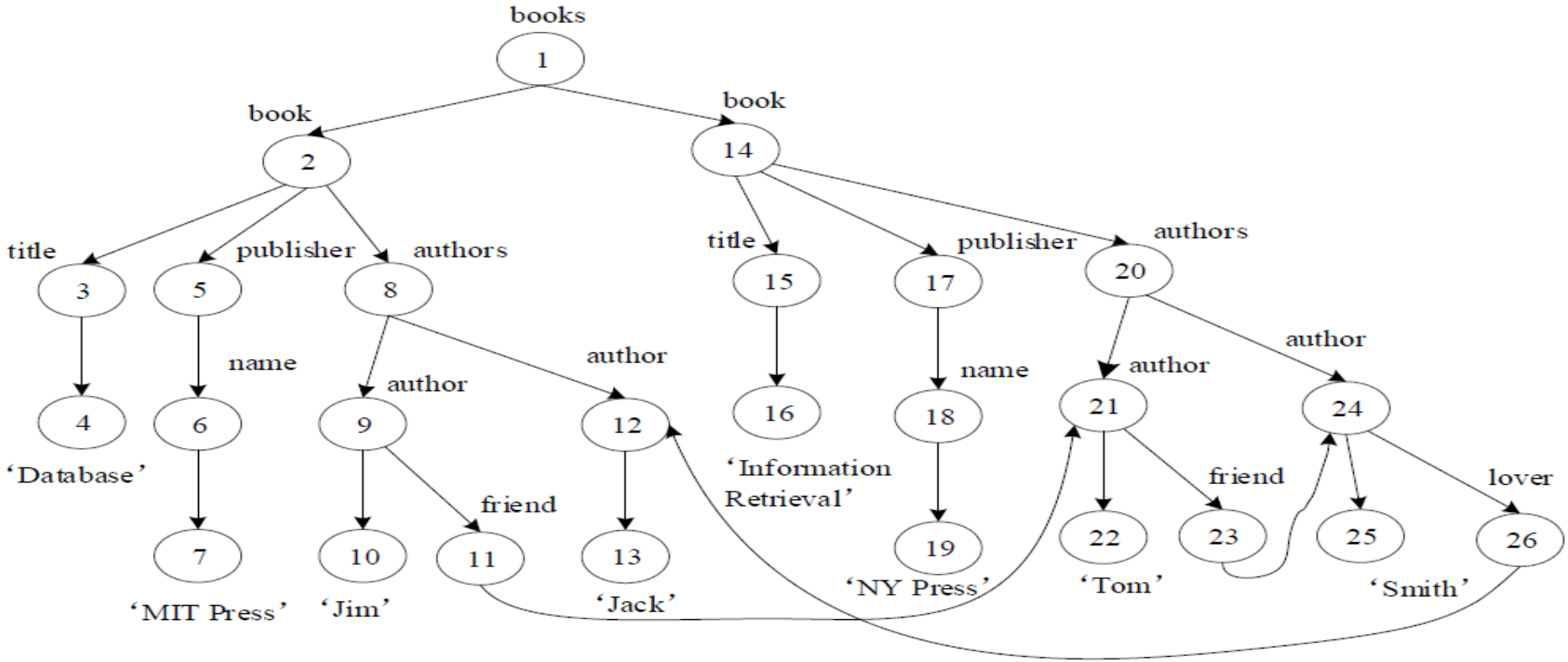
	MySQL	PostgreSQL
概念	关系型数据库管理系统	对象-关系型数据库管理系统
运行环境	Windows, Mac OS X, Linux, BSD, UNIX, z/OS, Symbian, AmigaOS	Windows, Mac OS X, Linux, BSD, UNIX, z/OS, Symbian, AmigaOS
可扩展性	不支持	支持
接口	提供 GUI	提供 GUI
备份工具	Mysqldump, XtraBackup	在线备份
物化视图	提供临时表,不提供物化视图	提供临时表,且提供物化视图
数据对象	不支持	支持

数据库系统发展的特点--数据模型的发展

- **XML数据模型**：（extended markup language）可扩展的标记语言.
- XML是W3C在1998年制定的一项标准,用于网上数据交换.
- XML特点:

1. 基于内容的描述
2. 可扩展性
3. 自描述性
4. 数据与显示分离
5. 简洁性

XML 模型实例

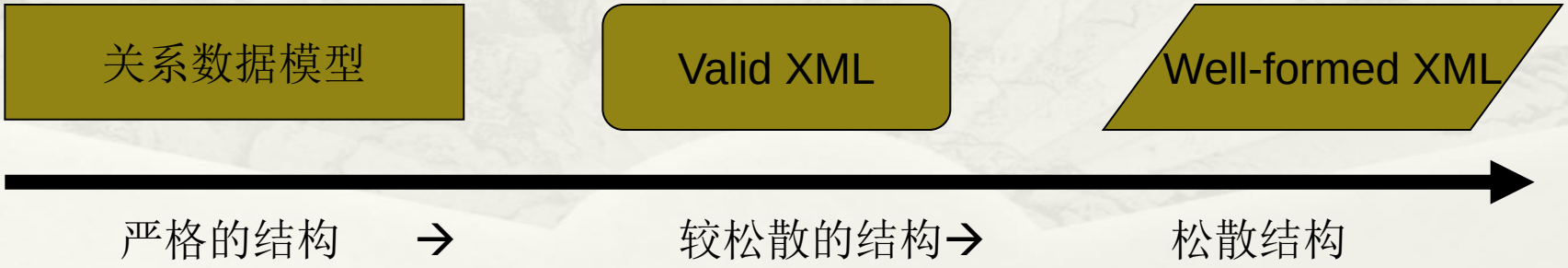

books
1
book
2
book
14
title
3
publisher
5
authors
8
title
15
publisher
17
authors
20
4
'Database'
name
6
author
9
author
12
16
'Information Retrieval'
name
18
author
21
author
24
7
'MIT Press'
10
'Jim'
friend
11
13
'Jack'
19
'NY Press'
friend
22
'Tom'
23
25
'Smith'
lover
26

XML 模型：树模型

- 元素节点:该类型的节点为XML 树中的标签;
- 属性节点:该类型的节点为XML 树中标签相关的属性.不同于元素节点,属性节点不是嵌套的,即,属性节点不能有任何子元素.相同的属性节点不能嵌套在同一个元素节点中,并且属性节点是无序的;
- 值节点(叶子节点):该类型的节点为标签的值;
- 有向边描述了各类型节点之间的关系.
- XML 图模型允许用户引入ID/IDREF 属性来对树模型进行扩展,其中,ID 属性唯一地标示了某个元素;IDREF 引用其他被ID 属性标示的元素,构成一个有向无环图.

XML数据模型：两种类型的XML文档

- *Well-Formed XML* :符合一定规范的文档,使用自己的标签。
- *Valid XML*符合预先定义的格式 DTD, or XML schema



符合一定规范的XML文档

```
<?xml version = “1.0” standalone = "yes" ?>
<BOOKS>
   <BOOK>
      <TITLE>’database’</TITLE>
      <AUTHORS><AUTHOR>Joe</AUTHOR><AUTHOR>Jack</AUTHOR>
      </AUTHORS>
      <PUBLISHER><NAME>MIT Press</NAME></PUBLISHER>
   </BOOK>
   <BOOK> … </BOOK>
</BOOKS>
```

- 只有一个根结点
- 具有开闭标记<tag></tag>，正确嵌套
- 在element 中属性attributes 唯一

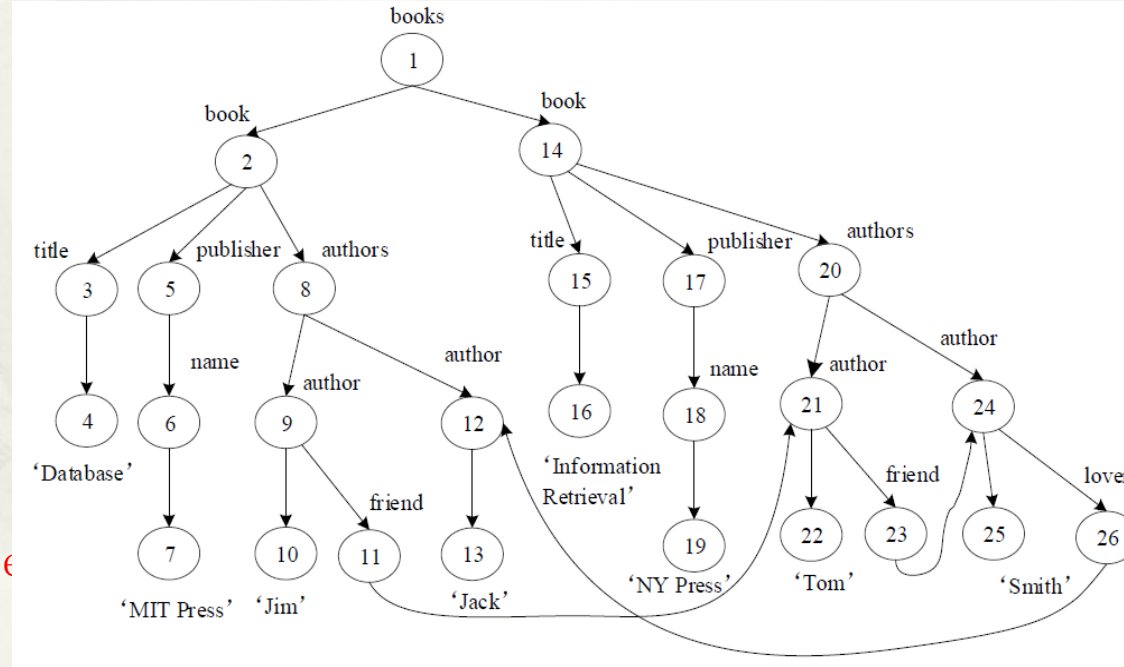
有效的XML文档

* 符合预先定义的：

1. DTD
2. XML Scheme

DTD 举例

```
<!DOCTYPE BOOKS [
  <!ELEMENT BOOKS (BOOK*)>
  <!ELEMENT BOOK
  ( AUTHORS+,PUBLISHER*)>
  <!ATTLIST BOOK TITLE CDATA )>
  <!ELEMENT AUTHORS (AUTHOR+)>
  <!ELEMENT PUBLISHER (NAME)>
    <!ELEMENT AUTHOR(#PCDATA)>
    <!ATTLIST AUTHOR nameID ID
  #required, <!ATTLIST AUTHOR frie
  IDREFs>
     <!ATTLIST AUTHOR lover IDREF
  #implied >
```



DTD 应用

<?xml version = “1.0” standalone = “no” ?>

1. 把DTD放在文档前面
2. 把DTD放在某一目录下面，在XML文档里包含：

<!DOCTYPE root SYSTEM “DTDname.dtd”>

XML Scheme（省略）

DTD 的缺陷

* Element, attribute 没有类型
* ID和IDREFS 都没有类型
* Subelement 集合的顺序难于规定。

XML scheme 解决了这些问题，较复杂。

SQL + XML

创建表：

```
Create table students (Sid char(8) primary key, Name
varchar(30), Dept char(4), Hobbies XMLTYPE)
```

插入数据：

```
Insert into students values (‘10403050’,’LI Hong’,’CS’,
’<Hobbies><English level>6</English
level><piano>8</piano><basketball>well</basketball></Hobbies>’
);
```

SQL XML (cont.)

Select * from “students”;

SID	name	dept	Hobbies
10403050	LI Hong	CS	<hobbies><English level>6</English level><piano>8</piano><basketball>well</basketball></hobbies>

Select XMLELEMENT(“students”, XMLFOREST(name as “StudentName, dept as “Department”)) from “students”;

XMLELEMENT(“students”, XMLFOREST(name as “StudentName, dept as “Department”

<students><StudentName>LI Hong</StudentName>
<Department>cs</Department></students>

数据库系统发展的特点--数据模型的发展

- **RDF数据模型** （w3c提出的资源描述框架）：一种描述WEB资源的标记语言，结构是（主语，谓词，宾语）构成的三元组
- 主语：网页的URL
- 谓语：属性 （网页的标题，作者等）
- 宾语：具体的值或另一个对象。

数据库系统发展的特点--数据模型的发展

- JASON 模型（JavaScript object notation）：易于读写的轻量级数据表示形式。
- 两种结构.

1. 对象:包含一系列非排序的键-值对,一个对象以“{”开始,并以“}”结束.每个键-值对之间使用“:”区分;
2. 数组:一个数组是若干值的集合,一个数组以“[”开始,并以“]”结束.数组成员之间使用“,”区分.

JSON 样例

- JSON vs. XML

```
{
   "firstName": "Green",
   "lastName": "Jim",
   "sex": "male",
   "age": 33,
   "address":
   {
      "streetAddress": "21 2nd Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10021"
   },
   "phoneNumber":
   [
      {
        "type": "home",
        "number": "212 555-1234"
      },
      {
        "type": "fax",
        "number": "646 555-4567"
      }
   ]
}
```

(a) JSON 文档举例

```
<firstName>John</firstName>
<lastName>Smith</lastName>
<sex>male</sex>
<age>25</age>
<address>
        <streetAddress>21 2nd Street</streetAddress>
        <city>New York</city>
        <state>NY</state>
        <postalCode>10021</postalCode>
</address>
<phoneNumber>
        <type>home</type>
        <number>212 555-1234</number>
</phoneNumber>
<phoneNumber>
        <type>fax</type>
        <number>646 555-4567</number>
</phoneNumber>
```

(b) 转换后的 XML 文档

数据库技术与相关技术的结合

* 分布处理技术结合→ 分布式数据库系统
* 并行技术结合→ 并行数据库系统
* 特定应用领域→ 数据仓库
* 特定应用领域→工程数据库，统计数据库
* 特定应用领域→空间数据库

分布式数据库简介

- 数据分布。
- 分布的数据是相互关联的。
- 数据由分布式数据库管理系统软件统一管理。

分布式数据库与分布式处理

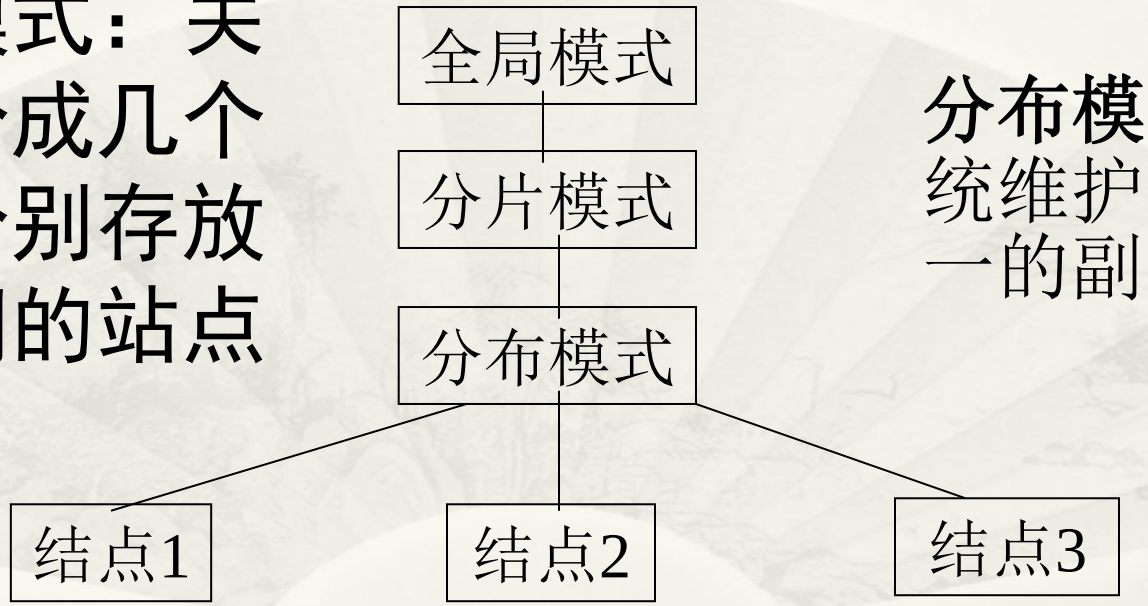
- 分布式处理不一定要分布式数据库，但分布式数据库必须要分布式处理。
- 分布式处理和分布式数据库都需要网络来把各个分布在不同地点的计算机连接起来。

分布式数据库的定义

* 分布式数据库是由一组数据组成，这些数据分布在计算机网络的不同站点上，逻辑上是属于同一个系统的。网络的每个站点具有**独立处理能力**，可以执行**局部应用**，同时也能通过网络执行全局应用。

分布式数据库系统结构

* **分片模式：关系被分成几个片段分别存放在不同的站点上。**



分布模式：系统维护几个同一的副本

水平分割：例如银行账户***account*** Relation

branch-name	*account-number*	*balance*
Shanghai	A-305	500
Shanghai	A-226	336
Shanghai	A-155	62

$account_1 = \sigma_{branch\text{-}name=\text{“Hillside”}}(account)$

branch-name	*account-number*	*balance*
Beijing	A-177	205
Beijing	A-402	10000
Beijing	A-408	1123
Beijing	A-639	750

$account_2 = \sigma_{branch\text{-}name=\text{“Valleyview”}}(account)$

垂直分割： *employee-info* Relation

branch-name	*customer-name*	*tuple-id*
shanghai	Lowman	1
Shanghai	Camp	2
Beijing	Camp	3
Beijing	Kahn	4
Shanghai	Kahn	5
Beijing	Kahn	6
Beijing	Green	7

account number	*balance*	*tuple-id*
A-305	500	1
A-226	336	2
A-177	205	3
A-402	10000	4
A-155	62	5
A-408	1123	6
A-639	750	7

混合分割

既有水平分割，又有垂直分割。

数据分片(割)的原则

* 完备性：不能丢失数据
* 可重构性：完全等价
* 不相交性：在水平分割时，没有重复的元组。

数据分布策略

- 数据分布的目的：提高访问的局部性。
- 数据分布的方式：
  - 划分式
  - 全重复式：
  - 部分重复式

分布式数据库系统：事务管理和并发控制

* 事务管理：

事务被分成了几个子事务，需要全部提交：两步提交协议和三步提交协议

* 并发控制

分布式数据库系统并发控制的特点

- ✓ 分布式数据库系统支持多副本的特点
- ✓ 事务的分布执行，封锁易引起全局死锁

分布式数据库系统：查询优化

* 集中式数据库系统查询处理的开销

I/O代价+CPU代价

* 分布式数据库系统查询处理的开销

I/O代价+CPU代价+通信代价

解决通信问题：半连接

分布式数据库系统查询优化举例

S(SNo, CITY) 在场地A 10^4 个
P(PNo,COLOR) 在场地B 10^3 个
SP(SNo,PNo) 在场地A 10^6 个

```
SELECT  S.SNo
FROM  S,P,SP
WHERE SP.PNo =
P.PNo AND
S.CITY='BeiJing'
AND P.Color='red'
```

策略1：把关系P 传到场地A，在A地做16.7分钟
策略2：把关系S，SP传到场地B，在B地做**2.8小时**。
策略3：2.3天
策略4：20秒
策略5：16.7分钟 （北京供应商装运单传B）
策略6：1秒 （把红色零件传到场地A）

数据管理技术的发展趋势

- 数据类型变化：多样化，异构化，半结构化，非结构化，流数据，不确定数据，嘈音数据等
- 数据处理变化：多媒体数据的快速获取，实时性，语义性要求→ 挖掘算法缺乏可扩展性，缺乏对异构数据的高效分析算法，缺乏大规模知识库的支持等。
- 计算机硬件技术：多核，大内存，集群，云计算平台等

总结

- 数据库技术发展的三个阶段：
- 数据模型（半结构化模型）
- 新技术内容（对象关系数据库，分布式数据库）
- 应用领域
- 核心是数据管理