万常选 编著

XML数据库技术

清华大学出版社

XML数据库技术

万常选 编著

清华大学出版社
北京

内 容 简 介

随着大量 XML 数据的出现，如何有效地存储、管理和查询这些 XML 数据已成为一个值得研究的重要课题。目前，XML 数据库技术是数据库领域的研究热点。

本书是在作者博士论文的基础上扩充而成的，共分 6 章。第 1 章介绍有关的基础知识，包括 XML、DTD、XML 模式、XPath 和 XQuery 等；第 2 章在介绍了 XML 数据的编码方案之后，对纯 XML 数据库的存储结构、索引技术和事务管理进行了综述；第 3 章讨论了基于关系的 XML 数据库技术，首先对各种映射 XML 数据到关系存储的方法进行了综述，然后重点讨论了新提出的 X-RESTORE 索引结构、关系存储模式以及查询中间件；第 4 章讨论了 X-RESTORE 下的 XML 查询的计算策略和转换 XPath 路径表达式到 SQL 查询的算法；第 5 章讨论了 XML 结构连接技术，包括各种计算祖先/后裔关系（含双亲/孩子关系）结构连接的直接归并结构连接算法、基于缓存的归并结构连接算法和 twig 模式结构连接算法，以及计算文档位置关系的结构连接算法；第 6 章讨论了 XML 的查询优化技术，主要包括查询最小化、视图查询、估算查询结果大小和选择结构连接顺序等。

本书可作为计算机及相关专业研究生或高年级本科生的教材，也可作为从事 XML 数据库研究或应用开发人员的参考资料。

图书在版编目(CIP)数据

XML 数据库技术/万常选编著. —北京：清华大学出版社，2005. 1
ISBN 7-302-10375-5

Ⅰ. X…　Ⅱ. 万…　Ⅲ. 可扩充语言，XML—程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2005)第 004954 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：焦　虹
文稿编辑：张为民
封面设计：常雪影
印 刷 者：国防工业出版社印刷厂
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：17　**字数**：401 千字
版　　次：2005 年 1 月第 1 版　2005 年 1 月第 1 次印刷
书　　号：ISBN 7-302-10375-5/TP・7058
印　　数：1～3000
定　　价：28.00 元

序

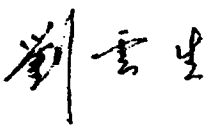
数据库与网络是计算机科学技术领域中近年来发展最快、最受人关注、应用最广的两个分支学科。数据库技术与网络技术已成为当今社会最重要、与社会各界关系最为密切的技术之一，它们共同构成了支持现代信息社会的技术基础。

在数据库领域中，30 多年来人们在数据的表示（包括逻辑模型和物理模型）、数据的处理（包括存储和操作）、数据的使用（包括传输与控制）等方面做了大量创造性的工作，取得了卓越的成绩。数据库系统从“面向记录”的第一代（层次型、网状型）系统，到“面向值”的第二代（关系型、逻辑型）系统，再到“面向对象”的第三代（扩展关系型、语义与函数型、对象关系型、永久对象型等）系统，发展到今天，“数据库园”中已是百花争艳、万紫千红。现代数据库技术发展的一个重要方向，是经典数据库技术与其他一种或多种现代信息/数据处理技术，如面向对象、多媒体、智能/规则处理、时序和实时处理、网络和 Internet/WWW 等的“有机融合”（seamless integration）。其中，与 WWW 的结合是最为必要且迫切的。

Internet/WWW 的出现是网络技术发展史上的一个里程碑。它促使了大量 Internet/WWW 数据的涌现与使用，同时也促进了数据库和 Internet/WWW 两种技术的结合。在当今信息社会，人们迫切希望和要求建立基于 Internet 的数据库系统，盼望能提供 4Wh-ever（Whatever、Whenever、Wherever、Whoever）式的数据处理服务。XML 数据库的应运而生，正是 Internet/WWW 和数据库这两种技术相结合的产物。

面对社会的挑战，广大数据库研究、开发和应用工作者在 Internet/WWW 数据库或 XML 数据库领域中进行了大量、广泛而深入的探究，取得了令世人瞩目的成绩。有关方面的文章如雨后春笋，但相对而言，这方面的专门书籍则要少得多。本书是经过作者多年辛勤工作（尤其是在攻读博士学位期间，远离家人，效“颜回之乐”，作了深入细致的开拓性研究，阅读了大量国内外相关文献），在博士论文的基础之上充实完善而成的，为“数据库沧海”添上了一粟。

当今的信息时代正是数据库的春天，我们广大园丁的辛劳与汗水必将换来数据库百花园中争奇斗艳、姹紫嫣红的美丽景色。



2004 年 11 月

前　言

网络技术的飞速发展改变了人们的学习、生活和工作方式，拓宽了人们获得知识和信息的途径，并且也改变了人们的思维方式。XML 就是这样一种迅速发展的技术。

XML 起初主要是为了增强应用程序从 Web 上获得文档的解释和操作能力而产生的。然而，从数据库的角度来看，XML 从另一个方面提出了一个令人兴奋的可能性：查询这些文档的内容。随着大量 XML 数据的出现，如何有效地存储、管理和查询这些 XML 数据，就成为一个值得研究的重要课题。

对 XML 数据库系统的研究主要有两种途径：一是纯(native)XML 数据库系统，它是为 XML 数据量身定做的数据库。它的优点是充分考虑到 XML 数据的特点，以一种自然的方式来处理 XML 数据，能够从各个方面较好地支持 XML 的存储和查询，但是，纯 XML 数据库要走向成熟还有很长的路。二是 XML 使能(XML-enabled)数据库系统，它是在已有的关系数据库系统或面向对象数据库系统的基础上扩充相应的功能，使其能够胜任 XML 数据的处理。目前，XML 使能数据库的研究主要是基于关系数据库的。这种方法的优点是可以充分利用已有的非常成熟的关系数据库技术，集成现有的大量存储在关系数据库中的商用数据。

对 XML 数据库系统的研究主要集中在 6 个方面：一是对 XML 的查询数据模型、查询语言和查询代数等进行研究；二是对 XML 数据的编码方案和索引结构进行研究；三是对纯 XML 数据库系统进行研究，包括存储结构、索引技术、查询技术和事务管理等；四是对基于关系的 XML 使能数据库进行研究，包括 XML 数据的关系存储模式、XML 查询技术和查询算法，以 XML 文档发布关系数据的技术，以及通过中间件实现以 XML 格式和 XML 查询语言查询关系数据并进行异构数据源的信息集成等；五是对 XML 的查询技术、查询算法，XML 查询的包含与等价、树模式查询最小化，以及查询优化技术等进行研究；六是对 XML 查询测试数据集、测试查询范例进行研究。

目前，XML 数据库技术是数据库领域的研究热点，近年来国际重要的数据库学术会议都收录了大量关于 XML 数据库方面的论文，例如，2004 年国际 VLDB(Very Large Data Bases)会议录用了 79 篇研究论文，其中专门为 XML 开辟了 4 个专题(Session)，录用研究论文 16 篇，占 20%；2004 年国际 ACM SIGMOD(Association for Computing Machinery，Special Interest Group on Management of Data)会议共录用 69 篇研究论文，其中专门为 XML 开辟了 4 个专题，收录论文 13 篇，占 19%。

本书共分 6 章。第 1 章是有关的基础知识，包括 XML、DTD、XML 模式、XPath 和 XQuery 等；第 2 章在介绍了 XML 数据的编码方案之后，对纯 XML 数据库的存储结构、索引技术和事务管理进行了综述；第 3 章讨论了基于关系的 XML 数据库技术，首先对各种映射 XML 数据到关系存储的方法进行了综述，然后重点讨论了我们提出的 X-RESTORE 索引结构、关系存储模式以及查询中间件；第 4 章讨论了 X-RESTORE 下

的 XML 查询的计算策略和转换 XPath 路径表达式到 SQL 查询的算法；第 5 章讨论了 XML 结构连接技术，包括各种计算祖先/后裔关系（含双亲/孩子关系）结构连接的直接归并结构连接算法、基于缓存的归并结构连接算法和 twig 模式结构连接算法，以及计算文档位置关系的结构连接算法；第 6 章讨论了 XML 的查询优化技术，主要包括查询最小化、视图查询、估算查询结果大小和选择结构连接顺序等。

本书是在作者博士论文的基础上扩充而成的，首先要感谢我的导师刘云生教授，他对本人在博士研究生阶段的学习和研究工作等进行了悉心指导和无倦教诲。在本书撰写完成之后，刘云生教授又仔细审阅了全书，提出了许多宝贵的意见，并为本书作序。

本书在编写过程中，作者参阅了许多专家的学术论文，特别是香港科技大学陆宏均教授、中国人民大学孟小峰教授等在该领域的大量研究成果，也得到了中国计算机学会数据库专业委员会主任王珊教授的热情鼓励和帮助。在此表示衷心感谢。

本书是作者所在课题组全体师生徐升华、刘喜平、林大海、凌传繁、钱忠胜、张治、唐志涌、江腾蛟等共同的成果。课题的研究得到了江西省教育厅科技项目（赣教计字[2001]387 号、赣财教[2003]73 号）和江西省自然科学基金（0411009）的资助，对江西省教育厅和江西省科技厅的资助表示衷心感谢。在成书的过程中，刘喜平硕士生对全书的初稿进行了阅读和修改，付出了辛勤的劳动，在此也表示感谢。

本书适合作为计算机及相关专业研究生或高年级本科生的教材，也可作为从事 XML 数据库研究或应用开发人员的参考资料。

尽管我们十分努力，以求本书尽善尽美，但是由于作者的水平有限，加之 XML 数据库技术目前还不成熟，每年都会有大量的研究成果出现，疏漏与谬误之处在所难免，恳请专家、同仁及广大读者批评指教。

万常选

2004 年 12 月

目　录

第1章 绪 论

网络技术的飞速发展改变了人们的学习、生活和工作方式，拓宽了人们获得知识和信息的途径，并且也改变了人们的思维方式。XML就是这样一种迅速发展的技术。

1.1 XML与模式

1.1.1 XML简介

XML(extensible markup language，可扩展的标记语言)是由World Wide Web Consortium(W3C)的XML工作组定义的。这个工作组是这样描述该语言的[1]："XML是SGML(standard generalized markup language，标准通用标记语言)的子集，其目标是允许普通的SGML在Web上以目前HTML(hypertext markup language)的方式被服务、接收和处理。XML被设计成易于实现，且可在SGML和HTML之间互相操作。"

XML是一套定义语义标记的规则，这些标记将文档分成许多部件并对这些部件加以标识。它不像HTML或格式化程序。这些语言定义了一套固定的标记，用来描述一定数目的元素。XML是一种元标记语言，用户可以定义自己需要的标记。这些标记必须根据某些通用的原理来创建，XML标记描述的是文档内容的结构和含义，而不是描述页面元素的格式化。可用样式单为文档增加格式化信息。文档本身只说明文档包括什么标记，而不是说明文档看起来是什么样的。

相对于HTML而言，XML具有很多优点：

(1) XML是自描述的。XML的最大能量来源于它不仅允许定义自己的一套标记，而且这些标记不必局限于对于显示格式的描述。XML允许根据各种不同的规则来制定标记，比如根据商业规则，根据数据描述甚至根据数据关系来制定标记。XML实现了用定义它们自己的标记集来说明文档内容的功能，这些说明的精确度是实现者自己制定的。

(2) XML支持对文档内容的验证。XML文档的结构和内容是由其语法定义的。文档类型定义(document type definition，DTD)就是这类语法的一种，正在形成的还有XML模式。有了模式，就可以方便地验证文档的有效性。

(3) XML允许开发各种不同专业(如音乐、化学、数学等)的特定领域的标记语言。有了这些语言，这个领域的实践者们可以相互自由地交换短文、数据和信息，而不必担心对方是否利用特殊的、专用的软件来创建数据。事实上，目前已经开发出了一些特定领域的标记语言，如MathML(用于数学领域的一种标记语言)。

(4) XML是非专有并易于阅读和编写的。这使得它成为在不同的应用间交换数据的理想格式。XML不是第一种公共文档格式，但它与已有的文档交换格式相比具有很多优点。XML是源文档的最佳格式，因为它允许用最佳的输出格式，例如HTML、PDF(portable document format)和PostScript格式，并格式化应用程序，例如电子数据交换

(electronic data interchange,EDI)。

(5) XML 是基于 W3C 定制的开放标准,从而使得基于 XML 的应用具有广泛性。

(6) 支持高级搜索。因为可以知晓文档内容的结构和含义(根据它的语法规则),所以很容易在 XML 文档中进行搜索。在 Internet 上如果 Web 页是 XML 格式的,则搜索会更高效,而且不仅可以搜索数据,还可以在搜索中加入与数据相关的上下文信息,这样就形成了更精确的搜索机制。

有人说,XML 是下一代 Web 语言;甚至有人说,XML 是 21 世纪的"世界语"。不管是否确切,这些说法都显示出了 XML 的巨大潜力。下面是 W3C 阐述的 XML 的 10 个设计目标[1]:

- XML 应该可以直接用于 Internet;
- XML 应该支持各种应用程序;
- XML 应该与 SGML 兼容;
- 编写处理 XML 文档的应用程序应该很简单;
- XML 中可选特性的数目应该尽可能地少,理想情况是零;
- XML 文档应该便于阅读而且相当清晰;
- XML 的设计应该很快准备好;
- XML 的设计应该正式而且简洁;
- XML 文档应该易于创建;
- XML 标记的简洁性是最不重要的。

目前 XML 在很多方面都有应用。如果需要获得更多的 XML 应用列表,包括每个应用的详细描述,请参见 SGML/XML 的 Web 页(http://www.oasis-open.org/cover/xml.html/#applications)。

XML 的语法十分简单易学。只要了解其中一些简单的规则,就可以轻易上手。虽然它也有一些比较复杂的规则,但是这些规则不难理解。

1. XML 声明

XML 声明必须在文档的第一行,而且其中的字母是区分大小写的。首先声明使用的 XML 版本号,如<? xml version="1.0"? >,虽然 XML 1.0 是目前惟一的版本,但是仍然要声明版本属性。

文字编码声明位于版本属性之后,其形式为 encoding = "UTF-8"(Unicode Transformation Format-8)。文字编码声明指出文档是使用何种字符集建立的,默认值是 Unicode 编码(UTF-8 或 UTF-16)。

独立文件声明位于文字编码声明之后,如 standalone="yes"。独立文件声明使用的属性值可以为 yes 或 no。属性值 yes 表示所有与文件相关的信息都已经包含在文件中,即文件中没有指定外部的实体,也没有使用外部的模式;属性值 no 表示应用程序需要取得文件以外的信息才能完成文件解析。

完整的 XML 声明如下所示:

```
<? xml version="1.0" encoding="UTF-8" standalone="no"? >
```

2. 元素

通常，元素组成了 XML 文档中的大部分内容。元素有名字(即标记名)，也可能有后裔，后裔可能是元素、处理指令、注释、字符数据(CDATA)段或者字符。一个良构的(well-formed，也称为格式正确的)XML 文档必须至少包含一个元素，即文档中必须有根元素。元素由一对标记(即起始标记和终止标记)串行化而成。起始标记的形式是＜标记名＞，终止标记的形式是＜/标记名＞，元素的后裔位于起始标记和终止标记之间。如果元素没有后裔，则称为空元素。空元素也可以用一种速记法来表示，即＜标记名/＞。

XML 中的元素名称是区分大小写的。它必须开始于字母或下划线(_)，后面可跟任意长度的字母、数字、句点(.)、连接符(-)、下划线或冒号。

3. 属性

元素可以用属性来注释。属性通常用来给元素提供所显示内容的额外信息。元素的属性在元素的起始标记中给出，形式为：属性名＝属性值。属性名与元素名有相同的构造规则，属性值必须出现在单引号或双引号中。一个元素可以有任意数目的属性，但是它们的名称必须不同。

4. 处理指令

处理指令通常用来为处理 XML 文档的应用程序提供信息，这些信息包括如何处理文档，如何显示文档等。处理指令可以作为元素的后裔出现，也可以作为文档的顶层结构出现在根元素的前面或后面。处理指令由两部分组成：处理指令的目标或名称、数据或信息。其格式为＜? target data? ＞，目标的构造规则与元素名的构造规则一样。例如，处理指令：

```
<? display table-view? >
```

5. 注释

XML 支持注释。注释可以作为元素的后裔出现，也可以作为文档的顶层结构出现在根元素的前面或后面。注释分别使用字符序列“＜！ －”和“－－＞”作为开始和结束，注释的文本内容在这两个字符序列之间。

6. 命名空间

因为 XML 允许设计者自己选择标记名，所以就可能出现重名的情况。XML 提供了某种机制来解决这个问题，即命名空间。命名空间相当于一个词汇表，它限定了与之关联的所有元素的作用范围。命名空间本身也有名称，它的名称是统一资源标记符(uniform resource identifier，URI)。命名空间的名称和元素的本地名称组合在一起形成了一个全局惟一的名字，即限定名(qualified name，QName)。

命名空间声明通常出现在元素的起始标记中。通常习惯于把一个命名空间的名字映射为另一个很短的字符串，即命名空间前缀(namespace prefix)。命名空间声明的语法为

xmlns: prefix='URI'；如果不使用前缀，则语法为 xmlns='URI'。这两种情况下，URI 都出现在单引号或者双引号中。一个元素只能有一个默认的命名空间声明出现，但是非默认的命名空间声明则是不限的。

所有的 XML 文档都应该是良构的。良构的 XML 文档应该是这样的：所有的构造从语法上都是正确的；只有一个顶层元素，即根元素；所有的起始标记都有与之对应的终止标记，或者使用空元素速记语法；所有的标记都正确嵌套；每一个元素的所有属性都是不同名的。

图 1-1 给出了一个良构的 XML 文档的例子，这个例子将在本书中多次使用。它是一个包含出版物信息的 XML 文档，它有一个 pub 根元素，该元素中包含一个 library 子元素，以及若干个(即零个或多个)book、article 和 editor 子元素；每个 book 元素中有一个 title 子元素、一个 price 子元素和至少一个(即一个或多个)author 子元素，以及一个属性 year；每个 article 元素中有一个 title 子元素和至少一个 author 子元素，以及一个属性 editorID(引用该文的编辑元素)；每个 editor 元素有一个 ID 类型的属性 id，文档中的 editor 元素的 id 属性的值为"105"，一个文档中所有元素的 ID 类型属性的值在该文档中必须惟一；每一个 article 元素通过它的 IDREF 类型的属性 editorID 来引用另一个元素，被引用元素的 id 属性值与该元素的 editorID 属性的值相等，如文档中的 article 元素引用 editor 元素。

```
<? xml version="1.0"? >
<pub>
    <library>Beijing Library</library>
    <book year="2000">
        <title>Database System Concepts</title>
        <price>26.50</price>
        <author id="101">
          <name>Kaily Jone</name>
          <email>kjone@research.bell-labs.com</email>
        </author>
        <author id="102">
          <name>Silen Smith</name>
        </author>
    </book>
    <book year="2001">
        <title>Introduction to XML</title>
        <price>18.80</price>
        <author id="103">
          <name>Kaily Jone</name>
        </author>
    </book>
    <article editorID="105">
        <title>A Query Language for XML</title>
        <author id="104">
          <name>Kaily Jone</name>
        </author>
    </article>
    <editor id="105">
        <name>A. Deutsch</name>
    </editor>
</pub>
```

图 1-1　一个 XML 文档实例

1.1.2 DTD 简介

XML 文档本质上是保存信息的结构化载体。为了得到有效的 XML 文件，还必须要明确文件中的信息必须遵守哪些结构，即需要一种用来描述 XML 文档中信息结构的数据模型。这种数据模型不仅建立了 XML 文档中可以使用的 XML 词汇表(如果 XML 文档没有遵守这个词汇表，那么应用程序就无法从中读取数据)，还定义了 XML 文档中元素的顺序和元素的嵌套关系的内容模型，并建立了文档数据的数据类型。解决方案之一是 DTD(document type definition，文档类型定义)。DTD 列出了可用在文档中的元素、属性、实体和符号表示法，以及这些内容之间可能的相互关系。DTD 指定了文档结构的一系列规则。例如，DTD 可以确切地规定每个 book 元素只有一个 title 子元素，只有一个 price 子元素，有一个或多个 author 子元素。这些规则都是在 DTD 中定义的。

虽然 DTD 不是必不可少的，但它确实可带来方便。DTD 有三个基本用途：

- 对标记编制文档；
- 加强标记参数内部的一致性；
- 使 XML 语法分析器能够确认文档。

XML 的强大功能来自多个文档可以共享常用的 DTD。这些 DTD 可以由不同的人编写，它为形成一致的标记标准并使其文档化提供了一种方式。DTD 还可以确保不同的人员或程序能够互相读懂文件。例如，如果使用一个 DTD 表示基本的化学符号，就可以确保能够阅读和理解对方的文章。DTD 显示了文档的一般结构，这与每个具体文档中的实际数据是相脱离的。DTD 确切定义了在文档内允许出现什么，不允许出现什么。DTD 建立了查看和编辑软件必须支持的标准，而且还建立了一种规则：超出这些 DTD 声明的扩展是无效的。因此，DTD 有助于防止软件销售商为了把用户限制在他们自己专有的软件中，而对开放的协议进行增强和扩展。

DTD 最初出现在 SGML 中，它依靠特定的语法来描述 XML 文档的结构。使用 DTD 作为数据模型的描述方法的一个主要好处是：原有的 SGML 工具可以很容易被修改为支持 XML。图 1-2 显示的是一个 XML DTD 实例，图 1-1 所示的 XML 文档符合该 DTD 的描述。

```
<! ELEMENT pub(library, (book | article | editor) * )>
<! ELEMENT book(title, price, author+)>
<! ATTLIST book year CDATA #IMPLIED>
<! ELEMENT article(title, author+)>
<! ATTLIST article editorID IDREF #IMPLIED>
<! ELEMENT author(name, contact?, email * )>
<! ATTLIST author id ID #REQUIRED>
<! ELEMENT editor(name, contact?, email * )>
<! ATTLIST editor id ID #REQUIRED>
<! ELEMENT library (#PCDATA)>
<! ELEMENT title (#PCDATA)>
<! ELEMENT price (#PCDATA)>
<! ELEMENT name (#PCDATA)>
<! ELEMENT contact (#PCDATA)>
<! ELEMENT email (#PCDATA)>
```

图 1-2　一个 XML DTD 实例

一个 DTD 通过具体说明每一个元素和属性的名称、元素与子元素之间的嵌套关系、子元素的出现次数等来定义 XML 文档的结构模型，其中可以利用操作符 *（0 次或多次）、+（至少 1 次）、?（0 次或 1 次）、|（或选）来定义子元素的出现次数。DTD 假设所有取值都只能是字符串值，但 ANY 类型能够是一个任意 XML 片段。它还有一个特殊的类型为 ID 的属性 id，用来在一个文档中惟一标识一个元素，因此在一个文档中其取值必须惟一。如果一个元素中的属性定义为 IDREF 或 IDREFS 类型，且该属性的取值为另一个或另几个元素的 id 属性的值，则可实现该元素对另一个或另几个元素的引用。DTD 没有根的概念，符合该 DTD 的文档可以用该 DTD 中的任何一个元素作为其根元素。

最简单的使用 DTD 的方法是在 XML 文件的序言部分加入 DTD 描述，其位置紧接在 XML 声明之后。这实际上是定义了一个内部 DTD。一个内部 DTD 的例子如下：

```
<! DOCTYPE pub [
    <! ELEMENT pub(book * )>
    <! ELEMENT book(title, price, author+)>
    <! ATTLIST book year CDATA #IMPLIED>
    <! ELEMENT author(name)>
    <! ATTLIST author id ID #REQUIRED>
    <! ELEMENT title (#PCDATA)>
    <! ELEMENT price (#PCDATA)>
    <! ELEMENT name (#PCDATA)>
]>
<pub>
  <book year="2000">
    <title>Database System Concepts</title>
    <price>26.50</price>
    <author id="101">
      <name>Kaily Jone</name>
    </author>
  </book>
</pub>
```

如果为每个 XML 文件加入一段 DTD 定义，则相当繁琐。因此，通常为一批 XML 文件定义一个相同的 DTD。XML 规范提供了解决这个问题的方法——外部 DTD。外部 DTD 将 DTD 和文档分离，通常把 DTD 存储在一个后缀为.dtd 的文件里。外部 DTD 的好处是，可以方便高效地被多个 XML 文件所共享。只要写一个 DTD 文件，就可以被多个 XML 文件所引用。事实上，需要统一数据交换格式时，就是通过外部 DTD 来完成的。这样做不仅简化了输入工作，而且需要对 DTD 进行改动时，不用一一改变每个引用了它的 XML 文件，只要改一个公用的 DTD 文件就足够了。有一种称为公用 DTD 的外部 DTD，它是由权威机构制订的，可提供给特定行业或公众使用。引用一个外部 DTD，只需要在 XML 声明后面加上一行 DTD 声明。例如，如果将图 1-2 所示的 DTD 存为 pub.dtd 文件，那么外部引用这个 DTD 的 XML 文档为：

```
<? xml version="1.0"? >
<! DOCTYPE pub SYSTEM "pub.dtd">
<pub>
  <book year="2000">
    <title>Database System Concepts</title>
    ...
  </book>
  ...
</pub>
```

DTD 使语法分析器能够明白它所解析的 XML 文件的内容和结构，说清文件结构中的每一个细节，这些细节是通过元素类型定义来定义的。元素类型定义由它们的元素内容模型来描述，其形式为：

```
<! ELEMENT 元素名 (元素内容模型)>
```

在图 1-2 中，DTD 的第一行定义了 XML 文档的根元素 pub。pub 元素有四个子元素 library、book、article 和 editor。其中，library 是必需的，只能出现一次；book、article 和 editor 都是可选的，可以出现 0 次或者任意多次。第二行定义了元素 book，它包含了三个子元素：title、price 和 author；title 和 price 只能出现一次，而 author 可以至少出现一次，并且这三个子元素只能按照这个顺序出现。

在 DTD 中定义属性时，使用下面的格式：

```
<! ATTLIST 元素名 (属性名 属性类型 默认声明) * >
```

元素名是属性所属的元素的名字，属性名是属性的命名，属性类型则用来指定该属性是属于有效属性类型中的哪种类型，默认声明用来说明该属性在 XML 文件中是否可以省略以及默认值是什么。共有三种默认声明，一是 #REQUIRED，表示该属性在 XML 文件中是必须出现的；二是 #IMPLIED，表示该属性在 XML 文件中是可以省略的；三是声明默认属性值。在第三行中，定义了 book 元素的属性 year，其类型是 CDATA，CDATA 指的是纯文本，即由字符、符号 &、小于号(<)和引号(")组成的字符串，它表明 year 属性的值是经过语法分析的字符数据。后面的 #IMPLIED 关键字表示文法解释器不强行要求在 XML 文件中出现该属性，这是对属性的最低要求，现实中经常用到。

有了图 1-2 中的 DTD，阅读符合该 DTD 的 XML 文档的人员(以及对它进行语法分析的语法分析器)就可知道描述出版物信息的文档结构了。关于 DTD 的详细介绍，可以参考文献[1, 2]。

1.1.3 XML 模式简介

DTD 是 XML 从 SGML 继承而来的。SGML 是为描述性文档(例如书籍、报告、技术手册、宣传资料、网页等)而设计的，DTD 用来满足这类文档的需要，已经做得很好。但是 XML 超过了 SGML 的使用范围。XML 可用于股票交易、远程过程调用、图形文件格式以及很多看上去与传统的描述性文档无关的应用，在这些新的应用领域中，DTD 就显示

出了局限性。

第一个问题是 DTD 几乎完全没有数据类型的定义，特别是对元素的内容而言。DTD 中，一切都基于字符串，无法将 year 表示为一个四位数字，也不能将 month 表示为一个 1～12 之间的整数。上述这些内容对于 SGML 处理的描述性文档而言不是特别重要，但是要想进行计算机与计算机之间的信息交换而不是计算机与人之间的通信，涉及数据格式是很常见的情况。

第二个问题是 DTD 的定义不符合 XML 语法。例如下面是一个常见的元素定义：

```
<! ELEMENT title (#PCDATA)>
```

这不是一个合法的 XML 元素，它采用的是一种不同于 XML 文档中元素描述的方法。这给处理上带来了麻烦，如果 XML 足够强大，那么它应该能够描述其自身，不需要另外一种语法来描述有关信息结构的元信息。

第三个问题是 DTD 只能有限地进行扩展，而且扩展得不是很好。

第四个问题是 DTD 的约束定义能力不足，无法对 XML 实例文档做出更细致的语义限制；在 DTD 中，符号“?”、“*”和“+”分别指定“零或一个”、“零或多个”、“一个或多个”，但如果要表达形如 author 只能出现 1～3 次这样的约束，似乎有点为难。同样，要表达 price 必须精确到小数点后两位数也有困难。

第五个问题是 DTD 不够结构化，重用的代价相对较高。

XML 模式试图解决所有这些问题。模式的功能包括：

- 丰富而强大的数据类型；
- 基于命名空间的 URI 的有效性验证；
- 可扩展性和可伸缩性；
- 可重用性。

但是，XML 模式不能解决所有问题。特别是模式不能替代 DTD，可以对同一个文档使用模式和 DTD。DTD 可以进行模式不能进行的操作。当然，DTD 非常适用于最初设计时针对的经典的描述性文档；而且对于这种类型的文档，DTD 比相应的模式要容易编写。解析器和其他软件只要支持 XML 就会继续支持 DTD。

在 XML Schema 之前已经开发了四种模式语言：XDR(XML data reduced，XML 数据简化)、DCD(document content description，文档内容描述)、SOX(simple outline XML，简单 XML 概要)和 DDML(document definition markup language，文档定义标记语言)。这四种模式语言已经有些过时了，但它们是 XML Schema 的起点。W3C 于 1998 年开始制定 XML Schema 的第一个版本，在 2001 年 5 月正式由官方推荐。正式推荐的版本包括三部分[3～5]。

- XML Schema Part 0：Primer。这是对 XML Schema 的非标准介绍，提供了大量示例和说明。
- XML Schema Part 1：Structures。这部分描述了 XML Schema 的大部分组件。
- XML Schema Part 2：Datatypes。这部分包括简单数据类型，解释了内置的数据类型和用于限制它们的方面(facet)。