XML 文档信息的几种转换方法分析与应用实例

李力鸿 邵 敏 郑震坤 何 川 都志辉
(清华大学计算机科学与技术系高性能技术研究所 北京100084)

Typical XML Document Transformation Methods and an Application System

LI Li-Hong SHAO Ming ZHENG Zhen-Kun HE Chuan DU Zhi-Hui
(Department of Compater Science and Technology, Tsinghua University, Beijing 100084)

Abstract XML is a promising technology developed in recent years. Due to its superiority in extensibility and flexibility, XML has become the language over the internet. With more and more XML documents produced, the problem exists to transform them to other documents of various structures. This paper discusses and compares four methods to transform XML documents, then introduces XSLT, a W3C recommendation, by giving examples and outlining a project in brief.

Keywords XML, XSLT, XML Transformations

1. 引言

XML (eXtensible Markup Language, 可扩展标记语言)[1]是最近几年发展十分迅速的一项技术。由于其各种突出优点,以及 W3C(World Wide Web Consortium,万维网协会)正不遗余力给它制定的各项应用标准[2],可以预见,XML 将会在更广泛和更深入的领域取得重要的应用。

XML 文档与 HTML 文档在语法上十分类似。但是 HTML 只提供了固定的、有限的标签集来告诉浏览器如何显示内容,而没有对内容本身做出限制,也没有体现出内容之间的关系。因而,HTML 只适用于编写网页,而很难扩展成其他应用如语义 Web 技术等。相比之下,XML 是一种元语言(meta-language),是 SGML (Standard Generalized Markup Language,标准通用标记语言)一种简化形式。它的主要特点与应用有:

(1)灵活的扩展能力:和其他普通的标记语言如 HTML 不同,XML 没有固定的标签集合,它只提供了一种通用的、灵活的语法机制,用文本格式文件来记录数据以及数据之间的层次关系。使用者可以根据具体应用、自己的习惯来设计出高效的、可扩展的数据结构。而且 XML 既允许使用者随意设计 XML 文档,也通过命名空间的机制保证各个 XML 文档之间不会出现名字冲突。

(2)高度结构化、层次化的数据组织形式:XML 的一个很大的优点是它的数据是以树状层次结构保存数据的。每个 XML 文档都有且只有一个最顶层的元素,该元素下有文本数据或子元素,子元素也可以有自己的子元素和文本数据。这样构造出来的 XML 文档不仅层次结构十分清晰、便于定位元素,而且和计算机科学技术的很多数据结构建立了很好的对应关系。如算法设计中常用的"树";如果让关系数据库里的表对应 XML 文档的一个元素,那么表中各个字段也可以对应该元素的子元素。

(3)数据与表现形式的分离:HTML 文档的一个局限之处在于数据和表现形式没有分离,无论数据还是形式改变,文档中的数据和形式都要重新开发,既不利于数据的重复利用,也不利于数据一致性维护。而 XML 可以只保存数据,或者只表现形式,两者之一改变了,另一者仍然有效。

(4)数据结构的开放性:XML 规范是由 W3C 制定的,完全公开,不同组织开发的 XML 文件格式对于其他人和程序来说都是可读的。例如字处理软件,几乎其他所有字处理软件都无法读取 Microsoft Word 文档,因为它的格式保密。如果 Word 的文件格式采用 XML,那么将是在原码开放、格式共享的路上前进了一大步。

(5)设计与特定领域有关的标记语言:XML 允许各种不同的领域的工作者开发与自己的特定领域有关的标记语言。这就使得该领域中的人们可以方便地交换数据和信息,而不用担心接收端的人是否有特定的软件来解读数据。现在已有的专门应用如,多媒体领域的 SMIL、用于电子商务的 cXML、图形图像领域的 SVG 和无线领域的 WML 等。

由于 XML 的各种良好特性,将会有越来越多的数据以 XML 的格式保存和传输。在很多应用场合中需要把 XML 文档转成其他格式的文档。例如,构建网站时需要把 XML 转换成常见的 HTML 文档;在电子商务中,企业间通过 XML 文档交换数据,但在接收数据后就可能先要把 XML 文件转换成企业内部的数据格式,才能往下处理。

因为 XML 的灵活性和无限扩展性,对同一类数据结构,不同人之间、甚至同一个人不同时期设计的 XML 文件数据层次关系乃至元素命名都会有所不同,所以 XML 文件之间同样存在结构转换的问题。解决了这个问题,XML 才能更好地作为数据传输通用格式而在各种应用场合受到广泛应用。

本文介绍 XML 转换的四种方法,并通过对比讨论了这些方法之间的特点与适用范围,然后通过示例和一个实际项目较为详细地介绍了 W3C 标准 XSLT 1.0,最后对几种方法进行总结,并对未来的工作进行了展望。

2. XML 转换的几种方法

对 XML 进行转换有几种方法。就像不同的程序设计语言各有千秋一样,这些方法各有所长,也都有自己的局限性,所以它们分别适用于一定的场合,应该根据特定的要求在实际应用中找到相应的解决方案。

2.1 自己编制程序

和早期的编程一样,这种方法要求程序员自己编写代码解析文件,进而转换文档。这种方式的工作流程如图1所示,程序员根据转换的具体要求编写并调试好程序 P,工作时 XML 文件作为字节流输入,由程序进行词法和语法分析,对输入的

字节流根据要求进行处理，得到的输出字节流就是转换之后的文件了。

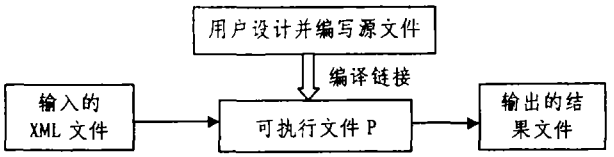

图1　编制程序转换 XML 流程示意图

这种方法的好处是转换器在设计良好的时候速度比较快；然而缺点是编制程序代价极大——开发效率很低，难度大，软件复用度低，而且正确性难以保证。所以一般情况下，除非任务比较简单、或是速度要求很高，否则不应该采用这种方法。和下面将要介绍的标准化、通用化的方法相比，它所增加的难度和工作量主要体现在词法分析和语法分析两方面。为了加快程序开发的速度和提高程序正确率，我们也可以借助一些很好的工具来解决词法、语法分析的问题。这里要介绍的是 LEX 工具[3]。

LEX 是一类广泛使用的工具，常见的有 FLEX 等。LEX 主要用于构造正规表达式所代表的语言的识别器，或者说构造一个能识别相应语言的自动机。我们编写 LEX 语言程序，用 LEX 工具对之编译，产生一个含有词法分析程序的 C 程序源文件 lex.yy.c，利用其中的词法分析函数可以识别出相应的语言。其过程如图2所示。

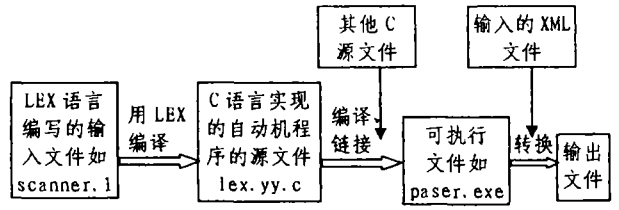

图2　利用 LEX/FLEX 工具转换 XML 流程示意图

2.2　利用 SAX

SAX(Simple API for XML，XML 简易应用程序接口)和下面将要提到的 DOM 是两种工作原理不同的 XML 解析器。DOM 允许程序员把 XML 文档看作是抽象的一棵树，可以让程序方便地访问。而 SAX 在解析 XML 文档时通过向应用程序报告解析过程中的事件流来告知应用所解析文档的内容，如一个元素的开始、结束，遇到可解析字符流等。SAX 工作方式如图 3所示。DOM 工作方式可以对比图4。

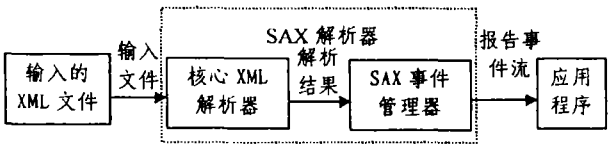

图3　SAX 解析器工作示意图

例如对清单 1，SAX 类型的解析器将会向应用程序报告下面一系列的事件：

```
(开始)
StartDocument
startElement("Books")
startElement("Book")
characters("Computer Networks")
endElement("Book")
StartElement("Book")
characters("Computer Architecture")
endElement("Book")
endElement("Books")
endDocument
(结束)
```

清单1　Books.xml

```
<?xml version="1.0"?>
<Books>
    <Book>Computer Networks</Book>
    <Book>Computer Architecure</Book>
</Books>
```

使用 SAX 比使用诸如 LEX 等工具方便不少，因为它已经提供了词法、语法分析的内容，也有报错的功能。使用者可以不必考虑这些繁琐的细节问题而把精力放在解决核心问题上。但是相对 DOM 来说，SAX 还是比较底层的，程序员无法要求访问文档中任意位置的节点；相反，是 SAX 在解析过程中主动向程序报告事件，于是，程序员往往属于较为被动的一方，从而加大了编写程序的难度。

虽然 SAX 有某些不足，但是它在某些应用场合却是 DOM 难以替代的，主要体现在：

(1)解析很大的 XML 文件时，SAX 比 DOM 需要的内存小得多，而且和文件大小基本无关。

(2)有时只需要访问 XML 文档里信息的一个很小的子集，这时 SAX 能把系统资源用于感兴趣的部分文档，从而显著提高内存利用效率和运行速度。

有了 SAX 和它报告的事件，我们可以在事件处理函数中加入自己的代码，从而完成 XML 转换。常用的 SAX 解析器有很多，如 Microsoft 的 Microsoft XML Core Services (MSXML)，IBM 的 Xerces，Sun 的 JAXP 等。

2.3　利用 DOM

DOM (Document Object Model，文档对象模型) 是和 SAX 并列的另外一种 XML 解析器类型。为了便于对文档进行操作，解析器先将文档全部读入并解析，由于 XML 文档的树状结构，可以把它看作一棵抽象的文档"树"，树上有各种节点，节点有自己的类型和相关的属性值。一旦文档被读入，就在内存中保留整个文档的所有信息，所以程序可以随意访问任意位置的节点的信息或是对之进行修改，程序可以在很高很抽象的层次上动作，而不必把精力花在底层工作上。DOM 工作流程如图4所示。

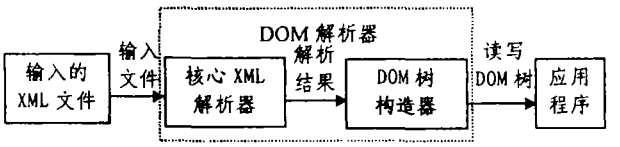

图4　DOM 解析器工作方式

例如清单1的解析结果就如图5所示。

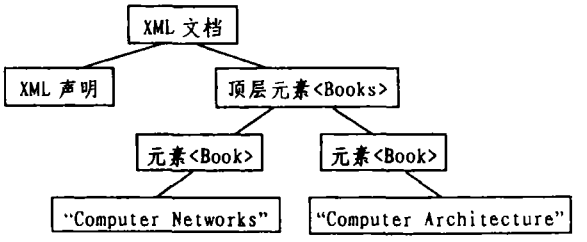

图5　文档 Books.xml 的 DOM 树结构

使用 DOM 最大好处就是方便编写程序。DOM 提供了大量的接口，常用的有 load(加载 XML 文档)、getElementBy-

TagName(找出与路径、名字匹配的元素列表)、createNode(为当前文档创建一个节点)等。

虽然DOM编程方便高效,但是程序运行效率不会太高,消耗的内存也相当大。尤其是解析大文件的时候,整个文档树信息保留在内存中会对内存产生很大压力。大多数情况下解析一个100k的XML文档至少需要1MB的内存[4],而整个程序所需要的内存就更大了。

图6和图7分别给出了SAX和DOM程序在运行时间和消耗内存量两方面的实验数据。这两个程序均用C++编写,分别实现了SAX和DOM接口。程序处理的XML文件根元素是〈Books〉,文件结构如清单2所示,"问题规模"指元素〈Book〉的个数。程序任务是改变〈Price〉元素的文本值。

清单2 Books.xml文件示例

```
<?xml version="1.0"?>
<Books>
    <Book ISBN="7-111-07315-0">
        <Title>Professional XML</Title>
        <Authors>
            <Author>Didier Martin</Author>
            <Author>Mark Birbect</Author>
            <Author>Michael Kay</Author>
        </Authors>
        <Price>95.00</Price>
    </Book>
</Books>
```

从图6可以看出,相对于问题规模,SAX程序运行时间基本上是线性的(注:为清晰起见,附图横坐标未按比例画出);而DOM程序则不然,其运行时间比SAX程序长,而且当问题规模增大时,这种差距更加明显。如上文所讨论的,原因在于两者工作原理的不同:DOM程序需要对庞大的DOM树操作,必然增加定位元素、维护树信息等开销;另一方面,这个程序只需访问和修改〈Price〉的值,而DOM程序在创建文档树的时候把其它无用的信息也预先处理了,这同样增加了不必要的开销。

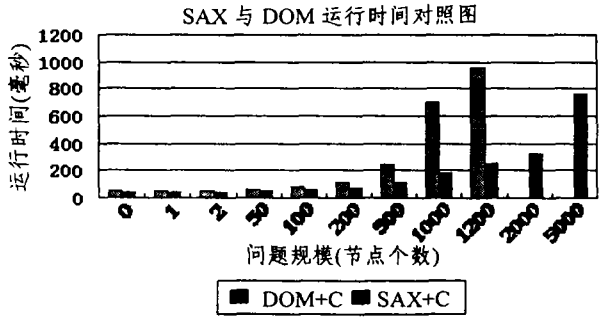

图6 SAX与DOM运行时间对照图

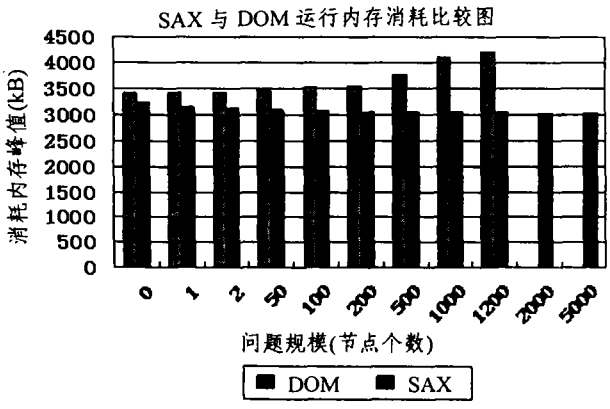

图7 SAX与DOM运行内存消耗比较图

从图7可以看出,SAX程序需要的内存比DOM程序要小,而且需要的内存数量不随问题规模增大而增大,这使得SAX在某些场合十分有用。相反,DOM程序不仅内存消耗量大,而且当问题规模增大到一定程度时由于内存不足等原因无法使用。

2.4 利用XSLT

本文要讨论的最后一个、也是最重要的一个转换方法是XSLT(eXtensible Stylesheet Language Transformation,可扩展样式单语言转换)。它是W3C正在制定的规范XSL(eXtensible Stylesheet Language,可扩展样式单语言)中的一部分,负责XML文档的转换。XSLT转换器(或称XSLT转换引擎)以XML文件为输入,利用预先编写好的转换模板(XSLT文件,也是XML文件)把它转换成目标文件。其中,目标文件可以是XML文件,也可以是其他格式的文件如HTML,甚至C程序源文件等等。XSLT不依赖于任何一种编程语言,它本身也是一种符合XML规范的标记语言,可以看作XML的一种应用,发挥了XML元语言的特点,而反过来对XML文档进行转换操作。

使用XSLT有很多优点:

(1)功能强大。XSLT提供了条件判断、排序、变量定义、表达式求值、函数库支持等手段来实现各种变换的要求。可以看出,XSLT已经部分具有高级编程语言的能力,足以应付多数场合下的变换要求。

(2)它可以很方便地把XML文件转换成HTML文件,从而利用XML+XSLT可以借助HTML在普及性和易用性等方面原有的优势迅速方便地构建Web网站。本文后面将会比较详细地介绍一个具体的构建Web网站项目。

(3)XML是独立于表现形式的保存"真正"数据的文件,而XSLT可以告诉浏览器(和别的应用程序)如何展示这些数据。因此,对于保存同一份数据的XML文件,我们可以设计不同的XSLT来产生不同的表现形式。这种应用是十分重要和有很好前景的,它使同一个XML文件呈现出不同的视角,大大拓宽了XML的应用领域[5]。

(4)XSLT符合XML的语法,可以利用XML各种良好的可扩展特性和工具。如编码、名字空间、XML编辑器等,甚至可以利用XML解析器来对XSLT文档进行处理。

(5)它是W3C制定的标准,通用性有保证。使用者不必考虑工作运行的平台或是编程语言的不兼容性等。

当然,XSLT也有不足的地方。首先是运行效率不会太高,而且和所采用的转换器性能有关。例如文[6]介绍:有人发现,如果用Microsoft XML Parser 3,则运行效率和XSLT中采用的句法也有很大关系:实现同一个功能的两种方法〈xsl:variable select="name"/〉和〈xsl:variable〉〈xsl:value-of select="name"/〉〈/xsl:variable〉相比,前者效率高得多。

然而,XSLT无可置疑的是XML一个极为重要的应用。本文下面将以具体示例来介绍这种转换方法。

3. XSLT介绍

XSLT[7]是W3C制定的XSL标准中的一部分,与之密切相关的还有XPath[8],负责定位XML文档中的元素或属性。

3.1 XPath简介

XPath是W3C关于查询部分XML文档的通用语言标准,XPath1.0已经在1999年11月16日发布(和XSLT1.0的发布是同一天)。在W3C的XML查询标准系列中,还有

XPointer 等。XPath 的设计同时也是为 XPointer 服务的。它给 XPointer 和 XSLT 提供一套定位 XML 文档的通用机制。这套机制在 XML 文件中的作用就像是 SQL 在关系数据库上的作用一样[9]。XPath 不仅可以用于 XSLT，也可以用于其他定位 XML 的场合，如配合 DOM 解析器来编程等。为了更方便、更灵活地定位 XML 文档，XPath 还提供了对字符串、数字和布尔值的基本函数。

基本的 XPath 表达式包含了三部分：轴(axis)、节点测试(node-test)和谓词(predicate)，形式为：axis::node-test[predicate]。从当前节点集合(或称上下文节点)出发，XSLT 转换器根据这三个部分依次筛选符合条件的 XML 文档节点，最后得到一个节点集合，此集合可能为空，也可能有一个或多个节点。例如清单1中，如果上下文节点是文档的顶层元素〈Books〉，则 child::Book[position()=1]得到的节点集合是〈Books〉的第二个〈Book〉子元素(索引从0开始)。在实际运用中，可以使用表达式的简化形式，如上面的例子可以简写成 Book[1]，这样的表示方法就更接近现代编程语言的风格了。

3.2 XSLT 原理

这部分讨论 XSLT 转换的原理和过程。在进行转换之前，XSLT 转换器先把 XSLT 文件转换成一个内部的数据模型——树，不妨把它称为"模板树"。转换的数据源——XML 文件也转换成一棵树，不妨把它称为"源树"。模板树里定义了一个或多个模板，它(们)指定了源树中的节点在结果中如何转换。众多模板组成了转换 XML 之后输出的数据框架。XSLT 引擎把源树中的各个节点和模板树里的模板进行匹配测试。如果匹配上了，就执行这个模板，输出相应数据；否则不输出任何结果。当测试结束之后，转换也就结束了。

查看 XSLT 转换结果可以有多种方法：

(1)用 XSLT 把源文件转换成 HTML 文件，在 IE5.0以上版本浏览；

(2)用 XT、MSXML Parser 等转换引擎转换，结果保存到文件里；

(3)利用其他命令行工具，如 Microsoft 的 MSXSL。

如前所述，XSLT 提供了强大的机制和功能来实现多种变换，也部分具有高级编程语言的特点，所以利用 XSLT 可以方便灵活地转换不同结构的 XML 文件。在数学领域中有两个主要的 XML 应用——OpenMath 和 MathML。文[10]描述了这两种标记语言和 XSL，并且讨论了利用 XSL 对 OpenMath 和 MathML 进行相互转换的问题。

强大的 XSLT 也能把 XML 文件转换成其他非 XML 文件，这就使得 XML 可以方便地应用于原来某种文件格式已经十分流行的领域，最常见的是转换成 HTML。文[11]也介绍了一个方法把用 XML 表示的 Java 文档(JavaML)转换成传统形式的 Java 源程序文本文件。

3.3 相关工作

XML 对于当今学术界和工业界也不再是陌生的名词了。由于种种优良特性，它已经在众多领域展开了应用。与此同时，和 XML 转换相关的大量工作也在进行中。

文[5]介绍了一个利用 XML 技术实现的课件系统。该系统利用 XML 保存课件内容，利用 XSLT 来重新组织课件和指示课件对外展示的风格，使课件在不同场合不同情况下根据要求展现不同的外在风格。这项工作与后面介绍的"文档管理系统"一样，都是通过引入 XML 实现数据和表现形式的分离。

文[10]介绍了两个数学领域的 XML 应用：OpenMath 和 MathML，也讨论如何利用 XSL 转换这两种结构不同的 XML 文档。利用 XSL，两种标识语言可以相互转换；利用 XML 规范中的命名规则，可以保证两者不会出现命名冲突。

文[11]为 Java 程序设计了一套基于 XML 的文件格式——JavaML，给出了相应的 DTD。传统源程序都是以文本文件形式保存与书写，作者也描述了如何利用转换程序把 Java 源文件转换成 JavaML 文件，以及如何利用 XSLT 把 JavaML 转换成传统的文本格式文件。把文本源文件转换成高度结构化的 XML 文档，相当于给源文件加入结构化信息，大大有利于程序分析工具和软件工程分析工具正确有效地理解源文件。

与前面的文献有所不同，文[12,13]从较为理论的角度出发，利用数学工具建立了 XML 的数据模型，然后讨论了关于 XSLT 模式的语法与语义，在语法和句法方面对 XSL 进行了较为深入的讨论和研究。这些工作对于形式化的研究 XML 和 XSL 是十分重要的。

4. 我们的工作

这里介绍我们实现的一个具体项目——利用 XML＋XSLT＋Servlets 开发在线交互式论文管理系统 DMS[14]。

4.1 系统结构

DMS 运行在清华大学计算机系研制的机群 THNPSC-2 上，操作系统采用 Linux，数据库系统采用 Sybase。系统使用方法与用 ASP 等技术实现的 Web 网站完全一样：用户登录后，通过链接选择不同页面，系统提供查询、统计、提交、修改、查看等功能。

和传统 Web 网站不同，DMS 完全采用 XML 作为数据传输格式，而用 XSLT 作为转换的工具，把 XML 文件转成 HTML 文件，方便在浏览器上浏览。各个页面对应的 XSLT 文件是固定的，保存在服务器端；而 XML 文件是用 Servlets 和 JDBC 根据用户输入和数据库查询结果动态生成。系统框架如图8DMS 系统框架所示。

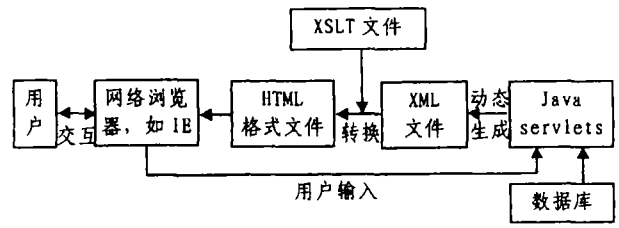

图8 DMS 系统框架

4.2 DMS 的文件格式

DMS 系统的文件除了上载和下载的文件(如 Word 文件或 PDF 文件)外都采用 XML 作为文件格式。可以说，XML 是 DMS 数据流的通用格式。采用 XML 作为文件格式有以下几个好处：

1. 分离数据和表现形式，实现了 Web 页显示的新机制。不同于以往在客户端解释 HTML 页面的机制，XML 可以在服务器端进行解释。这种做法的另一个好处是可以增强安全性，保密性较好。

2. XML 可以实现不同应用间的数据共享，也利于数据格式的向前兼容性。因而，如果扩展程序功能，可以编写新的应用程序而不必担心该程序不能解析 XML。同样，当日后需要

与其他已有的系统通讯，也可以直接把文件稍作处理（转换XML）就能传送数据而不必像以前那样在两个系统互相理解对方数据结构的工作上花很大力气。

3. XML 和数据库之间有着良好的对应关系，以 XML 为格式有利于和数据库模块的交互。

4. XML 和 Servlets 的综合应用[15]。可以用 Servlets 动态、灵活地生成、修改服务器上的 XML 文档，并利用 XSLT 把 XML 文档转为 HTML 格式的 Web 网页，最后将这些网页发送给客户机。这两种技术都是跨平台技术，可为系统带来更大的灵活性和可移植性。

4.3 数据库模块

和常见的信息系统一样，DMS 主要以数据库保存数据，如用户信息、论文信息等。为了便于访问、修改数据库以及平台的移植，在 DMS 系统中以 JDBC 为基础，采用分层的方法对数据库和数据表进一步封装，构造了一个比较完整的数据库服务中间件。针对不同类型的数据库，数据库中间件可以使用不同的 JDBC 驱动程序与之相配，所以这个数据库中间件可以同时支持多种不同种类的数据库，如本系统采用的 Sybase，或其他的 DBMS 如 Oracle、SQL Server 等。

通过 JDBC 构造的中间件解决了查询的简便化和平台移植问题，但逻辑上或物理上分布在各处的数据库（或更一般的说是数据来源）在传输和交换数据时一定要采用统一的数据格式。而 XML 正好擅长此项工作。

对于数据库来说，让 XML 文档的元素和数据库的字段对应起来是很自然和容易的事。而让 XML 文档作为关系数据库的一个文本字段进行存储也是十分方便的，因为 XML 本身具有良好的结构便于查询。同时，还可以利用 DTD/Schema 来限定 XML 文档的有效性，规定 XML 文档中元素的命名和相互层次关系。

4.4 Servlets 动态生成 XML 文件

所有用户请求的处理、数据库查询工作和 XML 文件的动态生成等都需要软件来实现。DMS 中整合各部分的是 Java Servlets。

Java Servlets 为构筑基于 Web 的应用程序提供了一种平台无关的方法和协议：以任何语言实现的客户端取得用户的请求，把它发给服务端，服务端运行的 Servlet 引擎把该请求转给 Servlets 处理，最后 Servlets 向客户端回发一个响应，从而完成一次 Client-Server 通信。用 Java 编写的软件最大的好处在于其平台无关性，因此 DMS 可以运行于任何平台。

我们主要利用 Servlets 的两个函数 doGet() 和 doPost() 分别处理以 GET 方式和以 POST 方式提交的请求：先获取客户端传来的各个参数，对这些参数进行相应处理，然后操作数据库，并用动态生成的数据生成 XML 文件，最后利用 XSLT 把 XML 文件转换成一般浏览器能解析的 HTML 文件，显示在客户端的浏览器中。

结论与展望 XML 是一种很有发展潜力的技术，而 XML 转换则是充分发挥 XML 无限扩展性的一个关键技术。本文总结了四种实用的转换技术，它们互有长短，使用时应该根据具体情况进行选择。

在这些技术中，XSLT 是一种应用前景广阔的技术。它已经被 W3C 标准化，XSL 的标准化也接近达到。XSLT 可以利用强大的类似高级编程语言的特性、功能让 XML 文件呈现多种多样的视角，从而大大拓宽 XML 的应用领域和提高它的实用价值。

作为"继 Java 之后的又一激动人心的技术"，XML 带给我们全新的数据格式，它使不同人、组织、应用程序之间方便的数据交换成为可能，同时也具备了无限扩展性等极好的优点。我们应该重视这种正处于蓬勃发展中的技术，不仅要积极地留意国内外的技术动态，还要用于把这种新技术用到具体工作中去，让 XML 切切实实地发挥应有的作用。

参考文献

1 Bray T, et al. Extensible Markup Language (XML) 1.0 Specification (Second Edition), 2000. Available at: http://www.w3.org/TR/2000/REC-xml-20001006

2 杨建武，等. XML 相关标准综述. 计算机科学，2002，29(2)

3 吕映芝，等. 编译原理. 清华大学出版社，1998

4 Martin D, et al. Professional XML, Wrox Press, 2000

5 Qu Changtao, et al. Collaborative courseware authoring and publishing based on WebDAV, XML, and XSLT. EUROCON'2001. Intl. Conf. on Trends in Communications. Technical Program, Proceedings (Cat. No. 01EX439). IEEE. Part vol. 2, 2001, pp. 266-9 vol. 2. Piscataway, NJ, USA

6 Jones R. Sometimes Syntax Is Significant. XML Magazine, 2002

7 Clark J. XSL Transformations (XSLT) 1.0 Specification, 1999. Available at: http://www.w3.org/TR/1999/REC-xslt-19991116

8 Clark J, et al. XML Path Language (XPath) 1.0 Specification, 1999. Available at: http://www.w3.org/TR/1999/REC-xpath-19991116

9 Gudgin M. X marks the path[XPath]. Developer Network Journal, 2001, (25): 26～31

10 Carlisle D. OpenMath, MathML and XSL. Sigsam Bulletin, ACM, 2000, 34(2): 6～11

11 Badros G J. JavaML: a markup language for Java source code. 2000, Ninth International World Wide Web Conf. Amsterdam, Netherlands

12 Wadler P. A formal semantics of patterns in XSLT and XPath. Markup Languages: Theory & Practice, MIT Press, USA, 2000, 2(2): 183～202

13 Bex G J, et al. A formal model for an expressive fragment of XSLT, Computational Logic - CL 2000

14 邵敏，等. XML 及其在机群系统 THNPSC-2中的应用. 高性能计算机及应用学术研讨会，上海，2001

15 Dumbill E. Getting started with XSLT style sheets and Java servlets. Web Techniques, U. S. A, 1999, 4(12)