SXBP:基于 Pri-order 编码的 XML 文档存储方法

任家东　岳丽文

(燕山大学信息科学与工程学院　秦皇岛 066004)

摘　要　随着 XML 技术的发展,如何利用现有的数据库技术存储和查询 XML 文档已成为 XML 数据管理领域研究的热点问题。本文介绍了一种新的文档编码方法,以及基于这种编码方式提出了一种新的 XML 文档存储方法。方法按照文档中结点类型将 XML 文档树型结构分解为结点,分别存储到对应的关系表中,这种方法能够将任意结构的文档存储到一个固定的关系模式中。同时为了便于实现数据的查询,将文档中出现的简单路径模式也存储为一个表。这种新的文档存储方法能够有效地支持文档的查询操作,并能根据结点的编码信息实现原 XML 文档的正确恢复。最后,对本文提出的存储方法和恢复算法进行了实验验证。

关键词　编码模型,存储方法,文档恢复

SXBP : The Storage Method of XML Documents Based on the Pri-order Labeling Scheme

REN Jia-Dong　YUE Li-Wen

(College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004)

Abstract　With the development of the technique of XML, how to make use of database to store and query XML documents has become a hot topic. In our paper, a labeling scheme and a storage method of XML documents based on this labeling scheme are proposed. This method decomposes the document tree structure into nodes and stores them into the relational table according to the node types; it enables us to store any kinds of documents using a fixed relational schema. The simple paths of document were also stored with a table. This method supports the query and retrieval of original XML document efficiently based on the label of node. We report our experimental results on a real dataset to show the performance of our method at last.

Keywords　Labeling scheme, Storage method, Document retrieval

1　前言

随着网络技术的发展,XML 迅速成为 Internet 上数据表示和文档描述新的标准,在过去的几年中,基于 XML 技术的应用研究技术也在不断增加。如何利用现有的成熟的数据库技术操作 XML 数据,实现在数据库中有效的存储和查询 XML 文档已成为一个重要的研究领域,并逐渐成为学术界研究的热点问题之一[1~3]。根据存储系统底层使用的技术,一般将文档的存储方法分为以下几种:文件系统方式[4]、专有系统方式[5]和数据库系统方式[6,7]。将 XML 数据存储到关系数据库系统中是目前最普遍应用的一种映射方法,采用关系数据库存储主要有以下两个优点:1)关系数据库能够处理大量结构复杂的数据;2)关系数据库中有成熟的查询优化技术和事务管理机制。采用基于模型的映射的转换方法,也保证了关系存储模式不受文档逻辑结构的影响。模式间的转换通常有两种映射方法:基于结构映射和基于模型映射,传统的文档存储方法 FK[2]和 XRe[3]都是基于模型映射的方法。

设计一个有效的文档存储模式,也需要考虑转换后文档的恢复和更新频率,文档的恢复和更新则受到文档编码方式的影响。

本文在分析现有的编码技术的基础上提出了一种新的文档编码方式,并基于这种编码方式提出一种新的文档存储方法,称为"SXBP",这种存储方法是对传统的 XRel 方法的改进。这种新的文档存储方法主要有以下几个优点:1)数据库模式独立于文档模式,能够处理任意结构的文档;2)按结点类型将文档结构分解,存储到对应的关系表中;3)根据结点的编码方式,方法保证文档有较高的恢复和更新效率。

2　Pri-order 编码

文档编码是数据库模式间转换的关键技术之一,通过为结点分配编码保存结点之间的顺序。文档的恢复和更新的可行性也受到文档编码模式的影响,一种好的编码方式能够在这两种操作间达到平衡。

本文提出的文档编码方案称为 Pri-order 编码,利用一个三元组(pri,ord,level)来表示结点的编码。在这种编码方式中,pri 是按素数编码方式[8]的性质为结点分配编码,首先深度优先遍历文档树,为文档中每个结点分配一个从未使用的素数作为结点自身的编码,pri 是结点自身分配的编码与其父结点编码 pri 部分的积。可以看到,利用这种编码方式,后代结点的 pri 编码能整除祖先后结点的 pri 编码,通过 pri 部分的整除关系能够快速地确定文档树中任意结点间的祖先后代关系。同时文档的更新操作不影响结点的 pri 部分的编码值,当有新的结点插入时不会影响其它已存在结点的编码,只需为新结点分配未被分配的素数作为结点自身的编码,然后

任家东　教授,博士,主要从事时态数据模型、时态数据挖掘以及 XML 数据模型的研究;**岳丽文**　硕士研究生,主要从事 XML 数据存储和查询的研究工作。

计算各部分编码值。pri 部分的编码大小主要依靠文档树的深度而不受文档中结点的扇出大小影响。编码的 ord 部分是按照结点在兄弟结点间的顺序分配的一个整数，结点是第一个孩子编码的 ord 值为 0，ord 值表示结点间兄弟关系的前后顺序，文档中结点编码 ord 部分的最大值表示此文档的最大扇出。当文档中结点编码 ord 部分的最大值表示此文档的最大扇出。当文档有更新操作时，只需要为具有同一父结点的所有兄弟结点重新分配 ord 值，重新编码的结点范围很小。level 表示结点在文档中的嵌套层次，即结点在文档树中的深度，文档结点树中根结点所在的层次为 1。结点所在的 level 越大，结点编码的 pri 值就越大，结点的 level 编码值不会受到文档的更新操作的影响。

Pri-order 编码方法利用三元组描述结点的编码，根据编码的三部分可以容易地确定结点间的关系，结点间的关系判定有以下的性质。

性质 1 文档树 T 中任意两个结点 N1 和 N2，当且仅当 N1. pri mod N2. pri=0 时 N2 是 N1 的祖先结点。如果 N1. level−N2. level=m，则结点 N2 为 N1 的 m 代祖先结点。当且仅当 N1. level−N2. level=1，则结点 N2 为 N1 的父亲结点。

3 文档的存储方法

3.1 XML 文档

XML 文档通常可分两种，一种是以数据为中心的结构化 XML 文档；另一种是以文档为中心的非结构化 XML 文档；无论是哪一种结构的文档，无论其是否符合特定的模式，描述文档的数据结构都有严格的层次性，包括嵌套的元素结构，以一个根元素开始，元素的内容包含在属性或子元素中。元素严格地以开始标签和结束标签标识，空元素以空元素标签标识。下面是一个描述有关书信息的 XML 文档实例

```
example. xml。
  〈books〉
    〈book style="textbook"〉
    〈title〉Application of XML〈/title〉
    〈editor〉
        〈family〉zhong〈/family〉
        〈given〉sheng〈/given〉
    〈/editor〉
    〈author〉
        〈family〉li〈/family〉
        〈given〉guo〈/given〉
        〈family〉qian〈/family〉
        〈given〉ying〈/given〉
    〈/author〉
    〈summary〉
This book is the Appkication of〈keyword〉XML〈/keyword〉
    〈/summary〉
  〈/book〉
〈/books〉
```

这个简单的文档以 books 元素开始经，包含子元素 title，editor，author 和 summary，其属性为 style，author 是个嵌套元素，包含子元素 family 和 given。任何一个元素都有严格的层次性。

3.2 XML 文档的树结构描述

文档存储管理方法[2,9]大多采用树型结构来描述文档的逻辑结构，树中主要包含以下四种类型的结点：根结点，元素结点，属性结点和文本结点，还有指令操作等三类结点，但大多数模型结构中不出现这三类结点，本文在这里也不考虑这三类结点的特性，主要讨论上面提到的四种类型的结点。

根结点是标记文档名称的结点，有一个孩子头元素结点。标记文档结构的开始。元素结点有一个元素类型名作为标记，元素结点可以有零个或任意多个孩子结点。每个孩子结点的类型可以是元素结点，属性结点和文本结点中的任意一种，属性结点则有一个属性名和属性值作为标记，属性结点则没有孩子结点，因为描述属性的信息本身没有次序，当结点有多重属性时，不用区分属性间的顺序。本文结点是 XML 文档中的字符数据，本身没有孩子结点。一般出现在文档树型结构的叶子结点处。

图 1 所示表示 3.1 节描述的 XML 文档 example. xml 对应的文档树型结构。树中结点编码采用 Pri-order 编码方式。

3.3 SXBP：XML 文档的存储方法

XML 文档存储的关键是将 XML 文档的树型结构与结点相关的信息存储到关系表的元组中，将树型结构分解为存储模式对应的二维关系。本章采用的存储方法称为 SXBP 是一种基不同结构的 XML 文档，无论文档是否符合特定的模式，都可以映射到 SXBP 存储模式中。具体的关系模式定义如下：

```
Document(DocID integer, Docname string);
Element(DocID integer, name string, pri integer, ord integer, parord
  integer, level integer);
Attribute(DocID integer, name string, value string, pri integer, inte-
  ger, patord integer, level integer);
Text(DocID integer, value string, pri integer, integer, integer, parord
  integer, level integer);
Path(PathID integer, DocID integer, Path expression string)。
```

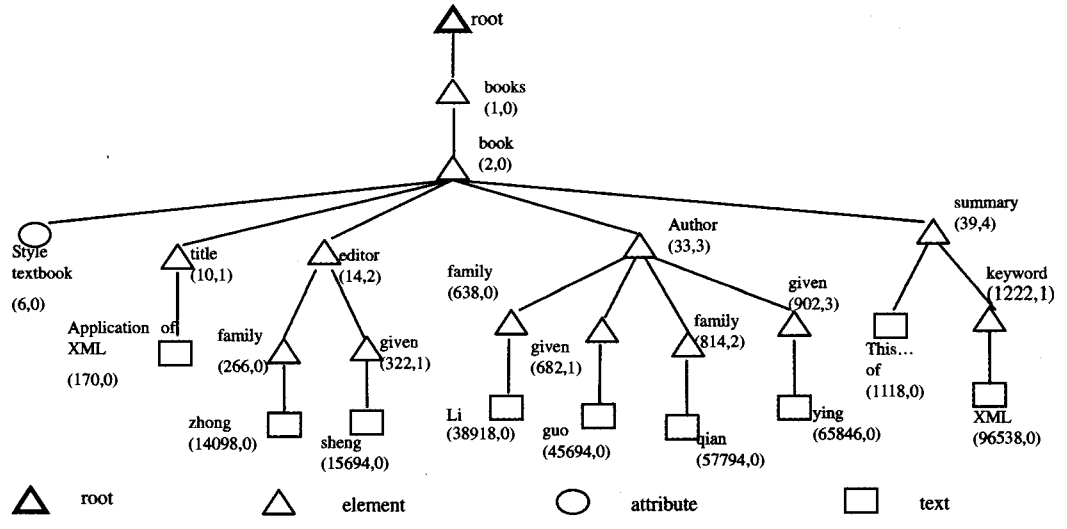

图 1 XML 文档树型结构及其编码

(1)Document 文档根节点映射为 Document 关系，DocID 唯一标识一个 XML 文档，Docname 文档名称标识文档来源。

(2)Element 元素节点转化为 Element 关系，DocID 表示元素结点所在的文档 ID，name 为元素名称，pri、ord 以及 level 分别为结点在文档树中的编码的三部分值，parord 为父结点在其兄结点间的顺序号。

(3)Attribute 属性结点映射为关系表 Attribute，value 为属性的值，其余的参数同(2)中描述。

(4)Text 文本结点映射 Text 关系，value 为字符串值，DocID、parord 及结点的编码值唯一确定了该文本节点所属的元素。

(5)Path 路径关系表示 XML 文档树中元素和属性结点间的绝对和相对路径。PathID 是唯一的路径标识。Path expression 存储文档中具体的简单根叶路径表达式。

SXBP 定义的存储模式可用固定的关系模式实现对任意文档的存储处理，无论文档是否属于特定模式，这种方法适合处理信息量大、结构复杂的文档，顺应了 XML 文档结构和内容不断变化的需要。对于那些属于同一模式的多个 XML 文档，它们的文档模式中含有相同的简单路径，这样可以在关系表 Path 中存储对应的 PathID 和简单路径信息，能够减少多个 XML 文档的重复存储的关系元组数，节省存储空间。

4 文档的恢复

SXBP 模式按照结点的类型对文档树进行分解，分别存储到对应的关系模式中，存储关系表中包含了所有与结点相关的信息。我们可以根据存储关系表中的 DocID、结点的编码信息有效地对原 XML 文档进行恢复。由于存储模式保存了文档中结点的编码信息，存储关系表中元组间有以下的关系性质定理，根据这些性质定理本文也给出了基于 SXBP 方法存储后文档的恢复算法。

4.1 关系表中结点间的关系定理

SXBP 定义的存储模式保存了文档的编码信息，同 Pri-order 编码一样，根据对应的存储关系表中对应的编码属性可以很容易地判定表中任意元组间的关系。

定理 1 根据存储映射方法 SXBP 定义的存储模式可知，存储关系表中的任意两个元组 n1 和 n2，如果 n1. DocID=n2. DocID∧n1. pri mod n2. pri=0∧n1. level=n2. level+1，则 n1 在 XML 结构树中对应的结点是 n2 对应结点的孩子结点。

证明：因为 n1. DocID=n2. DocID，所以 n1 和 n2 在 XML 结构树中对应的结点(设为 N1 和 N2)属于同一个 XML 文档；因为 n1. level=n2. level+1，则 N1 和 N2 有两种关系，一种是两个结点属于同一棵子树，二是两个结点不属于同一棵子树中。假设 N1 在 N2 的兄弟树中，即 N1 和 N2 不在同一子树分枝中。因此 n1. pri mod n2. pri>0，与题设矛盾，所以假设不成立；因此 N1 和 N2 在同一子树分枝中，由题设知 n1. pri mod n2. pri=0 并且 n1. level=n2. level+1，由结点编码的性质 1 可知，n1 在 XML 结构树中对应的结点是 n2 对应结点的孩子结点。

定理 2 根据存储映射方法 SXBP 定义的关系模式，关系表中的任意的两个元组 n1 和 n2，如果 n1. DocID=n2. DocID∧n1. parord=n2. parord∧n1. level=n2. level，则 n2 在 XML 结构树中对应的结点是 n1 对应结点兄弟结点；当且仅当 n1. ord=n2. ord+1，则 n2 在 XML 结构树中对应的结点是 n1 对应结点的第一个左兄弟结点。

定理 2 中，如果 n1. ord=n2. ord+k，则 n2 在 XML 结构树中对应的结点是 n1 对应结点的第 k 个左兄弟结点；相对和 n1 在 XML 结构树中对应的结点是 n2 对应结点的第 k 个右兄弟结点。

定理 3 根据存储映射方法 SXBP 产生的关系表中的任意的两个元组 n1 和 n2，如果 n1. DocID=n2. DocDID∧n1. pri mod n2. pri=0∧n2. level−n1. level=m(m≥1)，则 n1 在 XML 结构树中对应的结点是 n2 对应结点的 m 代祖先结点。

定理 2 和 3 的证明与定理 1 的证明相同，由于篇幅的原因，在这里不予以证明。

4.2 文档的恢复算法

根据上节中给出的定理 1、定理 2 和定理 3，将上述对应的存储关系表 Element，Attibute 以及 Text 表按 DocIDk 字段连接后组成的一个新表，新表中记录按字段 level、parord 和 ord 的值顺序升序排列。根据算法 Retrieval-document 对新关系表中的元组进行操作，重组成原 XML 文档。算法的输入为连接后新表的各字段值，输出为所有元组组成的 XML 文档。

算法 Retrieval-document

输入：关系表 example 中的 n 个元组为 $a_1,\cdots,a_n$，元组有相同 DocID 并按 ord、level 和 parord 的值升序排列。

输出：由表 example 中所有元组构成的原 XML 文档。

```
begin
initstack s=null;
outputfil("<a1. name>"): //恢复头元素
push(s,"</a1. name>");
for(j=2;j<=n;j++)//对其它元组进行恢复
      {if (aj. level−aj−1. level≠ null)
            {outputfile("<aj. mame>");
            push(s,"</aj. name>");}//元素结点的恢复
      else if(aj. value ≠ null and aj. name≠null)
          {outputfile("<aj−1. mame aj. name=aj. value)");
          push (s,"</aj−1. name)");}//属性结点的恢复
      else if (aj. value≠null and aj. name=null)
          {outputfile("aj−1. value");//文本结点的恢复
          for (k=1;k<=(aj−1. level−aj. level+1);k++)
             outputfile(pop(s));//输出与文本结点相关的元素标签
               end for}
          end if
    end if
   outputfile("<aj. name>");
   push(s,"</aj. name>");}
  end for
while (stackempty(s)=false)
     outputfile(pop(s));//恢复所有堆栈中的元素结束标签
end while
end
```

算法 Retrieval-document 通过关系表 DcoID 属性连接文档各类型结点的存储表，达到对原文档的正确恢复。算法首先定义一个堆栈存储元素结点的结束标签(line1～2)，当元素结点的开始标签被输出时，为了保存原文档对应的结构，将元素结点的结束标签存储堆栈中，直到此标签被唤醒。算法对 Element，Attribute 和 Text 三个不同类型的结点对应的元组分别进行处理(line3～8)，最后输出堆栈中所有元素结束标签。为了测试算法的正确性，通过对 3.1 节中的文档按照前一节提到的方法进行存储、连接，算法可以正确地输出原 XML 文档 example. xml，所以算法是正确的。通过分析可知算法的时间复杂为 $O(n)$。n 为文档中结点的个数。

5 实验分析

为了准确分析基于 Pri-order 编码的文档的存储方法

(下转第 136 页)

problem. European Journal of Operational Research, 1996, 94: 392～404
8 Solar M, Parada V, Urrutia R. A parallel genetic algorithm to solve the set-covering problem. Computers & Operations Research, 2002, 29: 1221～1235
9 陈亮，任世军. 一种遗传算法在集合覆盖问题中的应用研究. 哈尔滨商业大学学报（自然科学版），2006, 22(2): 67～70
10 Jacobs L, Brusco M. Note: A local-search heuristic for large set-covering problems. Naval Research Logistics, 1995, 42: 1129～1140
11 Lessing L, Dumitrescu I, Stützle T. A Comparison between ACO algorithms for the set covering problem. Lecture Notes in Computer Science, 2004, 3172: 1～12
12 Ohlsson M, Peterson C, Söderberg B. An efficient mean field approach to the set covering problem. European Journal of Operational Research, 2001, 133: 583～595
13 Beasley J E. An algorithm for set covering problems. European Journal of Operational Research, 1987, 31: 85～93

（上接第 118 页）

SXBP 的性能，采用 Shakespeare 剧本[10]作为实验数据集，数据集为 7.65MB，共有 37 个文档，总的元素结点、属性结点和文本结点数分别为 179618、0、147525，CPU 为 P4 2.4G，操作系统为 Windows XP，内存为 512M，采用 JAVA 语言编程，数据库为 SQL Sever 2000 关系数据库。实验分别验证基于 Pri-order 编码的文档恢复效率和路径查询效率。

5.1 文档恢复率

验证 SXBP 的文档恢复率，对数据集中的 37 个文档分别取出不同的 N 个存储到关系表中，利用文档重组算法对存储的关系表进行操作，取 3 次重组时间的平均结果。实验结果如表 1 所示，由结果可以看到本文的文档重组算法有很高的可扩展性，计算重组时间与重组元组数的比值可以看到算法的时间复杂度与理论上分析的具有线性关系基本吻合。

表 1 文档重组时间（单位：s）

重组时间(s) / 元组数(n)	重组时间(s)	线性比(s/n)
6347	248	0.0391
56390	1872.2	0.0332
106379	3486.8	0.0328
179618	5863.7	0.0326
327143	10689.7	0.0327

5.2 路径查询效率

对于路径查询效率的测试，同样采用 Shakespeare 剧本的 37 个文档作为实验数据集。将 XML 文档解析、编码后存入到关系数据库中。针对具体的查询实例与传统的方法 FK[2]和 Xre[3]在查询所需单上比较，结果如表 7 所示；从表 2 可以看出在路径查询上，本文的存储方法对于简单路径的查询与 FK 和 Xrel 相比所需的时间要稍长些，对带有祖先后代关系（//）的查询时间影响不大，在查询实例中间带有“//”时所需的时间会更长些。而对带有谓词约束的查询与不带谓词约束的和简单路径相比所需时间略有增加，但不是很大。

表 2 实例查询时间和查询结果

查询实例	FK(秒)	Xrel(秒)	本文方法(秒)	查询结果数
/play	0.08	0.17	0.26	37
//scene/title	0.13	0.24	0.28	750
/play/act/title	13.0	0.36	0.34	185
//act//title	16.51	0.31	0.32	951
/play/act/scene/speech [speaker='curio']	19.31	2.75	2.73	4

结束语 本文提出一种通用的在关系数据库中存储和恢复 XML 文档的方法 SXBP，它扩展应用了 XRel 基于一种新的编码方式 Pri-order。Pre-order 用一对整数为文档结构树中结点进行编码，编码的第一部分是通过素数编码获得，另一部分是一个整数，用来表示结点在其兄弟结点间的顺序，这种编码方式在对原文档进行恢复时利用保持结点间的顺序，能达到文档恢复和更新间的平衡。SXBP 方法根据结点类型将树型结构分解，采用基于模型映射的方法将结点的信息分别存储到相应的关系模式中。这种方式能够处理任意 XML 文档，无论文档有无特定的模式，并能够按照结点编码和结点所在层数信息有效地对原 XML 文档进行恢复。文中通过对真实数据集的实验证明，这种存储方法有很高的文档恢复率，在路径查询方面与传统的方法相比具有较高查询效率，节省查询时间。

参 考 文 献

1 Shanmugasundaram J, Tufte D, He Gang. Relational Database for Querying XML Documents: Limitations and Opportunities. In: Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999. 302～314
2 Florescu D, Komman D. Storing and querying XML data using an RDBMS[J]. IEEE Data Engineering Bulletioa, 1999, 22(3): 27～34
3 Youshikawa M, Amagasa T. Xrel: A path-based approach to storage and retrieval of XML documents using relational database. ACM Trans. Internet Technology, 2001. 110～141
4 Wen Lun, Zhang Rui, Lu XianLiang. The Design of Efficient XML Document Model. In: Proceedings ofteh First Intemational Conference on Machine Leaming and Cybemetis, Beijing, 2002. 1102～1106
5 Jagadish H V, Al-Khalifa S, Chapman A, et al. TIMBER: Anative XML database. In: Proceedings of the 28th VLDB Conference, 2002. 274～291
6 Halverson A, Josifovshi V, Lohman G, et al. ROX: Relational Over XML. In: Proceeings of the 30th VLDB Conference, Toronto, Canada, 2004. 264～275
7 Pardede E, Rahayu J W, Taniar D. Preserving Composition in XML Object Relational Storage. In: 19th Intemational Conference on Advsanced Information Networking and Applications, 2005. 695～700
8 Wu Xiaodong, Lee M L, Hsu W. A Prime Number Labeling Schemes for Dynamic Ordered XML Trees. In: Proceeding of the 20th International Conference on Data Engineering ICDE, 2004. 66～78
9 Fujimoto K, Shimizu T, Kha D. A mapping Scheme of XML Documents into Relational Databases using Schema-based Path Identifiers. In: Proceedings of the 2005 International Workshop on Challenges in Web Information Retrieval and Integration, 2005. 82～90
10 Bosak J. The Plays of Shakespeare in XML [EB/OL]. http//metalab. unc. edu/xml/examples/Shakespeare/, 1999