基于 Schema 路径分析的 XML 数据库存储策略*)

朱 震 左 春
(中国科学院软件研究所 北京100080)

XML Database Storing Strategy Based on Schema Path Analysis

ZHU Zhen ZUO Chun
(Institute of Software, Chinese Academy of Science, Beijing 100080)
(zhuzhen_ios@yahoo.com.cn)

Abstract With XML standards playing more and more important role at information exchanges in data management, the XML database storing strategy which supports huge amount XML data control from bottom layer of database has become the key part of fields such as content management. This article proposes an efficiently XML storing strategy from the perspective of constraint schema of XML instance based on XML Schema path analysis.

Keywords Database storing, Path analysis, Node type, XML Schema, XPath

1 引言

随着 XML 标准在信息交换等数据管理中的作用日益增强,从数据库底层支持海量 XML 的数据存储策略,已经成为诸如内容管理等领域的关键组成部分。本文从 XML 实例的约束模式出发,通过基于 XML Schema 的路径分析,建立一种用户透明的 XML 高效存储策略。

随着 Internet 上各种应用的开发,已经产生了越来越多的 XML 数据,很多情况下,这些都是以文件形式存储的。大量零散文件的存在,将在很大程度上影响资源的管理与核心应用的执行效率。现在需要一些方便有效的机制,能够将 XML 数据文件存储在数据库中(比如关系数据库),并且像处理一般的关系表元组那样简单。

到目前为止,已经提出了不少关于解决此类问题的方法,有些是关于 XML 数据索引的,如 Jong P. Yoon 关于 XML 文档三维位图索引[5]的讨论,还有些像 Christian Süß 等的 XML 数据分段存储[6]的讨论。

本文关于 XML 存储策略的讨论使用了非等长路经的方法,并根据结点类型划分路径集合,从而给出相应的存储结构。讨论划定了如下的范围:1. XML 实例是可以被约束的,即每个实例文档都可以被相应的模式定义。2. 涉及的是作为交换数据的具有指定模式的大量 XML 文件向数据库存储转化的情况,而不是一两个用作诸如管理配置的少量 XML 文件在数据库中的表示(尽管没有任何区别)。3. 涉及有关存储方面的讨论。

2 几个基本概念

XML 实例指具体用于数据存储的 XML 语法格式的文件或流,XML Schema 是 XML 实例的约束定义文件,其语法格式是 XML 的,相对于 XML 实例而言,它仅仅是数据的定义,不包含实际数据,具体的语法格式请参见 W3C 的相关标准。XPath 也是 W3C 定义的有关 XML 文档导航的语法格式,这里在对 XML Schema 进行路径分析时将使用 XPath 的语法格式。

一方面,XML 的数据库存储本身是由数据库表组织的,另一方面,从逻辑上看,Schema 定义相当于关系表的定义,一个 XML 实例文档相当于一条元组,请注意,这只是逻辑概念上的等同性,与具体的存储策略无关。

下面部分将采取逐步深入的方法讨论在 XML 数据库存储过程中将遇到的问题,并且将给出详细的存储算法描述,并伴随以一个较为复杂的例子说明存储策略。

3 存储策略分析

3.1 简单 XML 到关系表的映射

观察传统的关系数据库,在关系表的每一条记录即一个无序 n 元组($t_1, t_2, \cdots, t_n$)中,每一个分量 t_i 都隶属于一个预定义的值域 D。那么这个元组至多与 n 个值域有关。

这些值域之间的定义都是相互独立的,如属性列"姓名","年龄"等。这也就是传统数据库表达的方式。显然这里没有表达出这些值域之间可能存在的关系(独立性可以认为是众多关系中的一种最简单的关系)。

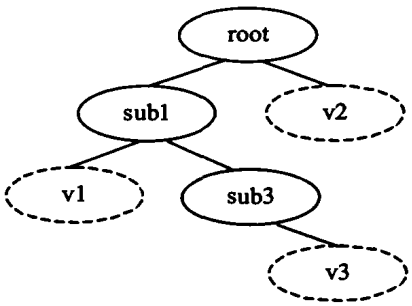

图1 一个简单 XML 文件的原始树形表示

如果要表达这些列定义之间的更为复杂的关系的话,或者说体现值域间的语义关联,那么在定义数据库表的时候或者说定义一个数据集合单元的时候,就需要更多的结构来说明。(见后面的3.4.1)

这里的一个基本观点是让列定义本身体现更多的信息。

*)该项目得到国家自然科学基金重点项目(19831020)的资助。朱 震 硕士研究生,主要研究方向为数据库理论及设计等。左 春 硕士生导师,研究员,主要研究方向为系统分析与集成,大型数据库等。

如一个 xml 文件的树形图(图1)。

可以看出:值 v1,v2,v3处在不同的层次上,他们的路径深度不同。不过我们可以换个视角,如图2。

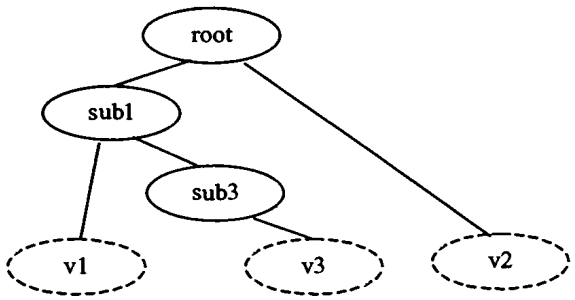

图2 变换视角后的 XML 树形表示

两幅图表达的 XML 文件没有任何区别,但图2把所有的叶节点都"拉"到了同一个层次,即 v1,v2,v3可以看作是关系表的一条记录的三个值。那么这张关系表的列该如何定义呢?

有一个非常直观的想法,v1属于路径/root/sub1,v2属于路径/root/text(),v3属于路径/root/sub1/sub3,那么就变成了使用路径来定义列的语义。上面提到的路径都是可以作为列定义的路径(并不是所有的路径都可以作为列路径,普通的关系表相当于所有值都处于原始树形结构的同一层)。

这里的一个关键是把类似于图1的 XML 结构映射为一张关系表,而不是按层次划分的几张关系表,可以看出:这张关系表每一列的定义所表示的 XML 结构的路径长度可能不同。

这种划分将作为后面复杂 XML 文件的一个基本单元来看待。后文将详细描述提取可作为映射关系表的 XML 结构的相关算法。

3.2 Schema 路径分析

XML Schema 为 XML 实例的约束性验证提供了很好的方式。这不仅仅是简单的数值校验等方面的检查,同时也定义了文件的层次路径结构的约束。通过分析 XML Schema,可以抽取制约 XML 存储的信息,即路径关系。下面将详细说明。

我们可以通过分析 Schema 建立一棵路径树,这棵树不同于 XML 文档实例构造出来的树。它是相对抽象的树形结构(见图3),(图3中 des3还有个属性 code 没有显示)。

首先,树中不含数据,即不含叶结点和属性结点的值,另外对于可重复结点加以标识。

请注意,树中的结点可以分为三种类型:

1. 被定义为可以重复出现的结点(可重复次数大于1),如 des3,des5。

2. 被定义为混合类型的结点,如 sub_tag2,图中没有说明。即它的值为 text 和子元素的混合。

3. 普通结点,除以上两种结点之外的结点,如属性结点,单独子元素结点等。

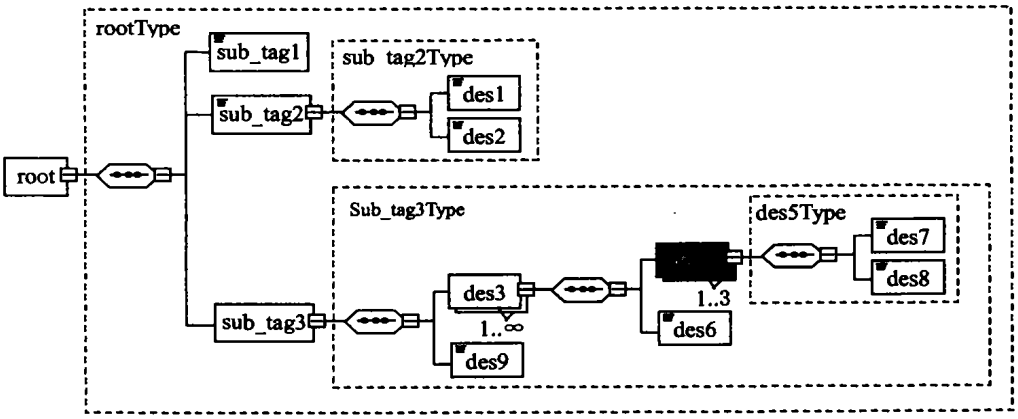

图3 一棵 Schema 路径树

我们看到这里需要发现可以作为列定义的路径,于是便存在一个路径划分的问题。路径划分的一个原则是尽可能选择长的路径作为列定义。拿图3来说,/root/sub_tag2/des1就是有效的列定义路径。而/root/sub_tag3相对于/root/sub_tag3/des9就不是最长的有效路径。

3.3 路径划分算法

令路径信息 PathInfo 为一个四元组 PathInfo=(p,pp,Type,SETID),其中 p 为当前路径,pp 为与当前路径连接的父路径,Type 为路径类型(C 为复杂路径,M 为混合路经,S 为简单路径),SETID 为该路径所属集合标识。

输入:Schema 路径树为 T;

中间变量:计数变量 i;起始结点 sn 和当前正在遍历的结点 cn;队列 Q 的每一个分量为一个二元组(node,path);当前路径集合标识 CP;当前父路径 cpp;

输出:路径信息集合 P(PathInfo∈P)。

算法步骤:

1. 初始化:置队列 Q,路径信息集合 P 为空;起始结点 sn 和当前结点 cn 均为 T 的根结点;i=0;cpp=null;

2. 将(sn,cpp)放入队列 Q;

3. 若队列 Q 为空,则算法结束。

否则,从队列 Q 中取出分量(node,path),其中 node 作为 sn,path 作为 cpp;从 sn 表示的结点起,深度优先遍历路径树,令当前路径集合标识为 CPi;

在首次访问某结点时:

3.1 若遇到类型1的结点,则设从 sn 到 cn 的路径为 p,设置四元组(cpp/p,cpp,C,CPi),并将此四元组添入 P;将(cn,cpp/p)放入队列 Q,遍历过程在此点回溯;

3.2 若遇到类型2的结点,则设从 sn 到 cn 的路径为 p;设置四元组(cpp/p,cpp,M,CPi),并将此四元组添入 P;

3.3 若一直遍历到叶结点都没有遇到类型1或2的结点,则设从 sn 到此叶结点的路径为 p,设置四元组(cpp/p,cpp,S,CPi);并将此四元组添入 P;

一次遍历结束时，i 增1，重新进行步骤3。

算法结束时，路径信息集合 P 中将记录所有可以作为有效列定义的路径。

结合上面的图例说明，我们可以得到如下的结果：

```
P={
(/root/sub_tag1,null,S,CP0),
(/root/sub_tag2,M,null,CP0),
(/root/sub_tag2/des1,null,S,CP0),
(/root/sub_tag2/des2,null,S,CP0),
(/root/sub_tag3/des3,null,C,CP0),
(/root/sub_tag3/des9,null,S,CP0),
(/root/sub_tag3/des3/@code,/root/sub_tag3/des3,S,CP1),
(/root/sub_tag3/des3/des5,/root/sub_tag3/des3,C,CP1),
(/root/sub_tag3/des3/des6,/root/sub_tag3/des3,S,CP1),
(/root/sub_tag3/des3/des5/des7,/root/sub_tag3/des3/des5,
S,CP2),
(/root/sub_tag3/des3/des5/des8,/root/sub_tag3/des3/des5,
S,CP2)
}
```

3.4 存储结构

有了路径分析的基础，就可以准备设计 XML Schema 的数据库存储结构了。在3.4中已经得到了路径信息集合 P 的结果。这个将成为存储结构的主要依据。(所有库表数据采用了上面的例子)。

3.4.1 *路径索引表* PathIndex Schema 需要一个路径索引表 PathIndex，来记录所有有用的路径信息。基本结构如下：

PathIndex：

编号	路径	表示	类型	前驱路径	子表名
1	/root/sub_tag1,	P1	S	Null	CP0
2	/root/sub_tag2	P2	M	Null	CP0
3	/root/sub_tag2/des1	P3	S	Null	CP0
4	/root/sub_tag2/des2	P4	S	Null	CP0
5	/root/sub_tag3/des3	P5	C	Null	CP0
6	/root/sub_tag3/des9	P6	S	Null	CP0
7	/root/sub_tag3/des3/@code	P7	S	P5	CP1
8	/root/sub_tag3/des3/des5	P8	C	P5	CP1
9	/root/sub_tag3/des3/des6	P9	S	P5	CP1
10	/root/sub_tag3/des3/des5/des7	P10	S	P8	CP2
11	/root/sub_tag3/des3/des5/des8	P11	S	P8	CP2

这张表可以认为是路径信息集合 P 的扩充。此表，实际上是 Schema 骨架结构的反映，通过它可以提取足够多的关于 Schema 的结构信息，并且反映出3.5.2中 CPi 表的列定义之间的关系。

3.4.2 *XML 数据的分散存储* 对 PathIndex 表的子表名进行划分，从 CP0到 CPn 总共可划分出 n+1个表，对 CP0 和 CPi (i>=1)分别处理有：

CP0：

文档编号	P1	P2	P3	P4	P5	P6

CP1：

文档编号	结点编号	P7	P8	P9

CP2：

文档编号	结点编号	P10	P11

说明：文档编号指每个独立完整的 XML 文档实例的编号(概念上如同关系元组的 rowid)；在全部 CP 表中，如果某个 Pi 的类型为 C 或 M，则它的值为一个结点编号，这个结点编号是在存储 XML 实例，分析其结构时自动产生的，下文还会有说明；在 CPi(i>=1)中有一列名为“结点编号”，它用来关联此表所有列的前驱路径所表示的文档实例的相应结点。

3.4.3 *混合类型数据的存储* Schema 本身在定义时，只采用了 mixed=true|false 一个关键字来说明如何表达混合数据的情况。比如：text1<a>atext2b与<a>a<./a>text2btext3在 Schema 看来并没有什么不同，Schema 本身不能确定 mixed 类型的文本数据可以出现的位置。

但是，在存储这些文本片断时候必须考虑位置这一情况，比如一篇新闻报道，被<a.>或标识的为需要突出重点的内容，其余的文本为正常的叙述，那么这些位于标签外的文本与标签内的文本之间的关系必须始终一致，文字在语义上才可能有意义。

考虑如下的表结构：

MixedText：

文档编号	路径表示	结点编号	值	前标签	后标签

在 PathIndex 中凡是路径类型被表示为 M 的，其文档实例如果出现了这样的文本片断，都将会在这个结构中有所记录。

比如：

<sub_tag2>hi,<des1>Mr. Li</des1>I have finished the <des2>article</des2>

则有：

CP0：

文档编号	P1	P2	P3	P4	P5	P6
1	…	3	Mr. Li	article	…	…

MixedText：

文档编号	路径表示	结点编号	值	前标签	后标签
1	P2	3	hi,	null	des1
1	P2	3	I have finished the	des1	des2

这里只需要分辨出文本前后的标签名就足够了。

3.4.4 *总体结构* 可以用图4表示上面那个 Schema 的存储结构。

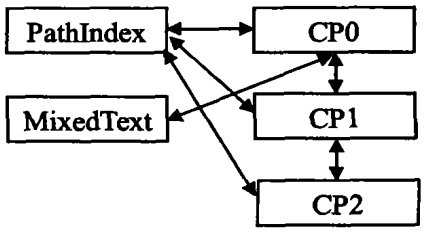

图4 存储结构关系

CPi 之间有层次关系，这些都由 PathIndex 表示，某些 CPi 还会与 MixedText 关联。

3.5 有待讨论的几个问题

首先，部分语义的问题。存储结构中列的定义是用完整路径表示的，如/r1/r2/a 与/r1/r2/r3/a 将被示为不同的路径。如果仅以 a 为查询条件，那么它到底要与哪个路径匹配，还是都匹配呢？问题本身与存储结构的关系不大，但是无论如何，最后的查询都会定位到完整路径上来，这也就需要查询的工作细化，或者要求用户说明，或者给出提示，或者采取缺省的做法，但最根本的是让查询在语义上而不仅仅是在语法上符合用户的想法，这可能会涉及到更多的工作。

另外，我们知道集合中的元素是无序的，作为关系表中的元组也反映了这个事实，不过顺序在 XML 的实际应用中可能是重要的，Schema 本身可以说明部分这样的特征，比如 sequence 关键字等，那么是否在存储片断时也要保证源文档的被要求部分的有序性呢？看来是需要的。最简单的做法可能是标号，不论是路径也好，实例结点也好，在内部实现时都可以为它们编号，并保证编号较小的一定先出现。这也包括处理了一些不能被 Schema 定义，但实例中又确实希望有序的情况，比如第一个 Location 标签可能会被认为比第二个 Location 标签更重要些，但在没有 Schema 约束的情况下，这些也只能是纯属猜测。尽管如此，在不甚了解语义的情况下，尽可能地保持原来的结构，也是一种比较稳妥的做法。

结束语 本文通过对 Schema 的路径分析，针对每一个 Schema 建立它自己的尽可能精简的 CPi 系列存储结构。并且将 XML 的数据分为片断存储，将不同 XML 文档实例的相同路径信息片断存储于一张关系表中，这样不仅为抽取 XML 的片断信息（或组件信息）带来了方便，而且可以对任何标签的内容进行数据库索引，在搜索指定标签内容的 XML 时，效率将极大地提高，同时也为标签组件的复用带来方便。

参 考 文 献

1 World Wide Web Consortium XML Schema Recommendation, May. 2001. http://www.w3c.org/XML/Schema

2 World Wide Web Consortium XML Path Language (XPATH)1.0 Recommendation, Nov. 1999. http://www.w3c.org/TR/xpath

3 Tatarinov I, Viglas S D, Beyer K, Shanmugasundaram J, Shekita E, Zhang C. Storing and Querying Ordered XML using a Relational Database System. SIGMOD 2002

4 Schmidt A R, Kersten M L. Bulkloading and Maintaining XML Documents. In: Proc. of the ACM Symposium on Applied Computing (SAC 2002), Madrid, Spain, March 2002. 407～412

5 Yoon J P, Raghavan V, Chakilam V. Bit Cube: A Tree-Dimensional Bitmap Indexing for XML Documents, SSDBM2001

6 Süß C, Zukowski U, Freitag B. Data Modeling and Relational Storage of XML-based Teachware. In: Proc. Informatik 2001, Wien 2001

7 Chantal Reynaud Jean-Pierre Sirot Dan Vodislav, Semantic Integration of XML Heterogeneous Data Sources. In: 2001 Intl. Database Engineering & Applications Symposium

8 Yang Xiaochun, Yu Ge, Wang Guoren. Efficiently mapping integrity constraints from relational database to xml document, Advances in Databases and Information Systems. In: 5th East European Conf. ADBIS 2001 Proceedings

（上接第68页）

询转换技术进行了一些研究。但这些研究还是远远不够的，还有许多问题没有解决。例如，并不是所有的 XML 查询功能都能够转换为 SQL 查询来实现；如何有机地实现 XML 数据的查询与关系数据的查询的有机集成；在关系查询引擎下如何利用模式信息和统计信息来选择 XML 查询的执行方案、执行算法、优化策略等；关系查询引擎在实现某些 XML 查询时性能还不是很理想，如何扩充关系查询引擎的功能以便更有效地处理 XML 查询；等等。因此，如何利用关系数据库来有效地实现 XML 查询是该领域今后研究的重点。

参 考 文 献

1 Bray T, Paoli J, Sperberg-McQueen C. Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation. Oct. 2000. http://www.w3.org/TR/REC-xml

2 World-Wide Web Consortium: XQuery 1.0 and XPath 2.0 Data Model. Working Draft, Dec. 2001. http://www.w3.org/TR/2001/WD-qury-datamodel-20011220.

3 Deutsch A, Fernandez M, Florescu D, et al. XML-QL: A Query Language for XML. 1998. http://www.w3.org/TR/NOTE-xml-ql/.

4 World-Wide Web Consortium: XQuery 1.0: A XML Query Language. Working Draft, Dec. 2001. http://www.w3.org/TR/2001/WD-xqury-20011220.

5 Florescu D, Kossmann D. Storing and querying XML data using an RDBMS. IEEE Data Engineering Bulletin, 1999, 22(3): 27～34

6 Florescu D, Kossmann D, Manolescu I. Integrating keyword search into XML Query processing. WWW9/Computer Networks, 2000, 33(1-6): 119～135

7 Shanmugasundaram J, Tufte K, He G, et al. Relational Databases for Querying XML Documents: Limitations and Opportunities. In: Proc of the 25th VLDB Conf. Scotland, Sep. 1999

8 郑仕辉，周傲英，等. 基于 SQL 的 XML 查询的有效实现. 计算机研究与发展，2001，38(4)：422～429

9 Li Q Z, Moon B. Indexing and Querying XML Data for Regular Path Expressions. In: Proc of the 27th VLDB Conf. Roma, Italy, 2001. 361～370

10 Zhang C, Naughton J, DeWitt D, et al. On Supporting Containment Queries in Relational Database Management Systems. In: Proc of the ACM SIGMOD Conf, Santa Barbara, California, May 2001

11 Wan Chang-xuan, Liu Yun-sheng. Efficient Supporting XML Query and Keyword Search in Relational Database Systems. In: Proc of the third Int'l Conf on Web-Age Information Management (Springer-Verlag LNCS 2419), Beijing, China, Aug. 2002. 1～12

12 Shanmugasundaram J, et al. Efficiently Publishing Relational Data as XML Documents. In: Proc of the 26th VLDB Conf. Cairo, Egypt, Sept. 2000

13 Fernandez M, Tan W, Suciu D. SilkRoute: Trading Between Relations and XML. In: Proc of the 9th Int' l WWW Conf. Amsterdam, May 2000

14 Carey M, Kiernan J, et al. XPERANTO: A Middleware for Publishing object-relational data as XML Documents. In: Proc of the 26th VLDB Conf, Cairo, Egypt, Sept. 2000

15 Carey M, Florescu D, Ives Z, et al. XPERANTO: Publishing object-relational data as XML. In WebDB Workshop, in conj. With ACM SIGMOD, 2000

16 Shanmugasundaram J, Kiernan J, Shekita E, et al. Querying XML Views of Relational Data. In: Proc of the 27th VLDB Conf, Roma, Italy, 2001

17 Paul F. Dietz: Maintaining order in a linked list. In: Proc of the 14th Annual ACM Symposium on Theory of Computing, San Francisco, California, May 1982. 122～127

18 Blake G E, McGill M J. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983