第 33 卷 第 5 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.5 Computer Engineering March 2007
·软件技术与数据库· 文章编号：1000—3428(2007)05—0053—03 文献标识码：A 中图分类号：TP311.13

基于 O-D 的 XML 编码及对信息查询与更新的支持

曹耀钦，宋建社，赵 霜，余 宁
(第二炮兵工程学院计算机系，西安 710025)

摘 要：提出了一种基于区间表示的 XML 编码方式，设计了相应的关系存储模式，研究了该编码方式对信息查询和信息更新的支持。鉴于 XML 关系存储下信息更新的困难性，采用基于 order-descendan(O-D)的 XML 编码方式，不仅可以完全支持 XPath 查询语言的 13 个查询轴的查询功能，而且通过简单的计算即可有效地支持 XML 数据的增加和删除，这种简单计算可以在关系存储模式下通过 SQL 语句方便地实现。

关键词：XML；编码；关系存储；信息查询；信息更新

Order-descendant-based XML Encoding and Support to Info-query and Info-updating

CAO Yaoqin, SONG Jianshe, ZHAO Shuang, YU Ning
(Department of Computer, Second Artillery Engineering College, Xi'an 710025)

【Abstract】An XML encoding based on region-presentation is proposed, correspondingly relational store model is designed, and support to info-query and info-update is researched. Because of the difficulty of info-updating in XML relational storage, an XML encoding based on Order-descendant is applied, which supports not only querying function of 13 query-axes in XPath query language, but also adding and deleting operation of the XML data by simple calculation. This calculation can be realized conveniently in relational store model with SQL language.

【Key words】XML; Encoding; Relational storage; Info-query; Info-updating

近年来，XML已逐渐成为分布计算与业务应用中数据表示及数据交换的标准格式[1～3]。为发挥XML的潜能，XML文档数据的有效存储是必须研究解决的关键技术。由于XML数据的半结构化性、XML的文档/数据两重性和多用途性、以及XML在不同应用领域的特殊要求，因此XML的存储问题至今难以找到一种通用的“ 最佳 ”解决方案[3]。这正是导致以XML存储为核心的“ XML数据库 ”至今尚无定论的关键所在[1,3]。目前的解决方法可归结为两大类[3]：(1)在已有的关系数据库或面向对象数据库基础上扩充相应的功能，使其能够胜任XML的数据处理工作，称之为XML使能数据库；(2)为XML数据量身定做的数据库即纯XML数据库。纯XML数据库充分考虑到XML数据的特点，以一种自然的方式来处理XML数据，能够从各方面很好地支持XML的存储和查询，并且能够达到较好的效果，但是，纯XML数据库要走向成熟还有很长的路[3]。由于关系数据库技术已相当成熟，同时基于各种因素综合考虑，军用网格采用了第 1 种方法，即在关系数据库下实现XML资源信息的管理。

1 XML 文档信息的 O-D 编码方案

为了有效支持XML查询，必须研究XML文档的各种对象流(如各种标记名、属性信息)的编码方案及索引技术。目前，XML数据的编码方案主要分为两大类：一种是基于区间的编码方案，主要有Dietz编码[4]、Li-Moon编码[5]、Zhang编码[6]、Wan编码[3]等，这些方案利用XML文档的有序特点，根据每一个元素结点在原XML文档中字典顺序的位置给每个元素赋予一个编码；另一种是基于路径的编码方案，主要有前缀编码[7]、位向量编码[8]等。与基于区间的编码方案不同，这些方案则是利用XML文档的嵌套特点，根据XML文档的嵌套结构，给从根结点开始所能到达的每个路径和元素结点赋予一个编码。这些方案在信息的存储、查询方面各有优点，但一个明显的缺点就是信息的更新极为不便。由于编码方式大多采用结点的某种次序(如访问顺序)作为编码的一部分，当在某结点前面插入一个新结点时，相应结点的访问顺序必然发生变化，因而其后续结点的所有编码都必须随之改变。因此，设计一种合适的编码方案，当结点需要插入或删除时，只需要经过简单计算就可以完成信息更新，成为XML编码方案研究的重点。一般情况下，在设计XML编码时应考虑以下因素：

(1)被编码数据的结构；
(2)支持祖先/子孙、双亲/孩子、文档位置等关系的结构查询；
(3)编码算法的复杂度；
(3)编码的最大长度或平均长度；
(4)编码后的查询执行时间；
(5)插入操作导致的重新编码代价。

根据以上原则，本文提出了一种基于先序遍历号和其子孙数相结合的 XML 编码方案。下面首先给出有关定义，然后讨论该编码方案对各种运算的支持。

定义 1 设 XML 文档树是 T，结点集合 V={V1, V2, …, Vn}，基于先序遍历号和子孙数的区间编码方案为一个二元组<order，num>，其中，order 是结点的先序遍历编号；num

基金项目：国家自然科学基金资助项目(60272022)
作者简介：曹耀钦(1962－)，男，在职博士生，主研方向：网络信息系统，网格计算；宋建社，博士、教授、博导；赵 霜、余 宁，硕士生
收稿日期：2006-03-20 **E-mail：**yqcao2004@tom.com

是以本结点为根结点的子树的所有结点数，即该结点的子孙数(含自身结点)。

为了叙述方便，下面将该编码方案称为 Order-descendant 方案，简称 O-D 编码方案。

为了说明本文提出的编码方案，假设给定一个 XML 文档树，如图 1 所示。图中结点的编号是按照先序遍历得到的。

O-D 编码方案与其它编码方案的比较如图 2 所示。其中前 4 个编码取自文献[3]。从图 2 可知，在 O-D 编码方案中，当结点的 num 为 1 时，表示该结点是叶子结点。当不是叶子结点时，其 num 值就是其子孙数(含自身结点)，意义非常直观。不仅如此，该方案还可以支持关系数据存储、支持 XPath 和 XQuery 查询、可以方便地实现 XML 信息更新。

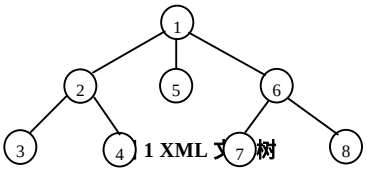

图 1 XML 文档树

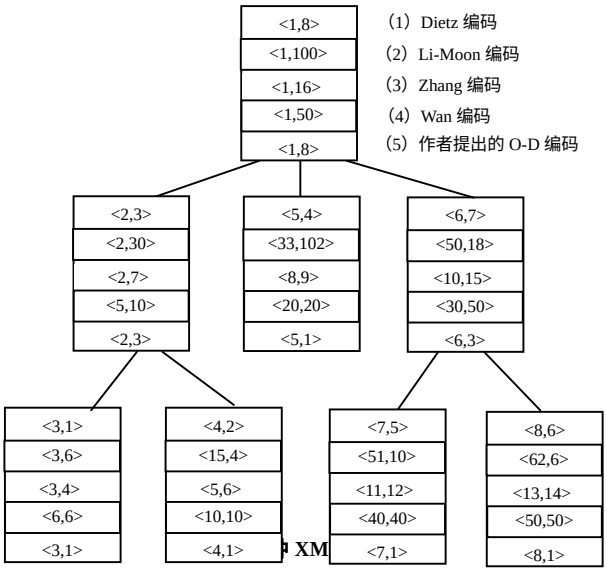

图 2 XML

2 XML 文档的关系存储模式

XML 文档的关系存储模式实质上就是采用关系数据库实现 XML 文档信息的存储、查询及更新操作。其核心是设计合适的关系模式，将文档解析后得到的有关信息存储到关系数据库中。

(1)对资源文档集合建立一个索引表，该索引表的每个记录是一个三元组(docID, URL, ResourceType)。其中 docID 是该结点所在文档的文档标识；URL 是文档资源的地址；ResourceType 是资源的类型，其中 CT 表示计算资源，CM 表示通信资源，HS 表示软件资源，HW 表示共享设备资源。

(2)对于每个标记相同的结点建立一个元素索引表，该索引表的每个记录是一个七元组(docID, order, num, parent- order, nodeType, depth, name)。其中，docID 是该结点所在文档的文档标识；order 是该结点在文档树中的先序遍历号；num 表示结点的子孙数；parentorder 是其父结点的编号；nodeType 是结点的类型，其值取 EE、ET 和 EN，分别代表元素结点的内容是子元素、文本和空；depth 表示该结点在文档树中所位于的层次；name 表示结点名。

(3)对 XML 文档树中的所有标记名建立一个索引表，该索引表的每个记录是一个四元组(docID, order, Tagname, ConType)。其中，docID 是该结点所在文档的文档标识；order 是该结点的编号；Tagname 是该结点的标记；ConType 是该结点的内容类型。

(4)对于所有文本结点和属性结点，需要建立一个值索引表，该表的每个记录是一个四元组(docID, order, value, valueType)。其中，docID 是该结点所在文档的文档标识；order 是该结点的编号；value 是以字符串形式存放的文本或属性结点的值；valueType 用来说明属性值或文本内容的原始数据类型。

据此，可设计如表 1 所示的 XML 文档的关系存储模式。

表 1 XML 文档的关系存储模式

Document(docID,URL, ResourceType)
Tagname(docID, order,Tagname,ConType)
Value(docID, order,value,valueType)
Elem-*Tagname*(docID, order, num, parentorder, nodeType,depth, name)

3 基于关系存储的 XML 信息查询

XPath 是一个 XML 查询语言，它用来描述在一个 XML 文档树中的查询导航，它是 W3C 所推荐的一种相对简单的查询语言。同时，XPath 也是强制的，各种 XML 查询语言都必须支持它。事实上，XPath 也正是各种查询语言的核心，例如 XQuery 就是基于 XPath 进行扩展的一个查询语言。

针对查询功能，XPath定义了 13 个查询轴[3]，在XML文档中实现相关信息的查询。这 13 个查询轴分别是：

(1)child 轴：选择上下文结点的孩子。

(2)descendant 轴：选择上下文结点的子孙。

(3)parent 轴：选择上下文结点的父结点。

(4)ancestor 轴：选择上下文结点的祖先。

(5)following-sibling 轴：选择上下文结点所有在其后(右)的结点。

(6)preceding-sibling 轴：选择上下文结点所有在其前(左)的结点。

(7)following 轴：选择上下文结点所有在其后且排除其所有子孙的结点。

(8)preceding 轴：选择上下文结点所有在其前且排除其所有祖先的结点。

(9)attribute 轴：选择上下文结点的属性。

(10)namespace 轴：选择上下文结点的命名空间。

(11)self 轴：选择上下文结点自身。

(12)descendant-or-self 轴：选择上下文结点自身及其子孙。

(13)ancestor-or-self 轴：选择上下文结点自身及其祖先。

O-D 编码方案可以全面支持 XPath 查询，即支持 XPath 的 13 个查询轴。具体见表 2 所示(表中未列出 self 轴和 namespace 轴)。

从表 2 可知，O-D 编码方案通过简单的计算即可实现 XPath 查询语言的查询功能。

表 2 O-D 编码方案的 XPath 查询轴的计算判别条件

序号	XPath 查询轴	O-D 编码支持
1	child(u)或 attribute(u) {v|v 是 u 的孩子} 或{v| v 是 u 的属性}	order(u)= parentorder (v)
2	descendant(u) {v|v 是 u 的子孙}	order(u)+num(u)≥order(v) 且 order(u)<order(v)
3	descendant-or-self(u) {v|v 是 u 的子孙或自身}	order(u)+num(u)≥order(v) 且 order(u)≤order(v)
4	following(u) {v|v 位于 u 之后}	order(v)>order(u)+num(u)
5	following-sibling(u) {v|v 是 u 的右兄弟}	order(v) >order(u) +num(u) 且 parentorder(u)=parentorder(v)
6	parent(u) {v|v 是 u 的父结点}	order(v)=parentorder(u)
7	ancestor(u) {v|v 是 u 的祖先}	order(v) < order(u)且 order(u)≤order(v)+num(v)
8	ancestor-or-self(u) {v|v 是 u 的祖先或自身}	order(v)≤order(u)且 order(u)≤order(v)+num(v)
9	preceding(u) {v|v 位于 u 之前}	order(v)+num(v)< order(u)
10	preceding-sibling(u) {v|v 是 u 的左兄弟}	order(u) >order(v)+num(v) 且 parentorder(u)=parentorder(v)

4 关系存储下的 XML 信息更新

关系存储下的 XML 信息更新问题一直是 XML 关系存储的难题。有关 XML 关系存储和查询方面的研究成果较多，但涉及到信息更新方面的论文则鲜见。其原因是采用关系存储时结点需要进行编码，而无论增加或删除结点则编码都要进行相应变化，可见编码方式对更新操作影响极大。而本文提出的 O-D 编码方式则可以较好地支持关系存储下的信息更新。

4.1 关系存储下的 XML 信息增加

当文档树中需要增加新的结点时，可分为以下几种情况：

(1)将结点 v 增加在结点 u 之后，u 在 XML 文档树中是一个叶子结点，即 num(u)=1。

在这种情况下，可通过以下步骤实现：

1)结点 u 的子孙数加 1，即 num(u)=num(u)+1；

2)结点 v 的编号为 order(v)=order(u)+1，结点 v 的子孙数为 1，即 num(v)=1；结点 v 的双亲结点为 u，即 parerntorder(v)=order(u)；结点 v 在文档树中的层次为 depth(u)+1。

3)文档树中所有比结点 u 编号大的结点 p，其编号加 1，即 order(p)=order(p)+1；

4)结点 u 的所有祖先结点的子孙数(num)全部加 1。根据表 2，可知结点 u 的所有祖先 x 满足 order(x)> order(u)且 order(u)≤order(x)+num(x)，也就是当结点 x 满足 order(x)> order(u)且 order(u)≤order(x)+num(x)时，num(x)=num(x)+1。

(2)将结点 v 增加在结点 u 之后作为一个叶子结点，u 在 XML 文档树中不是叶子结点，即 num(u)≠1。

在这种情况下，可通过以下步骤实现:

1)结点 u 的子孙数加 1，即 num(u)=num(u)+1；

2)结点 v 的编号为 order(v)=order(u)+num(u)，结点 v 的子孙数 num(v)=1；结点 v 的双亲结点为 u，即 parerntorder(v)=order(u)；结点 v 在文档树中的层次为 depth(u)+1。

3)结点 u 的所有祖先结点的子孙数(num)全部加 1。根据表 2，可知结点 u 的所有祖先 x 满足 order(x)> order(u)且 order(u)≤order(x)+num(x，也就是当结点 x 满足 order(x)> order(u)且 order(u)≤order(x)+num(x)时，num(x)=num(x)+1。

4)文档树中所有比结点 u 编号大的结点 p，其编号加 1，即 order(p)=order(p)+1。

(3)将结点 v 增加在结点 u 之后其某个孩子 w 之前。

在这种情况下，可通过以下步骤实现:

1)结点 u 的子孙数加 1，即 num(u)=num(u)+1；

2)结点 v 的子孙数 num(v)=num(w)+1；结点 v 的双亲结点为 u，即 parerntorder(v)=order(u)；结点 v 在文档树中的层次为 depth(u)+1；结点 v 的编号为 order(v)=order(w)；

3)结点 w 的双亲结点为 v，即 parerntorder(w)=order(v)；结点 w 在文档树中的层次为 depth(w)+1；结点 w 的编号为 order(w)=order(v)+1；

4)结点 u 的所有祖先结点的子孙数(num)全部加 1。根据表 2，可知结点 u 的所有祖先 x 满足 order(x)> order(u)且 order(u)≤order(x)+num(x)，也就是当结点 x 满足 order(x)> order(u)且 order(u)≤order(x)+num(x)时，num(x)=num(x)+1。

5)文档树中所有比结点 w 编号大(包括 w)的结点 p，其编号加 1，即如果 order(p)>order(w)，则 order(p)=order(p)+1；

6)结点 w 的所有子孙结点的层数+1。根据表 2，可知结点 w 的所有子孙 x 满足 order(w)+num(w)≥order(x)且 order(w)<order(x)，也就是当结点 x 满足 order(w)+num(w)≥order(x)且 order(w)<order(x)时，depth(x)= depth(x)+1。

4.2 关系存储下的 XML 信息删除

当文档树中需要删除结点为 v，其双亲结点为 u 时，可分为以下几种情况：

(1)删除的结点 v 是一个叶子结点，即 num(v)=1。

在这种情况下，可通过以下步骤实现：

1)结点 v 的双亲结点 u 的子孙数减 1，即 num(u)=num(u)-1；

2)文档树中所有比结点 v 编号大的结点 p，其编号减 1，即 order(p)=order(p)-1；

3)结点 u 的所有祖先结点的子孙数(num)全部减 1。根据表 2，可知结点 u 的所有祖先 x 满足 order(x)<order(u)且 order(x)+num(x)≤order(u)，也就是当结点 x 满足 order(x)< order(u)且 order(u)≤order(x) +num(x)时，num(x)=num(x)-1。

(2)删除的结点 v 不是叶子结点，即 num(v)≠1。

在这种情况下，可通过以下步骤实现：

1)结点 u 的子孙数减 1，即 num(u)=num(u)-1；

2)结点 v 的所有子孙结点的层数-1。根据表 5 中 6，可知结点 v 的所有子孙 x 满足 order(v)+num(v)≥order(x)且 order(v)<order(x)，也就是当结点 x 满足 order(v)+num(v)≥order(x)且 order(v)<order(x)时，depth(x)= depth(x)-1。

3)文档树中所有比结点 v 编号大的结点 p，其编号减 1，即 order(p)=order(p)-1。

4)结点 v 的孩子结点为 x，其双亲结点修改为 order(u)，即如果结点 x 满足：order(v)= parentorder (x)，则 parentorder (x)= order(u)。

从上述分析可知，当采用 O-D 编码方式时，可以通过简单的计算即可实现 XML 资源信息的更新。上述计算过程可以方便地用 SQL 语句实现。

4.3 信息更新的 SQL 语言实现

下面以关系存储下的 XML 信息删除的第 1 种情况为例说明 SQL 的实现程序。

```
//获得结点 v 的编号及双亲结点编号
select   order，parentorder
into :vorder，:vparentorder
from B
where Name='v';
//删除结点 v
delete   from   B
where   order=:vorder;
//将结点 u 的子孙数-1
update B
set num= num-1
where order=: vparentorder ;
//将所有编号大于 v 的结点的编号-1
update B
set order= order-1
where order> :vorder;
//结点 u 的所有祖先结点的子孙数(num)全部减 1
update B
set num= num-1
where order<: vparentorder and : vparentorder < order+num;
```

上述程序中，:vorder 和:vprarentorder 是共享变量，需要在主程序中定义。同理，也可写出其它情况下所对应的更新程序。

5 结束语

本文提出了一种基于先序编号和子孙数的 XML 编码方案，该方案具有结构清晰、意义明确等特点，可以较好地支持 XML 查询和信息更新，已经应用到某军事应用网格中。

（下转第 58 页）