基于关系数据库的XML数据管理

万常选[1,2] 刘云生[1]

(华中科技大学计算机科学与技术学院 武汉430074)[1] (江西财经大学信息管理学院 南昌330013)[2]

An Overview of XML Data Management Based on an RDBMS

WAN Chang-Xuan[1,2] LIU Yun-Sheng[1]

(School of Computer Science & Technology, Huazhong University of Science & Technology Wuhan 430074)[1]

(School of Information & Technology, Jiangxi University of Finance & Economics Nanchang 330013)[2]

Abstract Currently, there are a great of research topics that focus on storing and querying XML data in an RDBMS, and publishing relational data as XML documents, and querying XML views of relational data. An overview of XML data management based on RDBMS is given in this paper. Some existing technologies of storing and querying XML data in relational databases, publishing relational data as XML documents, and querying XML views of relational data are sufficiently surveyed, their advantages, disadvantages, and causes are analyzed.

Keywords XML documents, Relational storage, XML queries, Relational data publishing

1 引言

XML[1]正迅速成为Web上数据表示和数据交换的标准,可以预期在不久的将来Web上将会有大量的XML数据出现。1998年,W3C制定了XML数据规范,引入XML的基本想法非常简单:为了确认数据的含义而标记数据元素,XML标记的作用是描述数据自身,而并不是具体说明数据应该如何进行版式设计(像HTML标记的主要目的是描述怎样显示一个数据项);以简单的元素嵌套和引用来表示元素之间的关系。XML的这种变化对Web应用有重要的影响:接受XML文档的应用程序能够用不同的方式解释数据;能够基于它的内容和结构对文档进行过滤;为了适当的应用需要能够重构数据;Web服务器和应用通过用XML来表示它们的信息,能够以一种简单的可用的格式迅速地将它们的信息变为可用,而且这种信息的提供者之间能够很容易地进行相互操作。

图1和图2分别是一个包含出版物信息的XML文档实例片断和它所遵守的DTD描述片断。

```
〈publication〉
  〈book year="2000"〉
    〈title〉Database System Concepts〈/title〉
    〈author id="101"〉
      〈name〉A. silberschatz〈/name〉
      〈email〉silb@research. bell-labs. com〈/email〉
    〈/author〉
    〈author id="102"〉
      〈name〉H. F. Korth〈/name〉
    〈/author〉
  〈/book〉
  〈book year="2001"〉
    〈title〉Introduction to XML〈/title〉
    〈author id="103"〉
      〈name〉M. Morrison〈/name〉
      〈email〉office-same@mcp. com〈/email〉
    〈/author〉
  〈/book〉
  〈article editorID="105"〉
    〈title〉A Query Language for XML 〈/title〉
    〈author id="104"〉
      〈name〉M. Morrison〈/name〉
    〈/author〉
  〈/article〉
  〈editor id="105"〉
    〈name〉A. Deutsch〈/name〉
  〈/editor〉
〈/publication〉
```

图1 一个XML文档实例

```
〈!ELEMENT publication(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,email?)〉
〈!ATTLIST author id ID #REQUIRED〉
〈!ELEMENT editor (name,email?)〉
〈!ATTLIST editor id ID #REQUIRED〉
〈!ELEMENT title (#PCDATA)〉
〈!ELEMENT price (#PCDATA)〉
〈!ELEMENT name (#PCDATA)〉
〈!ELEMENT email (#PCDATA)〉
```

图2 一个XML DTD实例

XML起初主要是为了增强应用程序对从Web上获得的文档的解释和操作能力。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:对于存储在XML文档中的数据,应该能够查询这些文档中的内容和结构。为了有效地查询XML数据,已经提出许多查询语言,例如,XPath、XML-QL[3]、Quilt和XQuery[4]等。这些查询语言的共同特点是利用正则路径表达式(Regular Path Expression, RPE)来导航XML文档的查询,以便抽取、合成和分析这些文档的内容。那么什么方式能够提供这种XML文档查询的最好查询性能呢?查询性能与存储结构和存储方式密切相关,通常XML数据的存储和查询有四种不同的选择,一是利用文件系统,二是利用关系数据库系统,三是利用面向对象的数据库系统,四是建立一个特殊的数据库系统(或称为本地的XML数据库系统)。采用文件系统来存储XML文档是最方便的,但是文件系统对XML文档的查询没有提供任何支持,

*)本课题得到江西省教育厅科技项目(赣教计字[2001]387号)资助。万常选 教授,博士研究生,主要研究方向:现代数据库技术、Web数据管理及电子商务技术。刘云生 教授,博士生导师,主要研究方向:现代数据库理论与技术及其集成实现、数据库与信息系统开发。

因此它不能充分发挥 XML 的潜能。面向对象数据库系统允许将 XML 元素和子元素进行聚集，这个特点对于某些应用来说可能是有用的，但是面向对象数据库系统对于复杂查询和大数据库还不够成熟。由于 XML 文档是一种特殊的半结构化数据，因此采用特殊的数据库系统（如 Lore、Tamino 等）来存储和查询 XML 数据好像就成了一种自然的选择，而特殊的数据库系统要走向成熟还有更长的路要走。相比之下在商业领域已得到广泛应用的关系数据库系统不仅成熟，而且具有很强的查询能力，因此，为了充分发挥二十多年来已投巨资研究和开发并已发挥巨大效益的关系数据库技术的优势，有许多研究者已经或正在从事有效地转换 XML 数据到（或从）关系数据库系统的研究[5~16]。

2 基于关系的 XML 数据存储与查询

对于利用关系数据库系统来存储和查询 XML 文档也有许多不同的方法和策略。一种策略是要求用户或系统管理者来设计用于存储 XML 数据的关系表结构，对于关系表中的数据可以直接以 XML 文档的方式进行发布[12]；也可以由用户或系统管理者使用 XML 查询语言或中间件提供的语言来定义该关系系统所对应的 XML 视图[13~16]，这样，其它应用就可以利用 XML 查询语言在虚的 XML 视图上构造一个查询，抽取 XML 视图中的数据片断并对抽取的部分进行物化，实现将关系数据转换为 XML 文档。

第二种策略是将一个 XML 文档看着是一个有序有向边标记图，设计一个（或若干个）关系来存储其边信息（称为边表），再设计一个（或若干个）关系来存储图的结点值（指叶结点值，称为值表），或直接将值表内联（Inlining）到边表中去。这些基本的存储模式已经在文[5]中进行了研究。

第三种策略就是从 XML 文档的 DTDs 或 Schema 来推断 XML 元素应该怎样映射到关系表，这种方法在文[7]中已经被研究。

第四种策略是对 XML 文档树[2]中的所有组件（包括元素结点、属性结点、文本“词”等）按某种遍历顺序（如先序遍历顺序）进行编码和存储，这种编码原则反映了这些组件在 XML 文档树中固有的相互关系（即祖先-后裔关系、双亲-孩子关系），以实现对 XML 文档的有效查询，这种方法在文[9~11]中已经进行了研究。

2.1 基于 XML 图的关系存储策略

一个 XML 文档能够用一个有序有向边标记图（称为 XML 图）来表示[5]。在这种图中，每一个 XML 元素被一个结点所表示，结点被标上 XML 对象的 oid；元素与子元素（或属性）之间的关系被图中的边所表示，并在边上标上子元素（或属性）名；为了表示 XML 元素中各子元素的顺序，可以对图中从某结点引出的所有边进行排序；XML 文档中的值作为图中叶结点（即属性或最底层子元素结点）表示。图3显示的是一个 XML 文档（见图1）的 XML 图，其中，标有数字的圆圈为非叶结点，数字为结点 oid；标有以“v”开头数字的圆圈为叶结点（即值结点），圈中符号为结点 oid（称为 vid）；每条边的名称后面的数字为边的序号；4号结点的 reviewer 属性为 IDREF 类型，它直接指向5号结点。

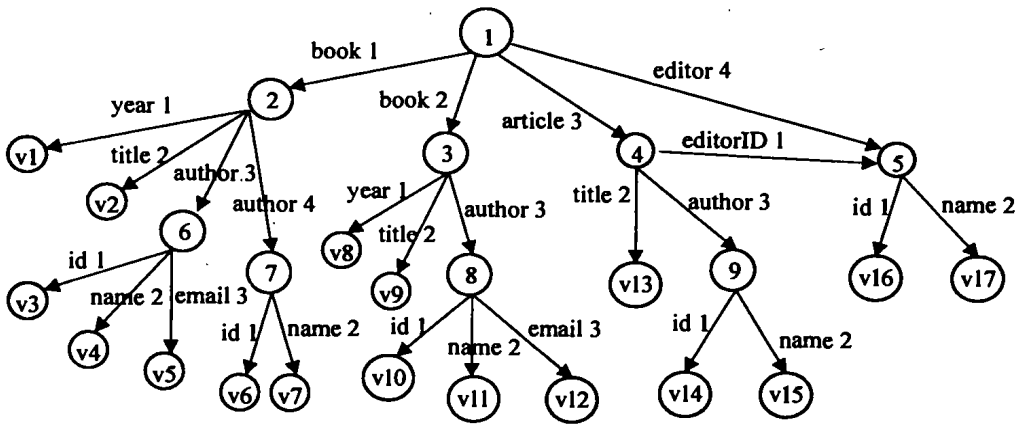

图3 XML 图实例

有了 XML 图后，就可以分别设计关系表来存储 XML 文档的边信息和值。对于用来存储边信息的边表有三种设计方案：第一种是用一个表来存储图的所有边信息，这种方法称为 Edge 方法；第二种是所有具有相同名称的边存放在一个边表中，这种方法称为 Binary 方法；第三种是采用一个边表来存储图中所有路径的边信息，该方法称为 Universal 方法。三种边表设计方案的性能比较是：Binary 边表方法优于 Edge 边表方法，Edge 边表方法又优于 Universal 边表方法。下面就 Edge、Binary 边表的设计思想进行简单介绍。

Edge 边表的关系模式为：Edge（source，ordinal，name，flag，target）。其中，source 和 target 属性分别用来存储边的引出结点和引入结点（即指向结点）对象的 oid，ordinal 属性用来存储该边在它的引出结点所引出的所有边中的排序号，name 属性用来存储边的名称（即该边所指向结点的名称），flag 属性用来存储该边所指向结点的类型。结点的类型分为两类：叶结点和非叶结点，对于叶结点的类型分别为 integer、string 等，分别表示叶结点的值为整数型、字符串型等；对于非叶结点的类型均为 ref。图4显示的是图3所示 XML 图所对应的 Edge 方法的表边。

Edge

source	ordinal	name	flag	target
…	…	…	…	…
2	1	year	integer	v1
2	2	title	string	v2
2	3	author	ref	6
2	4	author	ref	7
…	…	…	…	…
4	1	editorID	ref	5
4	2	title	string	v13
…	…	…	…	…

图4 Edge 方法的边表

$Value_{integer}$

vid	value
v1	2000
v3	101
v6	102
v8	2001
v10	103
v14	104
v16	105

$Value_{string}$

vid	value
v2	Database System
v4	A. Silberschatz
…	…

图5 分离值表

Binary 边表与 Edge 边表的原理相同，只是将所有具有相同名称的边存放在一个单独的边表中，Binary 边表的关系模式为：Binaryname(source，ordinal，flag，target)。概念上说，Binary 方法的边表是 Edge 方法中使用的边表的水平分割。

对于用来存储 XML 文档值的值表有两种设计方案：第一种是为每一种可能的取值类型设计一个值表，该方法称为分离值表；第二种是不单独设计值表，将值和边存储在同一个表中，这种方法称为内联方法，因此，对于每一种数据类型需要一个属性。

分离值表的关系模式为：$Value_{type}$(vid，value)。其中 vid 属性用来存储叶结点的 oid 值，value 属性用来存储叶结点的取值，它的类型与取值类型相同。图5显示的是图3所示 XML 图的分离值表。

三种边表设计方案，二种值表设计方案，合在一起一共有六种存储模式。文[26]对这六种基本的存储模式的结果关系数据库大小、执行不同类型的 XML 查询的执行时间、从关系数据重构 XML 文档的时间等三个性能参数进行了量化分析，结论是：Binary 边表方法优于 Edge 边表方法，Edge 边表方法又优于 Universal 边表方法；内联值表方法优于分离值表方法；Binary 边表带内联值表的存储模式能获得最好的综合性能。Edge 方法和 Universal 方法查询性能不好的主要原因是边表太大，因此执行连接操作非常费时。事实上，对于一个给定查询来说，边表中存放的数据大部分是不相关的。换句话说，在 Binary 方法中仅仅相关的数据被处理，即使卷入大量的连接和回路操作，由于现代关系查询引擎有非常强的查询处理能力，也能获得很好的查询性能。内联值表方法优于分离值表方法的主要原因是内联方法无需进行边表与值表之间的连接操作。即使是对于许多不同的数据类型，由于在关系数据库系统中空值通常是以一种紧缩的方式进行存储，空间就节省了，解决了大量冗余的问题。

2.2 根据 DTDs 映射关系模式的存储策略

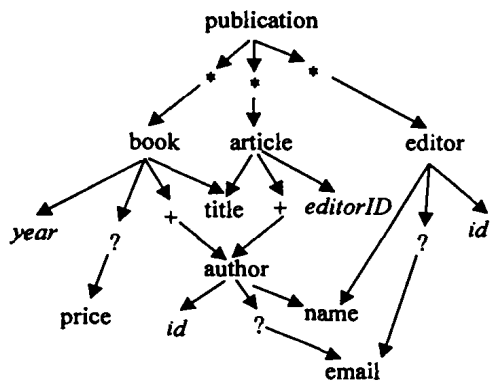

图6 DTD 图实例

根据 DTDs 映射关系模式的存储策略，首先需要对 DTD 进行适当地简化，然后产生 DTD 图及元素图，第三是根据 DTD 图或元素图生成关系模式，最后是将符合该 DTD 的 XML 文档数据装入关系数据库中。

一个 DTD 图表示一个 DTD 的结构，图的结点是 DTD 中的元素、属性以及正则路径运算符，每一个元素在图中出现且仅出现一次，属性和操作符在 DTD 图中出现的次数则与它们在 DTD 中出现的次数相同；图的边则反映 DTD 中元素之间的嵌套关系；图中的环表示回路的出现。图6显示的是根据一个 DTD(见图2)所产生的 DTD 图，其中斜体字表示的是根据属性生成的结点。

根据一个 DTD 图所生成的关系模式是该 DTD 图中每一个元素所生成的关系模式的并。为了说明如何生成一个元素的关系模式，需要引入元素图的概念。简单地说，一个元素的元素图就是从该元素出发以深度优先遍历 DTD 图的过程中所生成的一棵树，如果遍历过程中到达了一个已经遍历过的结点，则表示出现了回路，回路看成是逆向边，它以虚线边来表示，且该条路线不必再去重复遍历。

根据 DTD 图生成关系模式的方法有：基本内联法、共享内联法和综合内联法等。

基本内联法 对于每一个名为 e 的元素，根据它的元素图生成一个同名关系 e，元素图中根结点 e 的所有后裔叶结点均内联到该关系中来作为一个属性。但下列两种情况除外：①一个 * 或＋结点的直接孩子结点不包括在该关系中，即对于集合值孩子将另外生成一个新的关系；②产生逆向边的结点不包括在该关系中，即另外生成一个新的关系来处理回路。在生成的关系模式中，关系的属性是以从根结点 e 到内联结点的路径来命名的；* 或＋结点的直接孩子结点所另外生成的关系也必须以从根结点 e 到该结点的路径来命名；每一个关系需要有一个 ID 域来作为该关系的键；对于 DTD 图中有父结点的元素结点所生成的关系，还需要有一个 parentID 域来作为关系的外键。基本内联法将产生大量的关系，并且会产生大量的数据冗余，因此该方法基本上是不实用的。

共享内联法 使每一个元素结点出现且只出现在一个关系中。共享内联法按如下原则来判断哪些元素可以生成独立的关系：①DTD 图中入度大于1或等于0的元素结点生成独立的关系(模式)；②DTD 图中结点 * 或＋的直接后继元素结点生成独立的关系(模式)；③互为递归的入度均为1的元素结点，其中之一生成独立的关系(模式)；④其余的结点均生成关系属性。图7显示的是根据一个 DTD 图(见图6)按共享内联法所产生的关系模式。需要说明的是，由于 publication 元素结点下只有 * 结点，而 * 结点的后继结点分别生成独立的关系，因此这里对 publication 元素结点并没有生成相应的关系。共享内联法与基本内联法相比减少了数据冗余，但是在查询时却增加了许多连接操作。

book(bookID，book. year，book. price)
article(articleID，article. editorID)
editor(editorID，editor. id)
title(titleID，parentID，title)
author(authorID，parentID，author. id)
name(nameID，parentID，name)
email(emailID，parentID，email)

图7 共享内联法产生的关系模式

book(bookID，book. year，book. titl，book. price)
article(articleID，article. editorID，article. title)
editor(editorID，editor. id，editor. name，editor. email)
author(authorID，parentID，author. id，author. name，author. email)

图8 综合内联法产生的关系模式

综合内联法 在共享内联法的基础上，将所有入度大于1的元素结点也内联进入父结点所生成的关系中去，但是带回路的结点以及结点*或+的后继结点除外。综合内联法的出发点是充分吸取基本内联法和共享内联法的优点，克服其缺点。图8显示的是根据一个DTD图(见图6)按综合内联法所产生的关系模式。

2.3 基于文档树遍历序号的关系存储策略

对XML文档的查询包括内容查询和结构查询两个方面。前者是对XML文档中的元素内容进行限定的选择查询；而结构查询就是对构成XML文档的元素之间的内在关系进行查询，这种元素之间的内在关系包括祖先-后裔关系、双亲-孩子关系。

文[9]利用数字模式[17]来对XML文档中的所有元素和属性结点进行编号，这种编号称为数字模式(Numbering Schema)，以实现快速地确定在XML数据层次结构中任意结点对之间的祖先-后裔关系，从而有效地处理XML的RPE查询，并给出了实现祖先-后裔关系的连接算法。但他们并没有考虑支持关键字搜索，同时不是利用关系数据库来实现的，没有对支持XQuery查询进行完整的考虑。

文[10]利用逆序列表[18]来对XML文档中的所有元素结点和关键字进行编号，这种编号称为逆序列表(Inverted List)，以支持关键字搜索。他们主要是对XML的RPE查询和关键字搜索中遇到的一类包含连接操作在关系数据库系统中的实现性能进行了分析，并给出了实现祖先-后裔关系的连接算法，同时并没有对XML数据的关系存储和支持XQuery查询进行完整的考虑。

为了有效地实现对文档树中任意两个结点对之间的祖先-后裔关系或双亲-孩子关系的检测，同时实现按关键字搜索XML文档，将数字模式[9,17]和逆序列表[10,18]的思想相结合，我们对XML文档中的所有组件(包括元素结点、属性结点以及属性结点的值和文本结点的内容中的"词")进行先序遍历(即深度优先遍历)，产生这些组件的先序遍历序号，这种编号称为扩展先序列表(Extended Preorder List)。其基本思想是[11]：

(1)对于所有具有相同元素标记名Tag的元素结点建立一个扩展先序列表Elem_Tag，对于所有具有相同属性名Name的属性结点建立一个扩展先序列表Attr_Name，该列表中的每一个记录是标识该结点的一个四元组(docID,order,size,depth)。其中，docID是该结点所在文档的文档标识；order是该结点的先序遍历序号；size是为该结点中所包含的所有后裔组件的个数(包括预留的组件个数)，因此，必须满足$x.size \geqslant \Sigma_y.size$，这里y是x结点的所有直接孩子结点。order及size联合起来可以用来检测结点之间的祖先-后裔关系；depth是该结点在文档树中所处的层数，以反映祖先-后裔关系中的双亲-孩子关系。同时规定，对于一个元素结点而言，对它的所有属性结点的遍历必须先于对其子元素结点以及文本结点中的文本词的遍历。

(2)对于所有属性值和叶子元素结点内容中出现的所有的"词"建立一个扩展先序列表TextWord，该列表中的每一个记录是标识该"词"的一个四元组(word,docID,order,depth)。其中，word用于存储这个"词"；docID是该"词"所在文档的文档标识；order是该"词"的先序遍历序号；depth是该"词"在文档树中所处的层数，以反映它是直接属于那一个属性结点或叶子元素结点中。扩展先序列表TextWord的作用是实现按关键字搜索。

(3)进一步还要建立一个列表Document，该列表中的每一个记录与一个XML文档相对应，用来存储docID(文档标识)、URL(文档所在位置的URL)以及其它一些与该文档相关的信息。

有了扩展先序列表，文档树中任意两个组件之间的祖先-后裔关系或双亲-孩子关系可以通过如下命题直接进行判断：

命题1 对于一个XML文档森林中的任意两个组件x和y，组件y是组件x的后裔(或孩子)，当且仅当x.position contains (or directly contains)y.position。

其中，x.position contains y.position意味着x.docID=y.docID and x.order<y.order and y.order<=x.order+x.size。"directly contains"意味着在满足"contains"的条件基础上，还必须满足x.depth=y.depth-1。

根据扩展先序列表，我们设计了XML文档的关系存储模式，如图9所示。

Document(**docID**,URL,…)
Elem_*Tag* (**docID**, **order**, size, depth, attrValue, textValue, parentOrder,contentType)
Attr_*Name* (**docID**, **order**, size, depth, attrValue, parentOrder, contentType)
TextWord(**word**,**docID**,**order**,depth)
Atructure(**docID**,**order**,size,depth,nameString,node Type)

图9 我们提出的XML文档的关系存储模式

(1)关系表的码属性用粗体表示。我们假设所有的关系表都按码属性建立聚集索引，以加快查找的执行速度。另外，关系表TextWord还要按{docID,order}建立索引。

(2)对于Elem_Tag表，attrValue列存储元素结点的ID类型的属性值；textValue列存储叶子元素结点的文本内容；contentType列存储元素结点的内容类型，元素结点的内容类型要么是"E"，表示该结点是非叶子元素结点，要么是一个简单类型，表示该叶子元素结点的内容类型。对于Attr_Name表，attrValue列存储属性结点的值；contentType列存储属性结点的值的类型。parentOrder列存储元素结点或属性结点的双亲元素结点的序号order。

(3)Structure表用来存储所有元素结点和属性结点的扩展先序列表，即文档的总索引。其中，nameString列存储的是元素结点的标记名或属性结点的属性名；nodeType列存储的值是"EE"、"ET"或"A"，它们分别表示结点的类型是非叶子元素结点、叶子元素结点或属性结点。建立该表的目的是有效地实现XML文档的结构查询以及查询结果的文档树片断的重构。

2.4 四种关系存储策略的比较

第一种策略一般用来解决将目前商用关系数据库系统中已经存在的大量关系数据转换为XML文档进行发布，并没有解决Web上已经存在或正在产生的大量XML文档的有效存储和查询的问题。

第二种策略不能有效地处理RPE查询和XML结构查询，在重构查询结果的XML文档片断时效率特别低。文[6]是对文[5]的扩展研究，使之支持关键字搜索，但并没有将XML的内容查询、结构查询和关键字搜索进行集成考虑，因此数据冗余较大。

第三种策略必须依赖于DTD或Schema来生成关系模式，但是，我们知道XML模式并不是必须的，且模式易变或不完全，因此该策略有很大的局限性。

第四种策略抓住了XML文档中各元素之间的(即结构上的)固有内在特征,它的主要优点是:能有效地决定XML层次结构中任意结点对之间的祖先-后裔或双亲-孩子关系,从而加速RPE查询的计算,而RPE是XQuery的核心。例如,在文[5]中类似"chapter//title"的查询,要么从chapter元素开始逐步追逐赶直到title元素,要么从title元素开始逐步反向追赶直到chapter元素,且每一步追赶都要进行一次两个边表的连接操作,而我们的方法仅需要进行一次两个元素表的连接操作。它的关系存储模式不依赖于XML模式,但能够充分利用它的模式信息来优化存储和查询。因此,它代表了基于关系数据库的XML数据管理的当前技术发展水平。我们提出的XML数据的关系存储模式,能够有效地实现对XML文档的内容查询、结构查询、关键字搜索和对XML文档的修改,全面支持XQuery查询,快速地判断XML文档树中任意结点对之间的祖先-后裔关系或双亲-孩子关系,从而有效地支持XML文档的结构查询。例如,能够特别有效地支持按元素标记名或属性名的查询,能够有效地支持RPE查询,能够有效地支持文档顺序的查询,能够有效地支持解引用操作,能够有效地支持双亲、后裔和孩子操作,能够有效地支持FLWR查询,能够有效地支持文档过滤、文档结构重组、聚集等操作,能够有效地实现将关系查询结果重构为XML文档片断,能够有效地支持关键字搜索,等等。

2.5 基于关系存储的XML查询

利用关系数据库来存储和查询XML文档,有其优势,也有其不足。它的主要优点是:利用一种成熟的技术,利用一种已经存在的高性能系统,并且能够无缝地实现对XML数据和关系数据的查询。它的主要缺点是:要么一个XML查询在转换为SQL查询时可能会产生大量的SQL查询,要么虽然产生的SQL查询数量不多,但它们带着很多连接操作。事实上,在利用特殊的数据库系统处理XML查询时也会遇到类似的问题。

由于模型上的差异,如何完整、有效地查询存储在关系数据库中的XML数据是目前正在研究的问题。基于关系存储的XML查询最终都要将XML查询(如XQuery)转化为SQL查询,并将SQL查询的表形式的结果再转换为XML文档返回给用户或应用。但是XML查询语言却比SQL要复杂得多,它们一般通过RPE来对XML文档中的嵌套结构进行查询,而且RPE中还可以包含各种操作符,因此,改写XML查询为SQL查询是一个挑战,这种转换的技术及其实现性能对基于关系数据库的XML数据管理的前途起决定性的作用。

不同的关系存储策略,查询转换的技术是不一样的。对于根据DTDs映射关系模式的存储策略,文[7,8]对将XML-QL查询[3]转换为SQL查询的实现问题进行了研究,通过引入映射图的概念,提出了利用映射图将带RPE的XML-QL查询重写为一组简单路径查询的算法,以及利用路径实例的统计信息来扩展Kleene表达式的算法,然后进一步描述了将简单路径表达式查询重写为SQL查询的方法。

对于基于XML文档树的遍历序号的存储策略,RPE的查询通常被转换为一种实现祖先-后裔关系或双亲-孩子关系的连接操作,称为"包含连接"。文[9,10]都分别讨论了实现这种包含连接的算法,同时文[10]还对包含连接在关系数据库系统中的实现性能进行了分析。

3 以XML文档发布关系数据

以XML文档发布关系数据需要两方面的技术,一方面是需要一种语言来描述从关系数据到XML文档的转换,另一方面是需要一种实现技术来有效地实现这种转换。语言描述是用来说明怎样结构和标记一个或多个关系表中的数据成为一个层次的XML文档。

3.1 基于SQL的语言描述及其实现技术

文[12]中提出了一种基于SQL的语言描述方法,它利用并扩展SQL的功能,嵌套的SQL表达式被利用来描述XML文档的嵌套,扩展的SQL标量和聚集函数被利用来描述XML元素构造。在不放弃现有SQL语义的前提下,这些扩展能够容易地增加到现有关系系统中去。基于SQL的语言描述方法的优点是标准的APIs(像ODBC等)能够被利用到查询并获得XML文档中去,它使得现有的工具和应用程序能容易地应用到集成关系数据和XML文档。

可以将关系数据到XML文档的转换技术划分为引擎内部和引擎外部两大类。这里指的引擎内部是指标记和结构工作都全部在关系引擎内部完成,引擎外部是指有部分(并不要求全部)工作是在关系引擎外部完成。实验结果[12]显示:引擎内部方法比引擎外部方法有显著的性能优势,这是因为绑定输出元组到主机变量需要较高的开销。

3.2 基于XML视图的语言描述及其实现技术

该方法的优点是:它能实现最一般的、动态的将关系数据转换到XML文档;它能较好地保护关系模式的私有性,文[13~16]分别对这种方法进行了研究。

文[13]中提出的语言描述方法是定义一种全新的RXL(Relational to XML Transformation Language)语言,用来定义一个关系数据库的XML视图,该XML视图是虚的,某个应用再利用XML查询语言XML-QL[3]在虚的视图上构造一个查询,抽取XML视图中的数据片断并对抽取的部分进行物化,实现将关系数据转换为XML文档。

文[14,15]的研究工作与文[13]类似,他们开发的XPERANTO是工作在(对象)关系数据库系统之上的一个中间件,它的主要不同之处是:①基于纯XML,即视图的定义和查询都是基于XML-QL;②处理对象-关系结构,而不仅仅是平坦的关系结构;③实现关系数据(如属性值)和元数据(如属性名)的无缝查询;④XML-QL定义的视图或查询在内部采用称为XQGM(XML Query Graph Model)的形式表示,这种表示被设计成一种中立的语言,这使得它很容易地修改为支持其他的查询语言。文[16]对视图的定义和查询都是采用XQuery语言。

结论 使用XML来描述数据仅仅是使得在Web信息上进行可表示的、高效率的查询成为可能的第一步,许多查询处理的问题还需要被研究。该领域的研究还仅仅是刚刚开始,研究成果也只是初步的,还有许多需要进一步研究和有待解决的问题。

利用关系数据库来存储和查询XML数据也是如此。XML查询语言(如XQuery)与SQL相比更具灵活性,特别是路径表达式的使用,因此,将XML查询语言表达的查询转换为SQL查询是一个挑战。在基于关系数据的XML视图查询环境下,文[13~16]分别对这种转换技术进行了研究;在根据DTDs映射关系模式的存储策略环境下,文[7,8]也对这种查

(下转第72页)

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