第33卷 第10期 计 算 机 工 程 2007年5月
Vol.33 No.10 Computer Engineering May 2007
·软件技术与数据库· 文章编号：1000—3428(2007)10—0052—03 文献标识码：A 中图分类号：TP311

基于XML的实时数据一致性描述与查询处理

张 晶，张云生
（昆明理工大学信息工程与自动化学院，昆明 650051）

摘 要：实时数据查询技术在工业企业信息平台中具有广泛的用途，XML 数据标准能够实现各子系统数据的统一描述。该文用成熟的关系数据库查询机制处理符合 DTD 的 XML 文档，提出了一整套数据模型、转换规则、算法描述，可以将 XML 文档转换为关系元组，从而达到用 XML 实现基于关系数据库的实时数据一致性描述和查询处理的目的。

关键词：实时数据一致性描述；XML；DTD 图；Element 图；关系数据库

XML-based Real-time Data Uniform Description and Query Processing

ZHANG Jing, ZHANG Yunsheng
(College of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650051)

【Abstract】 Real-time data query processing has been widely used in industry information platform. XML is fast emerging as the dominant uniform data standard for representing data from every sub-system. This paper explores the more conservative approach of using traditional relational database engines for processing XML documents conforming to document type descriptors (DTDs). The paper develops a set of data models, transform rules, and algorithm descriptions to implement a prototype system that converts XML documents to relational tuples. It achieves the possibility of real-time data uniform description based on XML and query processing mechanism.

【Key words】 Real-time data uniform description; XML; DTD graph; Element graph; Relational database

1 概述

根据国内外综合自动化技术的发展趋势和网络技术的发展现状，一般说来，工厂综合自动化系统的关键都在于企业实时数据信息的综合集成。由此可以看到，实时数据查询技术是企业综合自动化的重要组成部分，对各个层次、子系统的技术和应用，提供了强大的支持。但传统工厂综合自动化系统各个子系统之间的数据交换方式没有统一的标准，造成接口众多、访问效率低、安全性差等多方面的问题。XML 数据标准的出现，为能够实现各子系统数据的统一描述提供了成熟的技术途径。

采用XML文档作为企业综合自动化系统各层次、各子系统之间数据交换的标准格式后，存放在XML文档中的数据成为了未来实时信息查询的焦点。因而如何对一系列XML文档内容进行查询，并且能够提取、综合、分析它们的内容是目前实时数据查询研究的热点。由于XML文档本身具有树型或网型结构的半结构化数据特性，很多研究者采用具有半结构化特性的查询技术，取得了一些成果[5]。但是，运用半结构化方法的前提是完全放弃 20 多年来成熟的关系数据库技术，这是有很大缺陷的；能否运用关系数据库技术开创新的方法提供基于XML文档的实时查询机制就成为我们工作的焦点。

本文提出的技术方法证明了用标准关系数据库系统来实现快速高效的XML文档实时查询是切实可行的。利用这种方法，能使不相关的XML查询、数据集和模式适应关系数据库领域。其前提是文档类型描述[2]（Document Type Descriptor, DTD）或XML Schema的存在，因为DTD或XML Schema是对一系列具有相同模式XML文档的有效解释。其技术思路是，当查询XML 文档时，可以首先运行DTD产生一个关系模式；然后解析符合DTD的XML文档并把它们装入标准关系数据库管理系统关系表的元组中；再将基于XML文档的半结构化查询 (用半结构化查询语言XML- QL或Lorel[1]定义)转换为相应关系数据库上的SQL查询。我们用不同的XML DTD验证了这种方法，证明在大多数情况下是切实可行的，并针对出现碎片和查询连接过多的问题，改进了方法。

2 XML和DTD概述

2.1 XML

XML 是在互联网中实现数据交换的一种层次化数据格式。一个 XML 文档是嵌套的元素结构，以根元素作为文档的开始；元素数据可以属性或子元素的形式存在。图 1 为包含一部电影信息的 XML 文档。在这个例子中，有一个含有两个子元素 movietitle 和 director 的 movie 元素，director 元素有一个值为“ Zhang ”的标识符属性并进一步嵌套提供 name 和 address 信息。关于 XML 的详细信息可查阅文献[3]。

基金项目：云南省自然科学基金资助重点项目(2000F0004Z)；云南省教育厅青年教师科研基金资助项目(5Y0676D)；昆明理工大学校青年基金资助项目(20033005)；昆明理工大学信息工程与自动化学院教改基金资助项目

作者简介：张 晶(1974－)，男，博士生、讲师，主研方向：Web技术，软件工程和实时控制软件；张云生，教授、博导

收稿日期：2006-06-29 **E-mail：**zhangji0548_cn@sina.com

```
<movie>
    <movietitle> Hero  </movietitle>
    <director  id= ” Zhang ” >
        <name>
            <firstname>Yimou</firstname>
            <lastname>Zhang</lastname>
        </name>
        <address>
            <city>Xi ’ an</city>
            <zip>  710000</zip>
        </address>
    </director>
</movie>
```

图 1 XML 文档示例

2.2 DTD

DTD[2]描述XML文档的结构，类似于XML文档的模式说明。DTD通过详细描述其子元素和属性来说明一个XML元素的结构；其子元素之间的关系可以用下列操作符描述 ：*（表示有 0 或多个元素），+(表示至少有一个元素)，?(表示可选)，|(表示或)。XML所有的值均为string类型，只有类型定义为 ANY 的情况下值才能为任意的XML片段。每一个元素有一个特殊属性 id，id唯一标识文档中的一个元素，其它元素可通过无类型的IDREF域来引用。DTD中没有根的概念——只要是DTD中的元素均可以充当符合DTD的XML文档的根元素。图 2 为一个DTD 描述，图 1 是符合此DTD 的XML文档实例。

```
<!ELEMENT   movie(movietitle,director)>
<!ELEMENT   mtv(title,director*,contactdirector)>
<!ELEMENT   contactdirector  EMPTY>
<!ATTLIST    contactdirector directorID  IDREF   IMPLIED>
<!ELEMENT   documentary(title,director,producer)>
<!ELEMENT   producer(documentary*)>
<!ATTLIST    producer  name CDATA  #REQUIRED>
<!ELEMENT  director(name,address)>
<!ATTLIST   director  id   ID#REQUIRED>
<!ELEMENT  name(firstname?,lastname)>
<!ELEMENT  firstname(#PCDATA)>
<!ELEMENT  lastname(#PCDATA)>
<!ELEMENT  address  ANY>
```

图 2 XML 文档的 DTD 描述

3 关系数据库中 XML 文档的存储

本节主要描述如何从 XML 的 DTD 中产生关系模式，主要解决两个方面的问题：(1) DTD 元素的复杂性问题；(2) 关系模式的两层结构 (关系表和属性)与 XML DTD 嵌套模式的冲突问题。

3.1 DTD 的简化

一般而言，将复杂的 DTD 转换成相应的关系模式很困难。这里采取的技术路线建立在简化 DTD 的基础上，产生一个能存储和查询符合此 DTD 规范文档的关系模式，但不必从产生的关系模式中重构 DTD。而且，能够满足：(1)任何符合 DTD 的 XML 文档都能在关系模式中存储；(2)任何符合 DTD 规范的 XML 半结构化查询都能用相应的关系数据库实例来执行。这里提出一套不破坏符合 DTD 文档查询效力的能用于“ 简化 ”DTD 的转换规则。

此类转换有 3 种形式：(1)扁平转换，将嵌套定义转换为扁平形式（也就是在任何操作中不再出现二元操作符“ , ”和“ | ”，见图 3）；(2)简化转换，将多个一元操作符简化为单一的一元操作（见图 4）；(3)组合转换，即组合具有相同名字的子元素（例如，两个 a*子元素聚合为一个 a*子元素，见图 5），另外，所有“ + ”操作符转换为“ * ”操作符。

		$\cdots,a^*,\cdots,a^*,\cdots\rightarrow a^*,\cdots$
	$e1^{**}\rightarrow e1^*$	$\cdots,a^*,\cdots,a?,\cdots\rightarrow a^*,\cdots$
$(e1,e2)^*\rightarrow e1^*,e2^*$	$e1^*?\rightarrow e1^*$	$\cdots,a?,\cdots,a^*,\cdots\rightarrow a^*,\cdots$
$(e1,e2)?\rightarrow e1?,e2?$	$e1?^*\rightarrow e1^*$	$\cdots,a?,\cdots,a?,\cdots\rightarrow a^*,\cdots$
$(e1\|e2)\rightarrow e1?,e2?$	$e1??\rightarrow e1?$	$\cdots,a,\cdots,a,\cdots\rightarrow a^*,\cdots$

图 3 扁平转换规则　　图 4 简化转换规则　　图 5 组合转换规则

这类转换保持“ 一对一或一对多 ”和“ 空或非空 ”的语义，但丢失了元素相对位置的信息。可以采取另外的技术策略来解决，当一个特定的 XML 文档被装入关系模式时这些信息能够被获取（举例来说，在元组中设置表示元素位置的字段）。

3.2 模式转化技术的目的

传统上，关系模式源于实体-关系(Entity-Relationship)模型。这种转化简单直接，因为实体和它们的属性能够清楚地分离开来；每个实体和它们相应的属性都能被映射为一个关系。 将一个 XML DTD 转化为关系就是试图将 DTD 中的每个元素映射为一个关系并将元素的属性映射为这个关系的属性。然而，DTD 的元素和属性与实体-关系模型的实体和属性之间并不存在对应关系。一个在实体-关系模型中被认为是“ 属性 ”的在 DTD 中通常很自然地被描述为元素。图 2 表明了这一点，在一个实体-关系模型中，director 是一个“ 实体 ”， 明显 firstname、lastname 和 address 是此实体的属性。而当设计一个 DTD 时，就不存在将 director 定义为元素，而将 firstname、lastname、address 定义为 director 属性的必然联系性。实际上，如果 XML 中的 firstname 和 lastname 是属性，由于 XML 规定属性不能嵌套，因此它们就不能嵌套在 name 元素中；但直接将 XML 元素映射成关系又会导致产生过多的文档碎片。如何解决这些问题是研究关注的焦点。

3.3 基本嵌入技术

基本嵌入技术通过将一个元素的子孙尽可能多地嵌入到一个单独的关系中来解决碎片问题。但是，因为 DTD 中的任意元素都可作为 XML 文档的根结点，基本嵌入技术能为每一个 XML 元素建立一个关系。例如，图 2 中的 director 元素被映射为包括 firstname、lastname 和 address 属性的关系；另外，firstname、lastname 和 address 也将被建立为独立的关系。

有两个复杂问题必须解决：集合属性和递归。图 2 的 DTD 例子中，当为 mtv 建立关系时，因为传统的关系模型不支持集合属性，不能嵌入一组 director。为了实现集合赋值，我们使用能够把集合存储在关系数据库中的标准化技术：为 director 创建一个关系，使用外部键将 director 关系与 mtv 关系关联起来。使用嵌入技术（若想处理能终止）必须要限制递归中嵌套的层数。因此用关系键的概念表达递归关系并用关系递归处理机制来查询关系。为了能用更一般的方式来处理这种转换，我们引入 DTD 图的概念。

DTD 图表达了 DTD 的结构。节点为 DTD 中的元素、属性和操作符。每个元素在图中仅出现一次，而属性和操作符出现的次数与 DTD 中相同，图 6 为图 2 DTD 对应的 DTD 图，DTD 图中的环表示递归。

为 DTD 创建的关系模式是为其每个元素创建关系的集合。为了确定某些关系的集合是否是为某个特定元素创建的，又提出了 element 图的结构，它是按照以下的方法构建的：

从要创建关系的元素节点开始深度优先遍历 DTD 图，每个首次到达的节点都标记为“ 已访问 ”，一旦确定它的所有孩

子节点都被遍历完毕后再将“已访问”标记取消。深度优先遍历时如果到达的是 DTD 图中未标记的节点，则为它在 element 图中创建一个同名的新节点。另外，创建一条从最近创建的节点指向此新节点的有向边，最近创建的节点与新节点在 DTD 图中对应节点的父节点同名。如果深度优先遍历到的是一个已标记过的 DTD 节点，则在 element 图中增加一条从最近创建的节点指向自身的有向回边。

图 6 所示 DTD 图中元素 producer 的 element 图如图 7 所示。为直观起见，element 图将 DTD 图中的相关部分展开为树结构。

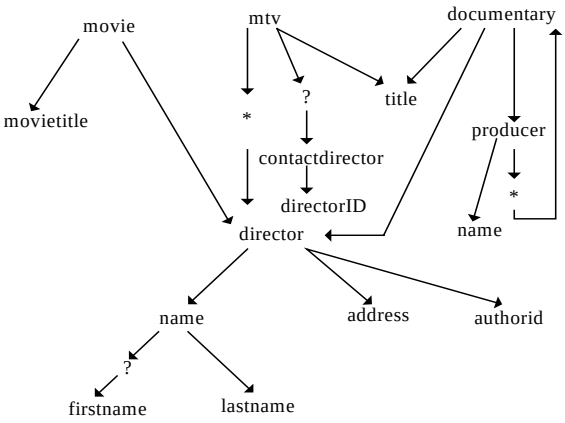

图 6 DTD 图

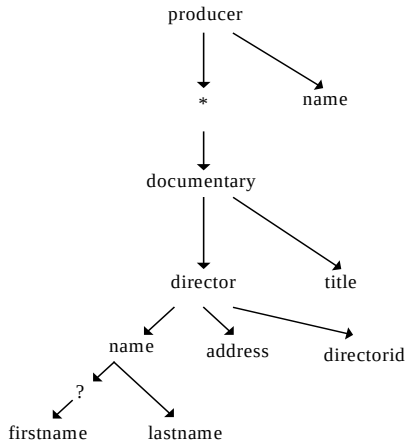

图 7 元素 Producer 的 Element 图

给定一个 element 图，相应的关系被创建如下。首先为图中的根结点元素创建一个关系，该元素的所有子孙都要被嵌入到此关系中，只有以下两种情况除外：(1)紧接“*”节点的孩子被转换成单独的关系，这是为集合属性子结点创建相应的新关系；(2)有回向边的节点被转换成单独的关系，这是为处理递归而创建的相应新关系。以下是图 2 所示的 DTD 所产生的关系模式：

movie(movieID:integer,movie.movietitle:string,movie.director.name.firstname:string,movie.director.name.lastname:string,movie.director.address:string,director.directorid:string)

movietitle(movietitleID:integer,movietitle:string)

mtv(mtvID:integer,mtv.contactdirector.directorid:string,mtv.title:string)

mtv.director(mtv.directorID:integer,mtv.director.parentID:integer,mtv.director.name.firstname:string,mtv.director.name.lastname:string,mtv.director.address:string,mtv.director.directorid:string)

contactdirector(contactdirectorID:integer,contactdirector.directorid:string)

title(titleID:integer,title:string)

documentary(documentaryID:integer,documentary.parentID:integer,documentary.title:string,documentary.producer.name:string,documentary.director.name.firstname:string,documentary.director.name.lastname:string,documentary.director.address:string,documentary.director.directorid:string)

producer(producerID:integer,producer.parentID:integer,producer.name:string)

producer.documentary(producer.documentaryID:integer,producer.documentary.parentID:integer,producer.documentary.title:string,producer.producer.documentary.director.name.firstname:string,producer.documentary.director.name.lastname:string,producer.documentary.director.address:string,producer.documentary.director.directorid:string)

director(directorID:integer,director.name.firstname:string,director.name.lastname:string,director.address:string,director.directorid:string)

name(nameID:integer,name.firstname:string,name.lastname:string)

firstname(firstnameID:integer,firstname:string)

lastname(lastnameID:integer,lastname:string)

address(addressID:integer,address:string)

基本嵌入技术对某些类型的查询是有效的，如：“list all directors of movies”，但对于其它的查询效率却非常低。例如，“list all directors having first name Zhang”这条查询语句的执行将被分解为 5 次独立查询执行的集合。基本嵌入技术的另一个缺点是它要创建大量的关系。

4 结论

本文的重点是研究使用传统关系模型来处理符合一定模式的 XML 文档实时查询的方法。通过实验证实，除了某些类型的复杂递归查询，用关系数据库可以处理大多数基于 XML 文档的查询。

经验表明，如果关系数据库系统在集合操作支持、无类型/可变类型引用、信息查询引擎、灵活的比较运算符等方面进行扩展，可以更有效地处理 XML 实时查询。这些扩展自身并不新颖，而且在其他研究中已提到。然而，在处理 XML 文档实时查询时，它们发挥了新的重要作用。更进一步研究这些处理 XML 文档的实时查询技术，将推动成熟关系数据库管理技术的发展，用于满足以 XML 为基础的企业综合自动化系统软件的实时查询需求。

参考文献

1 Abiteboul S, Quass D, McHugh J, et al. The Lorel Query Language for Semistructured Data[J]. International Journal on Digital Libraries, 1997, 1(1): 68-88.

2 Bosak J, Bray T, Connolly D, et al. W3C XML Specification DTD[Z]. 1999. http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm.

3 Bray T, Paoli J, Sperberg-McQueen C M. Extensible Markup Language (XML) 1.0[Z]. 2004. http://www.w3.org/TR/REC-xml.

4 Bray T, Frankston C, Malhotra A. Document Content Description for XML[Z]. 1998. http://www.w3.org/TR/NOTE-dcd.

5 Buneman P, Davidson S, Hillebrand G, et al. A Query Language and Optimization Techniques for Unstructured Data[C]//Proceedings of the ACM SIGMOD Conference, Montreal, Canada. 1996-06.

6 张云生. 实时控制系统软件设计原理及应用[M]. 北京: 国防工业出版社, 1998-01.