基于关系数据库的 XQuery 查询的实现

蔡 飞 贝 佳 陶列骏 潘金贵

(南京大学计算机软件新技术国家重点实验室,南京大学计算机科学与技术系 南京210093)

摘 要 XML 的应用日益广泛,XML 数据库技术也得到迅速的发展,但是从安全性、索引结构、事务处理等方面考虑,关系数据库仍然是更加可靠的选择。本文提出了在关系数据库的基础上提供虚拟的 XML 数据库、并实现 XQuery 查询的一种中间件的设计,结合了两者的优点。

关键词 XQuery,XML 数据库,关系数据库

The Implementation of XQuery on Relational Database

CAI Fei BEI Jia TAO Lie-Jun PAN Jin-Gui

(State Key Lab for Novel Software,Department of Computer Science and Technology,Nanjing University,Nanjing 210093)

Abstract XML becomes more and more popular in Web implementation as a well-structured language. XML database and related technologies are also rapidly developing. But if security, index structure, transaction processing are considered, relational database is still the more reliable choice. Based on relational database, we design a kind of middleware implementing the XQuery, so combine the best of both technologies.

Keywords XQuery, XML database, Relational database

1 引言

由于 XML 良好的层次性和结构性,人们发现 XML 极其适合用于数据存储,XML 数据库的概念也就此提出。所谓 XML 数据库,是可以对 XML 文档进行存取管理和数据查询的数据库。与传统的关系数据库相比,XML 数据库能够对半结构化数据进行有效的存取和管理;能够清晰表达数据的层次特征,更加适合管理结构比较复杂的数据;如果以 XML 格式存储数据,则 XML 数据库利于文档存储和检索;另外 XML 数据库能够存储和查询异种的文档结构,提供对异种信息存取的支持。

目前 XML 数据库主要有两种类型:XML 本源数据库(NXD, Native XML Database)[1]、支持 XML 的数据库(XEDB,XML-enable database),前者以 XML 作为数据存储的格式,后者则以传统的关系数据库为数据载体,增加对 XML 数据的管理功能,实现数据关系不太复杂的 XML 文档与传统数据库之间的转换。

如果建立的数据库是基于 Web 的,同时管理的信息具有半结构化特征,那么 XML 数据库是很好的选择,但是 XML 数据库技术不很成熟,在有效的存储组织、合理索引结构、数据库系统的安全性、事务处理、数据完整性、触发器、数据的聚合能力等方面还有待提高,而关系数据库经历了几十年的发展和完善,已经相当成熟,因此大多数商用数据库仍然选择使用关系数据库存储数据。基于关系数据库,XEDB 或者中间件技术可以为用户提供虚拟的 XML 数据库[8,9]。本文介绍了中间件 VegasXML (Versatile Executor of Generating And Saving XML)的设计和实现,该中间件基于关系数据库提供虚拟的 XML 数据库,实现 XML 查询语言 XQuery[4]的查询功能,为基于 Web 的数据操作提供了解决方案。

2 XML 查询语言 XQuery

随着 XML 数据库技术的发展,简单高效的查询语言成为研究的重点。很多查询语言,包括 Lorel[6],XQL,XML-QL,Quilt,XPath[3]都应运而生,XPath 可以用来查询多个 XML 文档集合,但它也存在着不足,其中最主要的缺点是缺少排序、分类和数据类型。Lorel 的功能则比较强。Quilt 综合了几种语言的优点,参考了 XPath,XML-QL,定义了 FLWR 表达式,而且支持用户自定义函数。文[5]中对几种 XML 查询语言进行了详细全面的比较。W3C 的 XML Query 工作组在1999年9月正式成立,任务是创建一种灵活的查询语言从 XML 文档中抽取数据。构建于 XPath 规范之上,以 Quilt 为基础,增加与 XML 模式、XML 查询代数的结合,工作组创建了新型查询语言 Xquery,为 XML 文档中的数据操作提供了一种强大的语法,解决了 XPath 中的许多缺点。到目前为止,XQuery 规范仍然处于"工作草案"状态,但是已经得到多种数据库的支持,当 XQuery 标准在 W3C 中成为最终规范时,我们将看到更多的数据库提供商采用该标准。

XQuery 构建于 XPath 规范之上,与 XPath 相比,XQuery 最大的新特性是 FLWR[4]表达式。FLWR 是 For-Let-Where-Return 的首字母缩略词,每个 FLWR 表达式都有一个或多个 for 子句、一个或多个 let 子句、一个可选的 where 子句以及一个 return 子句。

for 子句:用来指定一组笛卡尔元组,表达式的其余部分将对该元组求值。通过为这些笛卡尔组选定次序来控制求值的次序。let 子句:为一个变量赋一个值或一个序列。where 和 return 子句:指定一些条件,过滤不能满足条件的元组,作用与 SQL 中的 WHERE 子句很相似。return 子句定义每个元组要返回的内容。以下是 XQuery 的一个查询示例:

```
〈result〉
{
    for $u in document("users.xml")//users_tuple
    let $b := document("bids.xml")//bids_tuple[userid =
    $uid]
    return
        〈bidder attrvalue="{$b/@userid}" extensionattr="abc
            def"〉{$u/name}
            〈extend〉extendvalue〈/extend〉
```

```
        〈/bidder〉
    }
〈/result〉
```

由上例可以看出，XQuery 保留了 XML 的一些特性，比如标签(tag)、属性(attribute)、元素(element)等，这些特性为查询的结果提供结构框架，以生成 XML 格式的结果。数据源仍然使用 XPath 指定。XQuery 详细的语法定义见文[4]。

3 VegasXML 及其关键实现技术

3.1 VegasXML 体系结构

XML 数据与关系数据库的二维表相比，结构复杂得多，VegasXML 的设计目的是利用自身定义的映射规则集，保存 XML 中的结构信息，实现基于关系数据库的 XML 文件管理。以往的研究结果表明：以关系数据库作为 XML 数据的载体，需要将 XML 数据模型映射到关系模型，甚至要考虑到消除重复数据和数据类型的限制，VegasXML 定义的映射规则集和映射算法很好地解决了这些问题，充分利用了关系数据库成熟的管理技术。在体系结构的设计上，我们借鉴了 Lore 系统[2]。

VegasXML 由两部分组成：

(a)XMLGenerator：利用映射规则集和从关系数据库中取得的二维数据进行模型转化，生成 XML 文档。

(b)XMLSaver：利用映射规则集从半结构化数据，即 XML 文档中提取数据，转化到关系模型，存入关系数据库。

在这两个基本模块之上，可以进行更高层次的开发，使用 XML 查询语言，实现对数据的操作。如图1所示，基于 XMLGenerator，在 VegasXML 中，我们提供了对 XQuery 的支持模块 XQuerySolver，它根据指定的数据库连接和 XQuery，进行半结构化数据模型的提取，生成映射规则集，获取数据后，利用 XMLGenerator 结合映射规则集进行模型转化，即可生成最终的 XML 格式的查询结果。

由于 XQuery 规范目前并没有提供数据更新的功能，因此，基于 XMLSaver 进一步的开发，实现对数据库的更新，有待于 W3C 提出相应的规范，或者对现有的 XQuery 规范进行扩充，增加数据更新的功能，QuiLogic 公司推出的 SQL-IMDB 中就对 XQuery 自行进行了扩展，限于篇幅，本文中不多加描述。

基于以上设计，VegasXML 实际上为用户提供了一个虚拟的 XML 数据库，数据库中的每一张表就是一个虚拟的 XML 文档，以表1为例：

表1

BIDS

USERID	ITEMNO	BID	BID_DATE
U01	1003	34	2002-2-4
U02	1005	125	2002-9-5

它对应了如下虚拟的 XML 文件：

```
BIDS. xml
〈BIDS〉
  〈BIDS_tuple〉
    〈USERID〉U01〈/USERID〉
    〈ITEMNO〉1003〈/ITEMNO〉
    〈BID〉34〈/BID〉
    〈BID_DATE〉2002-2-4〈/BID_DATE〉
  〈/BIDS_tuple〉
  〈BIDS_tuple〉
    〈USERID〉U02〈/USERID〉
    〈ITEMNO〉1005〈/ITEMNO〉
    〈BID〉125〈/BID〉
    〈BID_DATE〉2002-9-5〈/BID_DATE〉
  〈/BIDS_tuple〉
〈/BIDS〉
```

3.2 关键实现技术

从图1可以看出，对 XQuery 的处理分为两步，上层的 XQuerySolver 对输入的 XQuery 进行解析，从数据库中抽取数据，同时提取出 XQuery 中的结构信息，以映射规则的形式存放；下层的 XMLGenerator 使用映射规则集，结合数据集，进行模型转化，生成 XML 格式的查询结果。

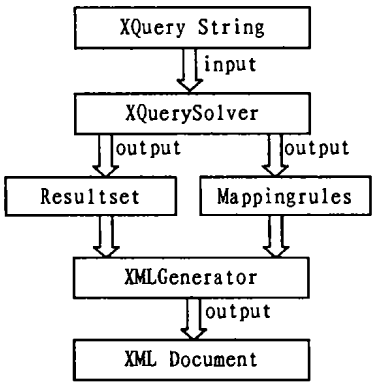

图1 XQuery 查询示意图

这个结构实现的关键在于：①映射规则的设计；②数据集的提取；③映射规则集的生成；④利用数据集和规则集生成 XML 文件。

3.2.1 *映射规则* 对于映射规则的定义，不仅要实现关系模型和半结构化数据之间的转换，二进制数据、空值、字符集等问题也应该得到解决。

通常有两种方法将二进制数据保存到 XML 文档中：未析实体和 Base64编码处理[7]。对于关系型数据库，这两种方法都可能存在问题。从数据库中保存和检索二进制数据的规则非常严格，因此中间件的设计应该考虑到这个问题。

XML 中空值概念的支持可以通过设置可选的元素类型或属性来实现。如果元素类型或属性值为 null，XML 只要在文档中不包含该元素或属性就可以了。但是对数据库而言，空的元素或包含0长度字符串的属性并不是空值 null，它们的值为长度为0的字符串。在 XML 文档和数据库结构之间相互映射过程中，必须特别注意那些可选的元素类型或属性是否对应于数据库中的空值项，否则很可能出现无效文档错误(当将数据从数据库读出时)。

除了一些控制字符，XML 文档能够包含任何的 Unicode 字符。但是许多数据库都限制或者不支持 Unicode，而且需要一些特殊的配置才能够处理非 ASCII 编码的字符数据，因此，规则集的设计还应该考虑到字符集的问题。

另外，将关系模型映射到 XML 数据模型时，需要考虑到消除重复数据。

针对模型映射的需要和以上要注意的问题，VegasXML 定义了四种映射规则：

(1)ParentRule：映射 XML 结构中含有子元素的元素，包括 id，tagname，parent，controller，controller-arg 等域。

(a)id：规则的标识，在规则集中唯一。

(b)tagname：标签名，映射 XML 文档的元素名。

(c)parent：映射父元素，用父元素对应的规则集的 id 标记。根元素此域为空。

(d)controller：为当前元素指定聚合规则，取值可以是

"ALL"或"KEY"。取值为"ALL",聚合所有子元素,所有子元素为兄弟;取值为"KEY",根据指定的子元素,将这些子元素相同的子女聚合为一个,这些子元素由 controller-arg 指定。

(e)controller-arg:controller 为"KEY"时,此域用来指定子元素,用子元素对应的规则集的 id 标记,用逗号隔开;controller 为"ALL"时,则忽略此域。

通过聚合,即可实现重复数据的消除。

(2)ColumnRule: 映射 XML 结构中具有 PCDATA 的元素。

(a)id, tagname,parent:作用同 ParentRule。

(b)colname:映射数据库表的列名,以此确定对应关系。

(c)colon:映射数据库表的列的位置,作用与 colname 相同,因此在规则中只能出现二者其一。

(d)null:设定如何处理空值。此域为"empty",生成空元素;"omit",忽略此元素,即不生成。

(e)handler:设定处理数据的方式,当数据库中对应的列是二进制数据时,设为"BINARYDATA"。

(f)handler-arg:handler 为"BINARYDATA"时,设置数据类型和编码集。数据类型可以是 TEXT 或 XML。设为 TEXT 时,需要对预定义实体进行处理,即对数据中的一些特定字符进行替换,比如"&",替换为"&"。

(3)AttributeRule: 映射 XML 结构中元素的属性。

(a) id,colname,colno,null,handler,handler-arg:作用同 ColumnRule。

(b)attrname:属性名。

(c)attrof:指定此属性所属的元素,用元素对应的规则的 id 标记。

以上三种 Rule 实现了 XML 文档中的元素和数据库表中的列之间的映射,但是 XQuery 查询本身可能包含了一部分数据,因此相应的规则也是不可缺少的:

(4)ExtensionRule: 映射 XQuery 中本身包含的数据,可以是元素或者属性。

(a)id:作用同 ParentRule。

(b)tagname,parent:映射到元素时使用这些域,作用同 ColumnRule。

(c)attrname,attrof:映射到属性时使用这些域,作用同 AttributeRule。

(d)extender:指定获得数据的方式。此域为"STATIC",数据从 extender-arg 中取得;"EMPTY",生成空元素,只能在映射到元素时使用。

(e)extender-arg:extender 为"STATIC"时,指定属性或元素值。

3.2.2 *数据集的提取* XQuery 本身并不能对关系数据库进行直接的查询,需要将其重写为相应的 SQL 语句。由于两种查询语言的差异,实现 XQuery 到 SQL 的重写相当复杂。WPI(Worcester Polytechnic Institute)的 Rainbow[10]利用自身定义的 XML 查询代数树实现了这种重写,具体的实现参考文[11]第四章,在这里我们不作重复的讨论。

3.2.3 *映射规则集的生成* XQuery 保留了 XML 中元素和属性的定义,用来控制查询结果的结构,因此,VegasXML 只需对 XQuery 中的元素和属性生成相应的映射规则。生成规则大致如下:

(1)对应 XQuery 中的元素,生成 ParentRule,tagname 用元素名设定,id 逐个加1,parent 为父元素的 id。为了使查询的结果写入一个 XML 文件,第一个 ParentRule 的 controller 为"ALL"。

(2)对应 XQuery 中的属性,生成 AttributeRule 或 ExtensionRule。

(a)属性值由 XQuery 中的变量指定,即从关系数据中取数据时,生成 AttributeRule,id 逐个加1,colno 根据 XQuery 中的 XPath 指向关系数据的某一列,colno 即列的位置,attrname 为属性名,attrof 为当前元素的 id。

(b)属性值在 XQuery 中已经明确,生成 ExtensionRule,id 逐个加1,attrname 为属性名,attrof 为当前元素的 id,extender 为"STATIC",extender-arg 为 XQuery 中指定的属性值。

(3)对应作为元素内容的查询结果,生成 ColumnRule 或 ExtensionRule。

(a)元素内容由 XQuery 中的变量指定,生成 ColumnRule,id 逐个加1,parent 为当前元素的 id,tagname 为变量指向的关系数据某列的列名,colno 即列的位置。根据 XQuery 查询的特性,如果某列数据为空,就忽略对应元素的生成,因此,null 默认为"omit"。

(b)元素内容在 XQuery 中已经明确,生成 ExtensionRule,id 逐个加1,tagname 为空字符串,parent 为当前元素的 id,extender 为"STATIC",extender-arg 为 XQuery 中指定的元素内容。

按照以上规则,对应第二节中的 XQuery 查询示例,将生成如下的映射规则集:

```
〈MappingRule〉
    〈parentRule id = "p1" tagname = "result" controller = "
    ALL"/〉
    〈parentRule id= "p2" tagname="bidder" parent="p1"/〉
    〈attributeRule id="a1" colno="1" attrname="attrvalue"
    attrof="p2"/〉
    〈extensionRule id="e1" attrname="extensionattr" attrof="
    p2" extender="STATIC" extender-arg="abc def"/〉
    〈columnRule id="c2" colno="3" tagname="name" parent
    ="p2" null="omit"/〉
    〈parentRule id= "p3" tagname="extend" parent="p2"/〉
    〈extensionRule id="e2" tagname="" parent="p3" extender
    ="STATIC" extender-arg="extendvalue"/〉
〈/MappingRule〉
```

3.2.4 *利用关系数据和规则集生成 XML 文件* 映射规则集是根据半结构化的 XQuery 生成的,现在需要结合数据集,生成半结构化的 XML 文档,并非意味着两者是相反的过程。由上例可以看出,映射规则集为标准的 XML 文档,为了便于结果的生成,XMLGenerator 采用树型结构作为其中间表示,并根据不同的映射规则,定义了以下节点:

(a)ParentRuleNode:对应 ParentRule,其子节点可以包含 ParentRuleNode 和下面定义的各种节点。

(b)ColumnRuleNode:对应 ColumnRule,其父节点为其 parent 域所对应的 ParentRuleNode。

(c)AttributeRuleNode:对应 AttributeRule,其父节点为其 attrof 域所对应的 ParentRuleNode。

(d)ExtensionColumnRuleNode:对应含 tagname 域的 ExtensionRule,其父节点为其 parent 域所对应的 ParentRuleNode。

(e)ExtensionAttributeRuleNode:对应含 attrname 域的 ExtensionRule,其父节点为其 attrof 域所对应的 ParentRuleNode。

(以上节点的定义均保留了相应规则的各个域)

(下转第76页)

成熟级别	关键过程区域	瀑布式	螺旋式	XP	TSP	RUP	净室过程
2级可重复级	需求管理	✓	✓	✓	✓	✓	✓
	项目策划	✓	✓	✓	✓	✓	✓
	项目的追踪	✓	✓	✓	✓	✓	✓
	子合同管理	×	✓	×	×	✓	✓
	软件质量保证	✓	✓	✓	✓	✓	✓
	软件配置管理	✓	✓	✓	✓	✓	✓
3级已定义级	机构过程焦点	✓	✓	✓	✓	✓	✓
	机构过程定义	✓	✓	✓	✓	✓	✓
	培训管理	×	×	×	×	✓	✓
	集成软件管理	✓	✓	×	✓	✓	✓
	软件产品工程	✓	✓	✓	✓	✓	✓
	组间协调	✓	✓	✓	✓	✓	✓
	同行评审	✓	✓	✓	✓	✓	✓
4级已管理级	过程的量化管理	×	×	×	✓	×	✓
	软件质量管理	×	×	×	✓	×	✓
5级优化级	过程变更管理	×	×	×	✓	×	✓
	技术变更管理	×	×	×	✓	×	✓
	缺陷预防	×	×	✓	✓	×	✓

✓：表示覆盖　　×：表示未覆盖

图2　典型过程与CMM的关系

3.7　适用范围

在瀑布开发方法中，只有到生命周期的后期，才能确知周围是否存在风险。瀑布开发不适合需求模糊的系统，显然也不适宜用来开发大型项目，因此瀑布开发主要应用在对于需求和需求的实现有很好的理解的场合。只有已经存在类似系统的情况下，采用瀑布开发才有一些优势。

螺旋开发方法虽然已经对瀑布开发方法进行了改进，但是由于没有明确的角色分工，仍然难以处理大的复杂的项目。

XP灵活多变，适合需求不清的项目。但是由于受开发活动的限制例如代码集体拥有、同一个房间里工作以及开发人员数量不能过多，因此XP不适用于大型的项目。

TSP的小组可以有2至20人，一般在10至12人，这样的小组效率最高，最能够发挥个人与小组的能力，适应能力也比较强。TSP适合开发小型、中型的项目。也可以组成一个多小组的TSP过程，例如包含150人，这样的组可以处理大规模的项目。

RUP定义了大量的角色、工件和活动，适用于大规模和超大规模项目的开发，由于RUP有上述诸多优点并且提供了需求管理和管理变更的工具，故RUP能够适应开发中的各种变化，并且使开发的风险总在控制之下。另一方面，RUP又是可裁剪的，可按项目需要减少角色、工件和活动，剪裁之后的RUP是一个类似XP的过程，虽没有XP那样简单，但是风险性较小。

净室过程通过数学分析使开发中二义性、不完整性和不一致性更容易被发现和纠正。但是开发费用高，需要背景知识和培训，与用户通信也比较困难。净室过程适用于开发要求严格的关键软件，以及如果发生错误会造成严重经济损失的场合。

小结　本文提出了一个简明的软件过程比较框架，框架主要包括六个方面的内容：软件过程中的活动、活动的执行方式、过程中的角色、过程产生的工件、工具支持、与CMM的关系和适用的范围。在此框架基础之上，我们对几个典型的软件过程进行了分析和比较。这些比较将有助于我们选择、改进已有过程，同时，也为新型过程的设计提供了一个基础框架。

参考文献

1　朱三元，钱乐秋，宿为民. 软件工程技术概论. 科学出版社，2002
2　冯玉琳，黄涛，金蓓弘. 网络分布计算和软件工程. 科学出版社，2003
3　周之英. 现代软件工程. 科学出版社，1999
4　郑人杰，殷人昆，陶永雷. 实用软件工程. 清华大学出版社，1997
5　Beck K. Extreme Programming Explained. ADDISON-WESLEY，2000
6　Humphrey W S. 小组软件过程. 人民邮电出版社，2000
7　Pressman R S. 软件工程实践者的研究方法(第4版). 机械工业出版社，1999
8　Humphrey W S. The Team Software Process^{SM}(TSP^{SM}). (CMU/SEI-2000-023)
9　McAndrews D R. The Team Software Process^{SM} (TSP^{SM}): An Overview and Preliminary Results of Using Disciplined Practices. (CMU/SEI-2000-TR-015)

(上接第71页)

对中间表示树进行中根遍历，根据3.2.1节中的映射定义从节点上或数据集逐行取得数据输出到XML文档，即可实现从关系数据到XML文档的转换，生成的XML文档结构与中间表示树的结构一致，生成的XML文档即XQuery查询的最终结果。

4　查询的优化

对于XML数据树的查询，可以在搜索空间、代价估算、计划枚举等三个方面进行优化，但是在VegasXML中，数据的查询最终是由SQL实现，查询的对象并不是XML文档，而是关系数据库，因此查询的效率仍然取决于DBMS，主要的优化工作也是由DBMS完成的。

结语　VegasXML为用户提供了虚拟的XML数据库，并且实现了XQuery的查询功能，利用了关系数据库成熟的管理技术，为面向Web的数据库应用提供了新的选择。但是由于关系模型本身的缺陷，一些半结构化数据无法映射，因此该系统适合于数据结构不太复杂的应用，仍然有很多问题需要研究。今后将在这些方面对系统作出改进，并期待W3C在XQuery中增加对数据库更新的规范，从而实现一个完整的XML数据库管理系统。

参考文献

1　Staken K. Introduction to Native XML Databases. http://www.xml.com/
2　Goldman R, McHugh J, Widom J. Lore: A Database Management System for XML. Dr. Dobb's Journal April 2000
3　XML Path Language (XPath) Version 1.0 W3C Recommendation. 16 Nov. 1999. http://www.w3.org/TR/xpath
4　XQuery 1.0: An XML Query Language W3C Working Draft. 15 Nov. 2002. http://www.w3.org/TR/xquery/
5　Ceri B. Comparative Analysis of Five XML Query Language-s. SIGMOD Record, 2000, 29(1): 68～79
6　Abiteboul S, Quass D, McHugh J, Widom J, Wiener J. The Lorel Query Language for Semistructured Data. International Journal on Digital Libraries, 1(1): 68～88
7　硕网资讯. 洞悉XML. 北京大学出版社，2001
8　王照岳，孙建伶，董金祥. XML数据库管理系统研究. 计算机科学，2002，29(1)：115～117
9　郑仕辉，周傲英，季文赟，等. 基于SQL的XML查询的有效实现. 计算机研究与发展，2001，38(4)：422～428
10　Zhang Xin, et al. I Shrunk the XQuery!—— An XML Algebra Optimization. Approach Workshop on Web Information and Data Management (WIDM'02), Nov. 2002
11　Zhang Xin. Rainbow: Relational Database Auto-Tuning for Efficient XML Query Processing. A Proposal for a Dissertation in Computer Science. May 2001