XML 数据流基于组着色的 XPath 查询模型*

刘景超,刘先锋

(湖南师范大学 数学与计算机科学学院,湖南 长沙 410081)

摘 要:提出了一种新的 XML 数据流 XPath 查询模型 GBRender,该模型通过组着色序列来直接处理元素,具有较高的处理效率与较强的适应性。

关键词:XML 数据流;组着色;XPath 查询

中图分类号:TP274 **文献标识码**:A

XPath query model based on group rendering of XML data stream

LIU Jing Chao, LIU Xian Feng

(College of Mathematics and Computer Science of Hunan Normal University, Changsha 410081, China)

Abstract: The paper proposed a new XML data stream XPath query model GBRender, the model through the group rendering sequence to deal directly with elements, has high processing efficiency and strong compatibility.

Key words: XML data stream; group rendering; XPath query

由于 XML 已经成为 Web 上数据交换的标准,用于各种应用和信息源之间的数据交换,而许多应用的特征是数据以快速、实时的数据流形式持续到达,适宜用数据流建模。因此,处理 XML 流数据的理论和技术目前成为流数据研究领域中的一个热点。XML 流数据处理系统通常运行在 Web 上,用户查询通常用 XPath[1]语言表示。由于一个用户可以提交若干查询,使查询的数量十分巨大。XML 流数据处理研究的一个关键问题是如何同时有效处理大量来自用户的查询并及时将结果返回给用户。

根据数据流环境的特点,对 XML 数据流和处理通常有以下要求:每一个 XML 元素节点最多只能访问 1 次;使用有限和最少的内存空间存储临时数据,处理算法具有尽可能小的空间复杂度;对每个节点的处理必须有很高的效率,以满足实时处理的需要。

目前针对 XML 流处理系统所采用的主要方法是基于自动机的方法。其他方法有:基于索引的方法[2]、基于 Bloom-Filter[3]的方法、Fist[4]方法等[5]。基于自动机的方法是将 1 个或 1 组 XPath 表达式表示为某种形式的自动机,通过状态转换来达到查询所需信息的目的。基于索引的方法是在数据传送之前加入索引信息或接收端引入某种索引机制来提高查询效率。以上方法处理 XPath 的方式都是通过对 XPath 本身形成处理器,即处理的过程主要集中在 XPath 本身,而且基于索引或通过 DTD 的优化或多或少都依赖于对 XML 结构信息的了解。而由 XML 数据流的应用环境决定了很多应用是在结构未知的情形下的数据序列查询。基于现有的研究及上述问题,本文提出了一种新的 XML 数据流 XPath 查询模型,以适合在结构未知的情况下的 XPath 查询,同时有效地减少流查询过程中对 XPath 的依赖程度。

1 背景及相关问题

由于流数据处理的广泛应用以及 XML 已经成为 Web 上数据交换的标准,流数据处理的研究已引起广泛的兴趣。

很多研究都采用自动机方法处理 XML 数据流。自动机技术用于 XML 文件查询的主要思想是:将 1 个或 1 组 XPath 表达式表示为某种形式的自动机,在要查询的 XML 文件上运行自动机,自动机根据当前状态及读入文档的节点判断下一步的行动,运行结束根据自动机是否处于接收状态来判断文件是否符合给定的 XPath 的

* 基金项目:国家自然科学基金(10571052);湖南省高校青年骨干教师资金湖南省教育厅科研资金资助

查询条件。自动机查询模型如图 1 所示。

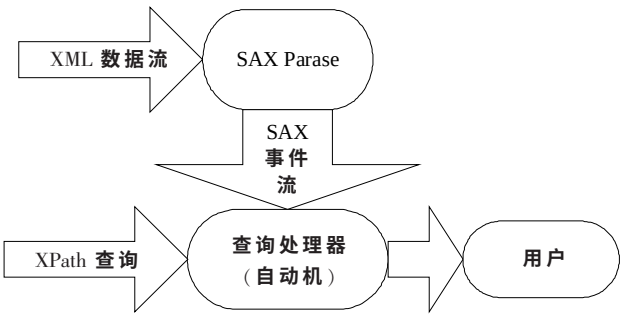


图 1 自动机查询模型

XFilter 首次利用基于有限状态自动机 FSM(Finite State Machine)的方法来过滤 XML 文档，它对每一个路径查询使用一个单独的 FSM，并在文档处理的过程中，同时运行所有的 FSM。YFilter 在 XFilter 的基础上进行了改进，它将所有的 XPath 查询合并成一个单独的非确定有限自动机 NFA (Non-deterministic Finite Automaton)[6]，并共享所有查询的公共前缀，YFilter 主要考虑了谓词中的 AND 谓词查询，采用后置处理的方式，这种方式产生大量中间结果影响系统性能。

XEBT (XPath Evaluation Based on Tree automata)[7]利用树自动机技术作为查询处理器，它基于表达能力丰富的树自动机，无须附加中间状态或保存中间结果，就能处理支持{[]}操作符的 XPath，所以能较高效地处理 XPath 查询。在优化策略上，主要包括基于 DTD 的 XPath 查询自动机的构造、在空间代价有限增加的情况下采用局部确定化减少并发执行的状态、采用自上而下和自下而上相结合的查询处理策略等。

其他处理 XML 流的方法主要有基于索引 (Index-Filter)的方法[2]、基于 Bloom Filter[3]的方法以及 FiST[5]方法。Index-Filter[2]采用基于索引的技术处理 XML 流数据，利用 XML 文档流的文档标记动态地建立 XML 文档的索引，从而避免处理一部分文档。在 Index-Filter 的方法中，建立索引要花费一定的时间，而且不能单遍处理 XML 文档。基于 Bloom Filter 的 XML 包过滤器是一种近似查询方法，利用 Bloom Filter 方法，将 XPath 表达式作为字符串，将 XPath 与 XML 包之间的匹配转换为字符串之间的匹配，从而提高查询性能，但它只是用来处理简单的 XPath 表达式且有一定的失误率。FiST 方法针对 Twig Pattern 提出的一种有别于 YFilter 的方法，将一组 Twig Pattern 转换为 prufer 序列，并对一组 Twig Pattern 与 XML 流数据进行整体性匹配。

2 GBRender 模型

2.1 GBRender 相关定义

当 XML 以流的形式进行处理时，在逻辑上实际是以先序遍历的形式对 XML 树中的结点进行访问，通常采用基于事件的 SAX 模型来进行解析。这样，在对 XML 结构未知的情况下，可以根据接收的元素信息来分析其结构，并根据得到的结构信息为后续服务。

为了用来更好地描述 GBRender 查询模型，下面介绍几个定义：

定义 1 (组)：XML 的任何一个元素都有一个从根开始的标签路径，但很多元素都会有相同标签路径，即 1 个标签路径可以表示 1 组对应的元素。这里把 XML 文档中每一个不同的标签路径称为组。如图 2 所示，root/person/name 即为一个组。

定义 2(组树)：组树是在处理 XML 数据流过程中，记录组、组之间关系的树形结构。其组织结构如图 3 所示。组树中，各组有指向其子组的链接，每个组有指向父组的链接，同时，同一层的组串接起来以便于处理时的需要。

定义 3(终结元素)：XPath 表达式对应的最后一个路径元素，称为终结元素，它表示 XPath 请求所需的结果。如//A/B//C，C 即终结元素。

定义 4(着色)：寻找每一个组的标签路径处于 XPath 表达式中的哪个位置，即当前 XML 元素的路径与 XPath 路径的关系并标记该组，这个过程称为着色过程。例如，组 root/person/name 对应 XPath 表达式//name 的终结元素 name，则可以标记该组为该 XPath 的终结组。

```
<root>
  <person>
    <name>Jim</name>
    <age>21</age>
  </person>
  <person>
    <name>Tom</name>
    <age>18</age>
  </person>
</root>
```

图 2 XML 文档

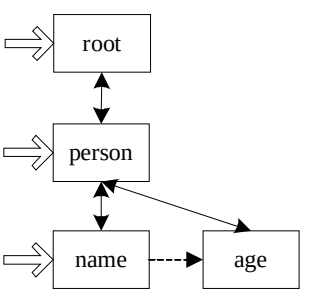


图 3 组树结构

2.2 GBRender 结构模型

目前的 XML 数据流 XPath 查询处理，基本结构模型如图 4 所示，以 XML 数据流作为 XPath 分析器的输入，分析器可能有多个，也可能合并为 1 个。对 XPath 查询包含的索引信息有 2 种：一是传输之前的索引信息；二是接收端进行处理形成的索引信息。

GBRender 查询模型如图 5 所示，采用基于着色的方式来处理查询，当某个组首次出现时，通过着色器对其与 XPath 的关系进行着色。着色之后，再次遇到该组时不再依赖于 XPath 而只依赖其记录的组着色信息，尤其在单枝查询的情况下，可以直接对查询进行回应。在 GBRender 模型中，组着色的过程，实际上相当于 XPath 确定化的过程。例如，对于图 2 的文档进行//name 查询，

当首次出现 name 时，其组标签路径为 root/person/name，即是对//祖孙关系在此文档上进行的确认化，而且当 name 出现在另一组时，会再一次进行确认化且不会相互影响。而常采用的 DTD 优化策略通常也是完成确认化的工作，当遇到 2 个不同组均存在 name 标签时，//就需要特殊处理了。

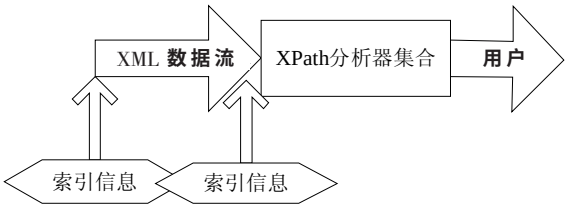


图 4 目前查询模型

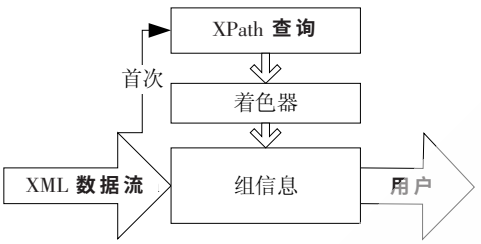


图 5 GBRender 查询模型

为了同时响应多个 XPath 查询，本文引入了着色序列的定义。

定义 5(着色序列)：根据 XPath 请求序列，对同一组生成对应的着色信息序列，此序列即称为着色序列。因为不同的 XPath 有不同的着色信息，因此也称 XPath 着色序列。

每组均有着色序列，其组树中组的结构如图 6 所示。

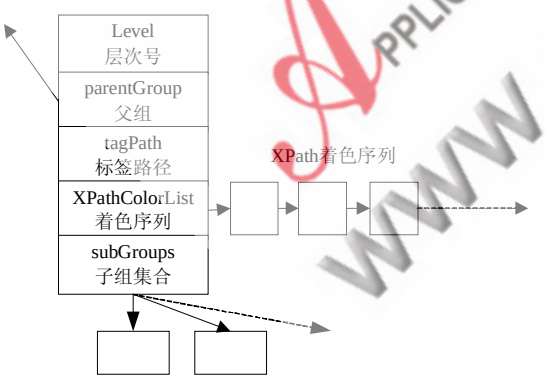


图 6 组结构图

2.3 GBRender 处理模型

基于组树的动态建立与组着色机制，GBRender 的任意组，仅当首次出现时进行 XPath 着色处理，之后则根据着色情况进行处理。

GBRender 的查询处理，其数据输入是 XML 的 SAX 解析事件流，包括 startDocument、endDocument、startElement、endElement 和 Text 事件，这里只给出 GBRender 处理模型在每一类事件的响应动作，而不讨论数据缓存处理。

```
startDocument; initiate the GroupTree GT and context
                                        //初始化组树 GT
StartElement(element)          // element 为新接收元素
If(not existsGroup(element))InsertGroupAndDrawXpathColor
(element);      //如果不存在该组，则生成该组并进行着色
                                        产生着色序列
    setCurrentElement(element); //置 element 为当前元素
Text (value)                          // 当前元素的值
    Add value into currentElement.currentValue;
                                        // 增加到当前元素值
EndElement(element)
    ProcessXPathColor(element);    // 根据当前元素的着
                                        色序列信息进行处理
    setCurrentElement(element.parentGroup); //
EndDocument
    cleanUp( );                        // 处理全部结束，清理
```

2.4 GBRender 查询模型的主要特征

GBRender 查询模型处理的操作方向与目前主要的查询处理不同，分析器并不是与 XPath 绑定，而是绑定到 XML 文档结构上，它对 XPath 只在组首次出现时进行着色的处理，之后则根据着色情况来进行处理。这种模型具有较强的灵活性，主要特点如下：

(1)对 XML 文档结构无需任何考虑。因为以接收的流数据为依据来动态建立组树。根据 XML 流数据的顺序性与元素之间关系的局部性，组树的访问总是发生在相邻元素之间，因此效率很高。

(2)无需依赖 XPath 分析器。一旦着色，相当于就对该组进行了确定化。同时，对于简单的 XPath 表达式在当前组即可直接判定作出处理。

(3)可以方便地利用现有的其他 XPath 查询处理机制，如自动机模型。因为在某种程度上，着色类似确定化 XPath，当处理复杂谓词时，其着色信息处理可以方便地吸收自动机模型的思想。

(4)非常容易扩展 XPath 查询数目，着色序列长度的增加对效率影响很小。

(5)方便进行类似关键词查询的流数据处理。如不考虑数据来源的结构，用户只关心信息中的 author 或 name 的信息。本文模型可以很容易且高效的去完成处理，仅需传入//author、//name 的 XPath 表达式串，就可以应用于任意的 XML 流数据源。

3 GBRender 的主要算法

以单枝查询为例简要介绍 GBRender 查询模型中的着色算法与处理算法。

在单一分枝的查询中，由于 XPath 表达式中仅有//与/关系，组标签对于 XPath 路径中的元素只存在 3 种关系，即无、中间元素或终结元素，而本文关心的正是那些终结元素且此处暂不需考虑缓存，为了便于描述，用

Color.Empty、Color.Mid 以及 Color.End 来表示 2 种着色状态。

(1)单一分枝下的组着色算法：

```
GroupRendering(G, XPaths)
输入：组 G 是当前元素所在组，表达式串 XPaths 是 XPath 表达式集合
{
  Foreach(xpath in XPaths)
  {
    If(G.tagPath not satisfy part xpath)      //未发现匹配，e.tagPath 为标签路径
      G.XPathColorList.add(Color.Empty);
    If(G.tagPath satisfy part xpath)          //有匹配
      If(G.tagPath is 终结元素)               //是终结元素
        G.XPathColorList.add(Color.End);
      else                                    //不是终结匹配但匹配中间元素
        G.XPathColorList.add(Color.Mid);
  }
}
```

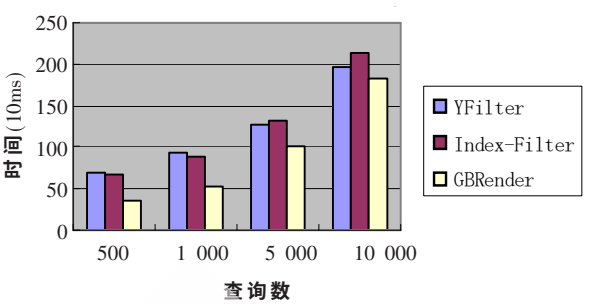
(2)endElement 时根据组着色情况对元素进行处理的算法：

```
ProcessXPathColor (G)
输入：组 G 是当前元素所在组
{
  Foreach(color in G.XPathColorList)
  {  //单枝查询情况下只需要对 Color.End 元素进行处理
    Costomeri ++;                             // XPath 逐个对应
    If(color is Color.End)
    {
      ToCustomer(Customeri, G.currentElement.Value); // 输出结果
    }
  }
}
```

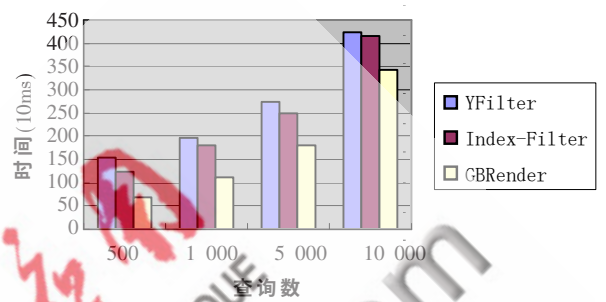
4 实验

本文实现了 GBRender 查询模型并应用在单枝查询上，软件环境是 Windows xp+eclipse3.3+jdk1.5。硬件环境是：联想旭日 410M 笔记本电脑，配置为：CPU 双核 T2080，内存 1 GB，硬盘 120 GB。

在实验中，综合了文档大小与查询数目的多少并与 YFilter 和 Index-Filter 作比较，结果发现，当文档相对较大时查询数量无论多少，GBRender 都显得非常有效；当文档相对较小而查询数量较大时，GBRender 与 YFilter 较为有效。实验结果如图 7 所示。



(a)文件大小 10 KB



(b)文件大小 100 KB

图 7 实验结果

本文提出了一种新的 XML 数据流 XPath 查询模型 GBRender，给出了该模型的结构特征、处理机制以及单枝查询下的多 XPath 查询算法和组着色的概念。通过大量的实验，证明了 GBRender 模型对 XML 任意数据流查询的有效性及其适用性强的优点。

GBRender 模型具有较强的适应性与灵活性，对某些结构简单的 XPath 查询极其有效。但在根据着色信息进行处理机制上有待进一步的研究。今后主要的工作是如何有效地在组着色机制中利用已有的好的查询机制或新的有效的处理机制进行复杂 XPath 查询处理，进一步完善系统。

参考文献

[1] BERGLUND A, BOAG S, CHAMBERLIN D, et al. XML path language (XPath)2.0 W3C working draft 16. Technical Report, WD-xpath20-20020816, World Wide Web Consortium, 2002.http://www.w3.org/TR/2002/WD-xpath2002-08-06.

[2] BRUNO N, GRAVANO L, KOUDAS N, et al. Navigation-vs. index-based XML multi-query processing.In: Dayal U, Ramaritham K, Vijayaraman TM, eds.Proc of the 19th Int'l Conf. on Data Engineering (ICDE 2003). Bangalore: IEEE Computer Society, 2003:139-150.

[3] ZWON J, RAO P, MOON B, et al. FiST:scalable XML document filtering by sequencing twig patterns.Proc. of the 31st Int'l Conf on Very Large Data Bases (VLDB 2005). Trondheim:VLDB Endowment, 2005. 217-228.

[5] 杨卫东，王清明，施伯乐.针对 XML 流数据的复杂 Twig Pattern 查询处理.软件学报，2007，18(4)：893-904.http://www.jos.org.cn/1000-9825/18/893.htm

[6] DIAO Y, FISCHER P. YFilter: efficient and scalable filtering of XML documents. In: Proc of the 18th Int'l Conf. on Data Engineering, 2002:341-345.

(收稿日期：2009-03-04)