基于 XQuery 的商业报告查询引擎的设计与实现

王晓琳[1]，朴 勇[1]，王秀坤[2]
(1. 大连理工大学 软件学院，辽宁 大连 116621；
2. 大连理工大学 电信学院，辽宁 大连 116621)

摘 要：基于 XQuery 查询语言的 XBRL 查询引擎首先借助 JavaCC 工具处理输入的 XQuery 语句形成抽象语法树，而后根据 XQuery 查询特点编写程序遍历此语法树来简化查询语言的处理流程，降低查询匹配的复杂度，提高查询效率，利用“SAX+ DOM”方式解析 XBRL 文件并提取 XQuery 语句所查询的数据信息。SAX 方法可以提高查询效率并节省内存消耗，DOM 方法可以支持对 XBRL 文件的上下文相关查询及频繁查询。实验证明，将二者结合起来应用满足高查询效率和低内存消耗双重需求。

关键词：XBRL；XQuery；解析器
中图分类号：TP311.52 **文献标识码**：A **文章编号**：1674-7720(2013)12-0001-03

Design and implementation of XBRL parser based on XQuery

Wang Xiaolin[1], Piao Yong[1], Wang Xiukun[2]
(1. School of Software, Dalian University of Technology, Dalian 116621, China;
2. School of Computer Science and Technology, Dalian University of Technology, Dalian 116621, China)

Abstract: The XBRL parser proposed in this paper firstly processes XQuery statement to abstract syntax tree by the JavaCC tool, then traverses AST according to the characteristics of XQuery inquires to simplify query language processing flow as well as reduce complexity of match. After the grammar analysis of XQuery, we use "SAX + DOM" mode to analyze and extract the relevant information of XBRL files, the SAX method can improve the query efficiency and save memory consumption, DOM method can support to related query and frequent query XBRL files. With the "SAX+DOM" mode, the XBRL parser can meet the double demand of high query efficiency and low memory consumption in practical XBRL project.

Key words: XBRL; XQuery; parser

XBRL[1](可扩展商业报告语言)是 XML 语言的扩展应用，主要面向金融财政领域，对其进行数据挖掘和分析具有重大意义，在实际项目中如何方便高效率地对其进行数据提取并降低内存消耗是需要解决的问题。

W3C 制定的 XQuery 语言中的 FLOWR[2]可以高效率地对 XBRL 进行查询处理，目前国内外有很多支持 XQuery 查询语言的查询引擎，如国外开源的 XQEngine 是一个基于 JavaBean 的用来查询本地 XML 文档的 XQuery 查询引擎，该引擎使用 SAX[3]解析方式来解析并提取 XML 中的所需信息，节省了内存也保证了查询速度。但是由于 XBRL 中的实例文件是承载财政数据的文件，其节点间存在密切的数据关联关系，用户多使用频繁查询及复合查询等查询实例文件进行数据分析，“边读边扔数据”的 SAX 方式无法满足这一需求。本文提出的 SAX+DOM 方式可以解决这一问题，同时在一定程度上比纯 DOM[4]读取方式节省了大量内存。

图 1 给出了基于 XQuery 的商业报告查询引擎的整体框架，主要分为 XQuery 预处理模块、查询优化器和查询执行器 3 个模块。传统的查询优化包括 Schema 模式

图 1 商业报告查询引擎平台框架

优化[5]、XQuery 语句查询重写、XML 文档索引技术、XQuery 语句规范化等[6]，而本文涉及的解析方式创新及抽象树的遍历简化则是从查询引擎内部实现角度对 XQuery 查询进行了优化，下面着重对查询引擎内部优化中的 XQuery 预处理模块及查询执行器进行具体阐述。

1 遍历 Javacc 生成的抽象语法树

XQuery 查询语言被设计用来查询 XML（XBRL 本质上也为 XML）文件，其语法中的 FLWOR 表达式是最重要的，它代表 For-Let-Where-Order by-Return。For 子句通过将节点绑定到变量，以便继续循环遍历序列中的每一个节点；let 子句为一个变量赋一个值或一个序列；return 子句定义每个元组要返回的内容；对于 where 子句，如果其有效布尔值为真，那么该元组就被保留，并且它的变量绑定用在 return 子句中，反之该元组就被废弃[7]。

在查询预处理模块，通过 JavaCC[8]工具和 W3C 已定义的 XQuery 语法规则来进行词法分析，将输入的 XQuery 语句查询语句中的每个成分都变成带有特定属性的节点并生成与之相对应的类（例如显示的是MainModule 等，则语法解析时 JavaCC 中的 JJtree 以这些类为节点在内存中生成树形结构），然后编写程序遍历抽象语法树。下面的算法流程为遍历 FLOWR 表达式中 For 语句对应的抽象语法树分支的流程，其中 MainModule 等均为使用 JavaCC 工具并根据 W3C XQuery 规范中 XQuery BNF 定义生成的语法树中的抽象树节点名字。XQuery 语句解析算法的流程图如图 2 所示，首先取得根节点，XQuery 开始处理语法树的内容，循环三次依次得到 QueryList、Module 和 MainModule 节点。

判断节点 MainModule 是否为 null，如为 null，则报错；如果不为 null，取出第一个节点 prolog 和第二个节点 QueryBody，分别对这两个节点进行分析处理。

对 prolog 节点进行处理，左节点 prefix 和右节点 url 为命名空间的格式，保存为名值对。在实际项目操作中，QueryBody 环节只会出现一个 FLOWR 语句，很少会有嵌套语句，本文只给出 FLOWR 的子节点为 ForClause 的情况，其他子节点暂不考虑。

判断 ForClause 子节点类型，若为用来表示变量所对应的 Xpath 值 PathExpr 节点，则取出 PathExpr 节点的子节点判断其类型。节点为函数的情况下则调用预定义函数，根据字段名称在节点树中定位并保存其内容；如节点类型为分隔符，则表示任意节点都可以作为下一个节点字段的起始状态。最后将 PathExpr 节点的处理结果 ResultList 与 Varname 变量节点对应并保存。

2 结合 SAX 和 DOM 解析 XBRL 文件

对于 XBRL 文件来说，存在 DOM 和 SAX 两种基本的解析手段。如果进行一次性查询或非频繁性查询，建议采用内存消耗小的基于事件驱动模型的 SAX 解析方式，无需像 DOM 解析方式中为所有节点创建对象。但当需要大量上下文相关查询及频繁查询时，尤其在查询后经修改合并而得到的结果集上进行复查询，则采用 DOM 方式为好。

XBRL 文件解析分为两种情况：(1)分类标准比较稳定，对其进行首次框架解析后将在数据库中将其保存，可以采取单独的 SAX 解析方式简单地进行逐条解析；(2) 实例文件承载大量重要数据，用户多使用频繁查询及复合查询等查询实例文件来进行数据分析，在这种情况下 DOM 解析方式是最好的选择。但是实例文件数量

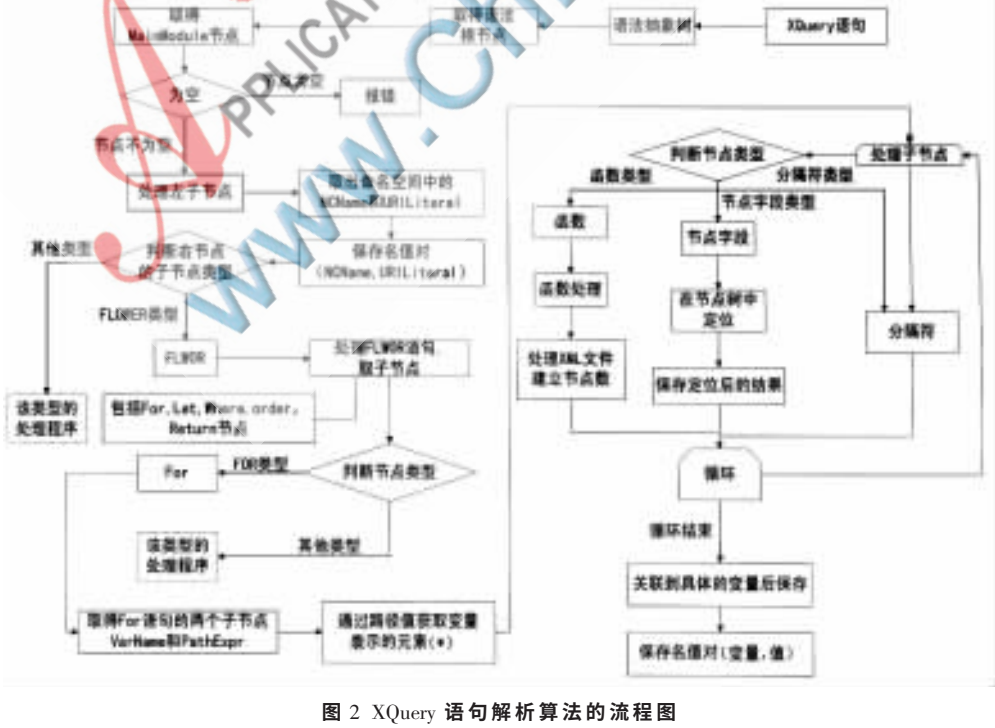

图 2 XQuery 语句解析算法的流程图

较多时，为 XBRL 实例文件中所有节点创建对象会大大增加内存需求。这时需要考虑将这两种方式结合起来使用。在读取实例文件数据时，利用 SAX 只读取用户所需要的若干小数据文件，并为小数据文件在内存中建立 DOM 对象，每次读取新的数据片段，则在已存在的 DOM 对象上进行添加或者修改等操作。这样做既有效地节省了大量资源，也充分利用了 DOM 对象的易操作性。

图 3 为两种解析方式结合的工作原理，其处理过程为：首先 SAX 对象 objSAX 的 parse 方法被调用来解析 XBRL 实例文件，在 startElement 事件中进行元素筛选，如为所需元素，则保存该元素及其子节点元素，在 endElement 事件中调用 DOM 对象的 loadXML()方法将数据转化为 DOM 树，从而可以方便下一步的数据处理(如频繁查询、信息更新等)。这样不仅保证了快速解析，也在更大程度上支持 XQuery 的上下文查询及信息检索。

图 3 SAX 和 DOM 解析方式结合工作原理

3 系统实验结果

实验环境采用 Inter Pentium Dual E2160 @1.80 GHz、1 GB 内存、Windows XP。

本文引擎解析文件时利用 SAX+DOM 方式，下面的实验结果证明了使用这种方式的优势，如图 4 所示。使用 SAX+DOM 方式处理时，只需将所需内容装入内存，故文档处理的时间不会因为文档的大小发生太大变化，事件均小于 0.003 s。而采用 DOM 方法处理文档，当文档大小达到一定程度时，需要使用硬盘的虚拟内存，其处理时间也会急剧增大。

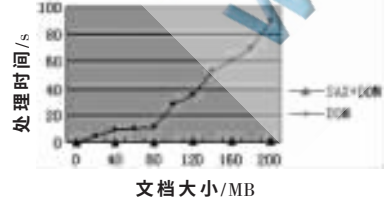

图 4 两种方法解析实例文件时间比

本文所提出的查询引擎简化了 XQuery 语句解析流程和复杂度，节点寻址路线相对于开源软件 XQEngine 变得简单。本实验利用上述两种引擎对中国 Taxonomy 的 cas_core_2010-09-30.xsd 文件进行查询，查询语句如图 5 所示，两个解析器都正确地返回结果 2847 个节点。本文提出的查询引擎用时 1.25 s，XQEngine 因其复杂的 XQuery 解析算法使得用时稍长，为 3.7 s。

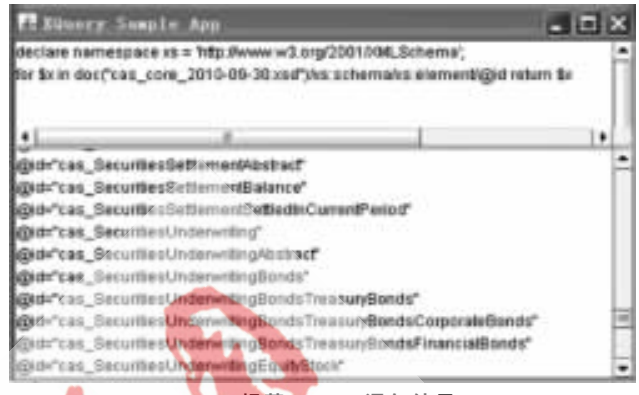

图 5 规范 XQuery 语句结果

本文在深入剖析 XQuery 查询语法特点后，结合“SAX+自定义 DOM”文件解析方式完成对 FLWOR 抽象语法树的遍历及查询信息读取，来简化 XQuery 语句的查询处理流程并降低查询匹配的复杂度，设计并实现了 XBR 查询引擎，提高了查询效率，降低了内存消耗。该引擎在项目中的数据信息查询模块起着重要作用。

参考文献

[1] 吕科，刘晓峰.XBRL 技术原理与应用[M].北京：电子工业出版社，2007.

[2] WALMSLEY P. XQuery 权威指南[M].王银辉，译.北京：电子工业出版社，2009.

[3] 冯进，丁博，史殿习，等.XML 解析技术研究[J].计算机工程与科学，2009，31(2)：120-124.

[4] 杨波.DOM 解析器 OnceDOMParser 的设计与实现[D].北京：中国科学院研究生院(软件研究所)，2005.

[5] 孟晓峰，王宇，王小峰. XML 查询优化研究[J]. 软件学报，2006，17(10)：2069-20.

[6] PAPARIZOS S, PATEL J M, JAGADISH H V. SIGOPT: Using schema to optimize XML query processing internati-onal[C]. Istanbul: International Conference on Data Engi-neering, 2007.

[7] 华珊珊，谢铉洋.XML 查询语言 XQuery 的研究与实现[J].计算机技术与发展，2009，19(4)：48-50.

[8] VISWANDAHA S, SANKAR S. Java Compiler Compiler (JavaCC)-The java papser generator[J].URL:https://javaCC.dev.java.net.2006.

(收稿日期：2013-01-30)

作者简介：

王晓琳，女，1988 年生，硕士研究生，主要研究方向：XML 数据库存储及优化。

朴勇，男，1975 年生，副教授，主要研究方向：智能计算与信息处理软件工程。

王秀坤，女，1945 年生，教授，主要研究方向：数据库决策支持系统。