Data & Knowledge Engineering 103 (2016) 1–18

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

LCA-based algorithms for efficiently processing multiple keyword queries over XML streams

Evandrino G. Barros[a], Alberto H.F. Laender[b], Mirella M. Moro[b], Altigran S. da Silva[c]

[a]Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, Brazil
[b]Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
[c]Universidade Federal do Amazonas, Manaus, Brazil

ARTICLE INFO

Article history:
Received 5 September 2014
Received in revised form 18 December 2015
Accepted 13 March 2016
Available online 31 March 2016

Keywords:
Multi-query processing
Keyword-based queries
XML streams
LCA semantics

ABSTRACT

In a stream environment, differently from traditional databases, data arrive continuously, unindexed and potentially unbounded, whereas queries must be evaluated for producing results on the fly. In this article, we propose two new algorithms (called SLCAStream and ELCAStream) for processing multiple keyword queries over XML streams. Both algorithms process keyword-based queries that require minimal or no schema knowledge to be formulated, follow the lowest common ancestor (LCA) semantics, and provide optimized methods to improve the overall performance. Moreover, SLCAStream, which implements the smallest LCA (SLCA) semantics, outperforms the state-of-the-art, with up to 49% reduction in response time and 36% in memory usage. In turn, ELCAStream is the first to explore the exclusive LCA (ELCA) semantics over XML streams.

A comprehensive set of experiments evaluates several aspects related to performance and scalability of both algorithms, which shows they are effective alternatives to search services over XML streams.

1. Introduction

A stream data processing environment presents challenges that go beyond the data and query processing on traditional database systems [41]. Specifically, data arrive continuously in the form of *streams*, i.e., data are unindexed and potentially unbounded. Also, query processing must evaluate data as they come in and produce results on the fly, in one sequential scan and with minimum space for handling temporary results [1, 13]. As XML has become the standard for data exchange, processing XML data streams has soon conquered its place in the research and industry communities [41].

As an example of real-world applications, data streams are commonly used with monitoring sensors such as those in airplanes and weather stations. Indeed, sensor monitoring uses many sensors distributed in the physical world that generate data streams to be grouped, monitored and analyzed [5, 29, 37]. Another example includes network traffic management systems, which monitor various data streams (characterized as unpredictable, arriving at a high rate, and including both packet traces

E-mail addresses: ebarros@decom.cefetmg.br (E. Barros), laender@dcc.ufmg.br (A. Laender), mirella@dcc.ufmg.br (M. Moro), alti@icomp.ufam.edu.br (A. da Silva).

http://dx.doi.org/10.1016/j.datak.2016.03.001
0169-023X/© 2016 Elsevier B.V. All rights reserved.

and network performance measurements) [10, 35]. Finally, there is also a huge set of general purpose streaming data environments in which data are not specified in XML [8, 42]. Our techniques may also work on those, provided that a proper mapping is employed, such as the one proposed by Mendell et al. [31].

More recent applications that present large amounts of data streams are online social networks, most of them based on XML. In these environments, millions of users share interests, opinions, likes and news, which provide instantaneous information to evaluate social and market trends [30]. Likewise, RSS feed-based applications make large use of data streams. RSS is an XML feed format to publish frequently updated entities such as news headlines and blog entries on the Web. RSS popularity has motivated XML stream-oriented applications, such as the RSS Watchdog system, which is capable of clustering news and monitoring instant events over multiple XML streams [20].

XML streams must be processed rapidly and without retention. Retaining streams may cause data loss due to the large data traffic in continuous processing. This scenario becomes more complex when thousands of queries must be processed simultaneously. Different approaches explore single and multiple query processing. However, they are based on structural languages such as XPath, XQuery and SQL-like ones [11, 17, 18, 22, 23, 26, 33, 36, 37, 41], which requires knowledge of their syntax and the underlying data structure to formulate queries. Keyword-based languages are a usual approach to submit queries informally, because they require minimal or no schema knowledge to formulate queries [4, 9, 19, 21, 25, 39, 43, 44, 48]. Most keyword-based methods for XML, however, address archived documents [25, 44, 48]. More recent techniques have focused on keyword-based search algorithms for XML streams, but they only run one query at a time [4, 19, 39]. To the best of our knowledge, only the algorithm proposed by Hummel et al. [21] address the problem of processing multiple keyword-based queries over XML streams and, therefore, it is the current state-of-the-art. However, it is specific to the SLCA search semantics.

In general, keyword-based algorithms consider the lowest common ancestor (LCA) semantic [4, 9, 19, 21, 39, 43, 44, 48]. Specifically, the LCA of two nodes u and v in an XML tree is the common ancestor of u and v that is located farthest from the root. The most popular LCA-based algorithms use the smallest LCA (SLCA) and the exclusive LCA (ELCA) semantics. Particularly, ELCA handles the ambiguity that might exist in an XML document because the same content can occur at different levels, such as keywords that correspond to XML labels occurring in different schema elements [2]. Thus, ELCA is considered one of the most effective semantics because it returns a larger number of results [48].

As this work shows later, current approaches are limited to one or more of the following: processing one-single query at a time, not supporting schema-free keyword queries, and returning the whole document or insignificant parts of it with actual results. With such challenges in mind, we propose two new, efficient LCA-based algorithms for processing multiple keyword-based queries over XML streams: SLCAStream and ELCAStream. Both algorithms exploit processing properties based on the LCA semantics and introduce optimization strategies that improve the overall performance. SLCAStream provides a new approach to SLCA evaluation that avoids the usual bitmap processing strategy [4, 21, 39]. Hence, it significantly improves existing SLCA algorithms' response time and memory consumption. Then, ELCAStream extends SLCAStream for the ELCA semantics, making it the *first* ELCA-based algorithm for processing multiple queries over XML streams. Our extensive experiments analyze the performance and scalability of the proposed algorithms against the state-of-the-art. The results show that both SLCAStream and ELCAStream are efficient alternatives for processing keyword-based queries over XML streams. In summary, the contributions of this article are:

- Two new algorithms for processing multiple keyword-based queries over XML, called SLCAStream and ELCAStream, which provide, respectively, new approaches to SLCA and ELCA evaluation that avoid the usual bitmap processing strategy, thus significantly improving response time and memory consumption of existing algorithms. Moreover, ELCAStream is the *first* ELCA-based algorithm for XML query processing in a stream environment.
- A thorough evaluation of the two proposed algorithms that includes a performance comparison against the state-of-the-art algorithm considering both time and memory consumption.

The rest of this article is organized as follows. Section 2 discusses related work. Section 3 addresses some fundamental concepts related to our work. Section 4 describes the two proposed algorithms, SLCAStream and ELCAStream, and Section 5 presents our experimental evaluation. Finally, Section 6 presents our conclusions and insights for future work.

2. Related work

XML query processing has solutions ranging from simple axis evaluation to full language specifications over *stored* data [14, 18, 22, 25, 34, 46, 47], and from simply filtering (find the documents that match a query [27,40,45]) to full querying (find the document parts that match a query [35, 37]) for *streaming* data. Considering streaming data, all XML processing techniques may be categorized according to three dimensions: language (XPath/XQuery or keyword), processing (single-query or multi-query) and output semantics (filtering or querying). Most of them are based on XPath and XQuery semantics, whose process depends heavily on the document schemas [24]. Instead of going through such an extensive state-of-the-art, this section focuses on XML stream environments with keyword-based semantics. Nonetheless, existing surveys on the literature cover specific topics on *stored* XML data, such as XPath and XQuery [14, 18], keyword search [28] and recent approaches on LCA, SLCA and ELCA-based semantics [46, 47, 49]. Finally, Fig. 1 puts everything in perspective: the three dimensions for *stream* processing, state-of-the-art examples and where our contributions fit in the big picture (i.e., keyword-based multi-querying process).