Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010) 209–224

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

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

ARAXA: Storing and managing Active XML documents

Cláudio Ananias Ferraz[a], Vanessa Braganholo[b], Marta Mattoso[a,*]

[a] Computer Science Department, COPPE/Federal University of Rio de Janeiro, Brazil
[b] Computer Science Department – IM/Federal University of Rio de Janeiro, Brazil

ARTICLE INFO

Article history:
Received 1 September 2008
Received in revised form
18 December 2009
Accepted 9 March 2010
Available online 3 April 2010

Keywords:
Active XML
Storage
Peer-to-peer

ABSTRACT

Active XML (AXML) documents combine extensional XML data with intentional data defined through Web service calls. The dynamic properties of these documents pose challenges to both storage and data materialization techniques. In this paper, we present ARAXA, a non-intrusive approach to store and manage AXML documents. We also define a methodology to materialize AXML documents at query time. The storage approach of ARAXA is based on plain relational tables and user-defined functions of Object-Relational DBMS to trigger the service calls. By using a DBMS we benefit from efficient storage tools and query optimization. Approaches without DBMS support have to process XML in main memory or provide for virtual memory solutions. One of the main advantages of ARAXA is that AXML documents do not need to be loaded into main memory at query processing time. This is crucial when dealing with large documents. The experimental results with ARAXA prototype show that our approach is scalable and capable of dealing with large AXML documents.

1. Introduction

Several aspects in real world nowadays are dynamic: dynamic Web pages, dynamic systems, dynamic databases, etc. In this dynamic world, interoperability is crucial. Web services provide simple and non-coupled access to service providers distributed over the Web, which makes application interoperation easier. On the other hand, XML documents have also been used to application interoperability. In this scenario, it is natural to think on dynamic documents. Such documents combine extensional content with intentional content, which is obtained through Web service calls. Abiteboul et al. [2] developed a framework to manipulate Active XML (AXML) documents. In their framework, the results of the service calls are embedded within the XML document. Fig. 1 shows an example of an AXML document. As defined by Abiteboul et al. [2], the <sc> nodes denote service calls. When called, the results of these services are inserted in the document as siblings of the corresponding <sc> node. This process is called *materialization*.

Such documents can be large, and since the tree-structure of XML documents is verbose, there may be problems to manipulate them in main memory. Alternative ways of storing and managing these documents are needed. Additionally, one should be able to pose queries to stored AXML documents. For this, an XQuery query engine or native XML DBMS could be used. However, the intentional content of AXML documents must be managed during query processing, that is, service calls must be coordinated. This is because the activation of service calls may be associated to some query criteria, that is, a service may need to be called to answer a given XML query. Service calls may also be completely disassociated from queries. They may need to be activated periodically, independently of query execution time. Due to all of these factors, AXML documents cannot be managed directly with available non-active XML management tools.

Abiteboul et al. [3,5] developed a platform to manage AXML documents. This platform is publicly available [9] and has strong "correction" properties, since it follows the AXML model, preserving document properties and types. Previous work on AXML materialization in this platform had mostly addressed typing control [25], XML query processing [1], and data and Web services replication [8]. In the first version of this platform, AXML documents were stored in the file system, which poses several drawbacks (security, indexing, etc.). Currently, AXML documents can be stored in the eXist [17] or Xyleme [42] XML DBMS. However, AXML materialization, i.e., query processing with service invocation is still processed apart from the DBMS. Thus, in this approach, queries need to be processed directly over those files, and documents still need to be loaded into main memory. This has serious scalability problems, especially when the documents are large. Thus, when storage and query capabilities are not within the same solution, the AXML document has to be manipulated by two different memory managers. An alternative approach would be to use a DBMS, since it provides

* Corresponding author at: Computer Science/COPPE/UFRJ, Federal University of Rio de Janeiro, P.O. Box 68511, 21941-972 Rio de Janeiro, RJ, Brazil.
Tel.: +55 21 2562 8694.
E-mail addresses: cferraz@cos.ufrj.br (C.A. Ferraz), braganholo@dcc.ufrj.br (V. Braganholo), marta@cos.ufrj.br (M. Mattoso).

1570-8268/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2010.03.001

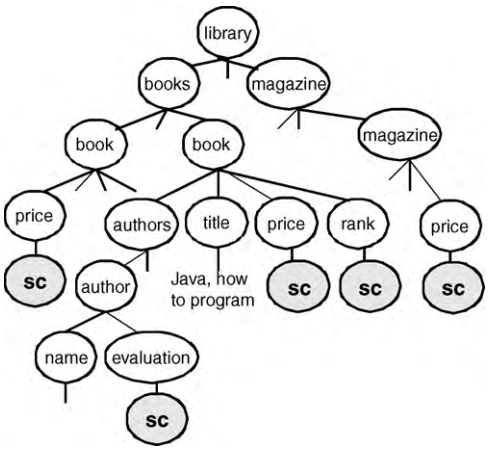

Fig. 1. Example of an AXML document.

both features, and, depending on how documents are stored, it can process queries without loading the documents entirely into main memory. This is an attractive solution for large documents, which are very frequent considering the verbose characteristic of XML. To use a DBMS two issues need to be addressed. One is the DBMS capability in storing and querying XML documents. The other is the DBMS ability in working with Web service calls.

There are several approaches to store and query XML documents in DBMS. Some use Relational DBMS [15,19,23,32,36,38], others use native XML storage [21,35]. However, the active part of AXML documents poses some challenges both to store and to query the documents. Relational and native XML DBMS do not support the active feature of such documents. Specifically, they do not know how to deal with the dynamicity of the content, nor with the external data sources (service providers). Relational DBMS is able to support some dynamicity through SQL *triggers*. However, service calls may need to be activated at query time and triggers cannot be activated by SQL SELECT clauses. Thus, they do not have the behavior nor the granularity needed to implement the active characteristic of AXML documents. Consequently, they are not the best alternative to the problem of storing and querying AXML.

Our proposed solution, ARAXA,[1] uses Object-Relational (OR) DBMS. Although they cannot explicitly model the active behavior of AXML, OR-DBMS are capable of dealing with complex objects and associated methods. Methods can be seen as an active component. This allows us to create a class of *active objects* that are responsible for coordinating service calls and their execution. By using these resources, services can be called within SQL queries. It is also possible to create an agent that verifies the periodicity in which a service needs to be called, and manages these calls automatically. To support XQuery within OR-DBMS, we can use existing XML-relational storage mappings [16,22,24,39], and consequently, existing algorithms that translate XQuery to SQL queries [22]. By using these algorithms together with our service call functions, queries can be processed with no need to load source documents into main memory, so very large documents can be processed efficiently without memory limitations. We focus on using standard resources in OR-DBMS, so that our solution can be applied to any OR-DBMS. In our solution, we keep the properties of the formal foundation of AXML documents [5,6]. At the same time, we offer more sophisticated storage resources allied with consolidated query processing capabilities.

[1] ARAXA is a Brazilian city and a Portuguese acronym that loosely translated to English means ***An*** *object-****R****elational* ***A****pproach to store* ***X****ML Documents with* ***A****ctive elements*.

In summary, we have two main goals, where solutions and experimental results are our main contributions:

(1) Scalability, which we address by managing query processing and service calls materialization in a single environment (the DBMS). This allows us to handle large XML documents without needing to load them into main memory. Processing an XML document in main memory is a requirement in previous solutions and has strong limitations even for small XML documents;
(2) Single environment: we take advantage of DBMS algorithms to deal with materialization and query processing in a single environment (the DBMS itself), which contributes to improving the materialization process.

The limitation of our approach resides in the mapping between OR and XML. However, our results show a negligible overhead in this transformation. In fact, this is highly compensated by the fact that an organization can now keep their traditional data and AXML documents in a single repository, thus maintenance cost can be reduced, among other benefits such as data integration. Additionally, XML support in these DBMS is always improving. Notice that our solution is DBMS independent. Any OR-DBMS can be used.

This paper is an extended version of a previous published paper [18]. In this paper, we detail our approach and present extensive experimental results. Moreover, we present a detailed description of our query processing methodology. This paper is organized as follows. Section 2 presents the Active XML Platform developed by the INRIA-GEMO group. Section 3 overviews related work and analyzes current solutions for storing and querying AXML documents. In Section 4 we identify the difficulties to the problem and propose a storage schema to AXML documents. Section 5 presents AXML query processing in our storage approach while Section 6 presents the software architecture of ARAXA and its prototype. Our experimental results are discussed in Section 7. Finally, Section 8 concludes this work.

2. Background: Active XML

The Active XML platform developed by the INRIA-GEMO group [9] is an open-source framework to support Active XML documents in a P2P distributed environment. Abiteboul et al. [2] defined a formal model for an AXML document where several materialization strategies can be applied [25,28]. The materialization of Active XML data can be either explicitly requested by the user or implicitly triggered by queries that require the (materialized) content of a document.

The internal architecture of an AXML peer, shown in Fig. 2, relies on the following modules [9]:

- The *AXML storage*, which provides persistent storage for AXML documents.
- The *evaluator*, whose role is to trigger the services calls embedded inside AXML documents and to update the latter accordingly.
- The *XQuery processor*, which executes XQuery queries.

Peers communicate with each other only by the means of Web service invocations, through their *SOAP wrapper* modules. They can exchange XML data with any Web service client/provider, and AXML data with AXML peers.

In this section we review some characteristics of the materialization of AXML documents as processed by the AXML platform. These specificities were defined by the AXML model [6]. Particularly, we discuss their approach in handling the active part of the documents. In Section 2.1 we show how services are analyzed for query processing and then in Section 2.2 some materialization