A

Active XML

Serge Abiteboul[1], Omar Benjelloun[2], and Tova Milo[3]
[1]Inria, Paris, France
[2]Google Inc., New York, USA
[3]School of Computer Science, Tel Aviv University, Tel Aviv, Israel

Synonyms

Active document; AXML

Definition

Active XML documents (AXML documents, for short) are XML documents [12] that may include embedded calls to Web services [13]. Hence, AXML documents are a combination of regular "extensional" XML data with data that is defined "Intentionally," i.e., as a description that enables obtaining data dynamically (by calling the corresponding service).

AXML documents evolve in time when calls to their embedded services are triggered. The calls may bring data once (when invoked) or continually (e.g., if the called service is a continuous one, such as a subscription to an RSS feed). They may even update existing parts of the document (e.g., by refreshing previously fetched data).

Historical Background

The AXML language was originally proposed at INRIA around 2002. Work around AXML has been going there in the following years. A survey of the research on AXML is given in [13]. The software, primarily under the form of an AXML system, is available as open source software. Resources on Active XML may be found on the project's website [11].

The notion of embedding function calls into data is old. Embedded functions are already present in relational systems as stored procedures. Of course, method calls form a key component of object databases. For the Web, scripting languages such as PHP or JSP have made popular the integration of processing inside HTML or XML documents. Combined with standard database interfaces such as JDBC and ODBC, functions are used to integrate results of (SQL) queries. This idea can also be found in commercial software products, for instance, in Microsoft Office XP, SmartTags inside Office documents can be linked to Microsoft's.NET platform for Web services.

The originality of the AXML approach is that it proposed to *exchange* such documents, building on the fact that Web services may be invoked from anywhere. In that sense, this is truly a language for distributed data management. Another particularity is that the logic (the AXML language) is a subset of the AXML algebra.

Looking at the services in AXML as queries, the approach can be viewed as closely related to

L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_782-2

recent works based on XQuery [14] where the query language is used to describe query plans. For instance, the DXQ project [7] developed at ATT and UCSD emphasizes the distributed evaluation of XQuery queries. Since one can describe documents in an XQquery syntax, such approaches encompass in some sense AXML documents where the service calls are XQuery queries.

The connection with deductive databases is used in [1] to study the diagnosis problems in distributed networks. A similar approach is followed in [8] for declarative network routing.

It should be observed that the AXML approach touches upon most database areas. In particular, the presence of intentional data leads to views, deductive databases, and data integration. The activation of calls contained in a document essentially leads to active databases. AXML services may be activated by external servers, which relates to subscription queries and stream databases. Finally, the evolution of AXML documents and their inherent changing nature lead to an approach of workflows and service choreography in the style of business artifacts [10].

The management of AXML document raises a number of issues. For instance, the evaluation of queries over active documents is studied in [2]. The "casting" of a document to a desired type is studied in [9]. The distribution of documents between several peers and their replication is the topic of [4].

Foundations

An AXML document is an (syntactically valid) XML document, where service calls are denoted by special XML elements labeled *call*. An example AXML document is given in Fig. 1. The figure shows first the XML serialized syntax, then a more abstract view of the same document as a labeled tree. The document in the figure describes a (simplified) newspaper homepage consisting of (i) some extensional information (the name of the newspaper, the current date, and a news story) and (ii) some intentional information (service calls for the weather forecast and for the current exhibits). When the services are called, the tree evolves. For example, the tree at the bottom is what results from a call to the service *f* at weather.com to obtain the temperature in Paris.

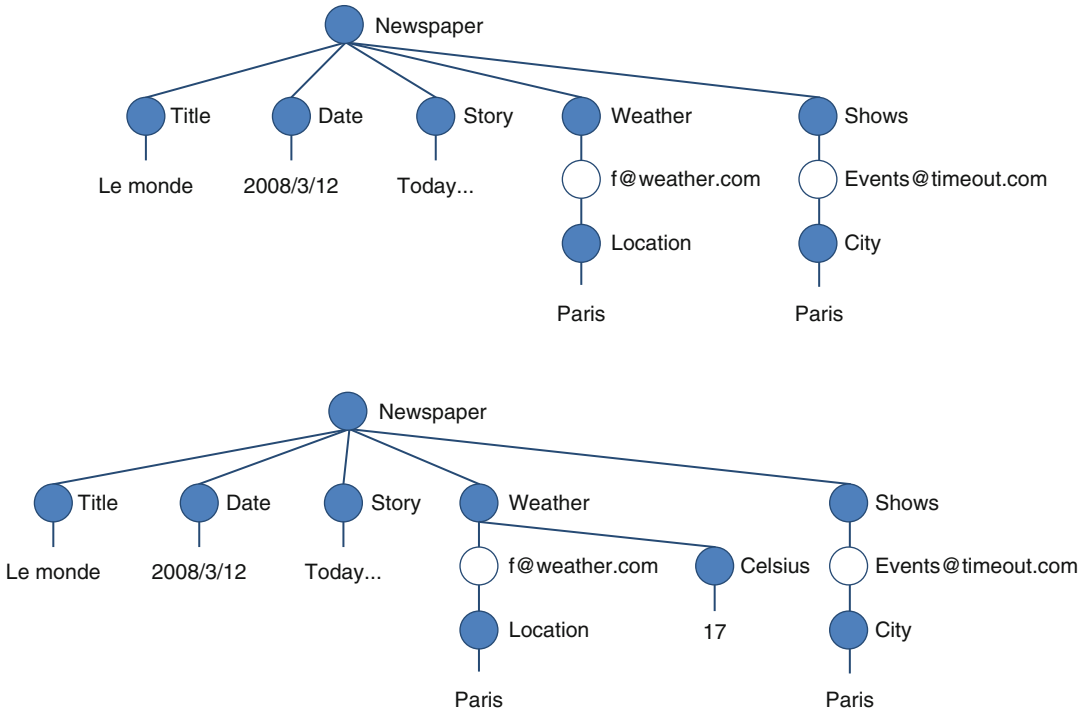
AXML documents fit nicely in a peer-to-peer architecture, where each peer is a persistent store of AXML documents, and may act both as a client, by invoking the service calls embedded in its AXML documents and as a server by providing Web services over these documents.

Two fundamental issues arise when dealing with AXML documents. The first one is related to the exchange of AXML documents between peers, and the second one is related to query evaluation over such data.

Documents Exchange: When exchanged between two applications/peers, AXML documents have a crucial property: since Web services can be called from anywhere on the Web, data can either be materialized before sending or sent in its intentional form and left to the receiver to materialize if and when needed. Just like *XML Schemas* do for standard XML, *AXML schemas* let the user specify the desired format of the exchanged data, including which parts should remain intentional and which should be materialized. Novel algorithms allow the sender to determine (statically or dynamically) which service invocations are required to "cast" the document to the required data exchange format [9].

Query evaluation: Answering a query on an AXML document may require triggering some of the service calls it contains. These services may, in turn, query other AXML documents and trigger some other services, and so on. This recursion, based on the management of intentional data, leads to a framework in the style of *deductive databases*. Query evaluation on AXML data can therefore benefit from techniques developed in deductive databases such as Magic Sets [6]. Indeed, corresponding AXML query optimization techniques were proposed in [1, 2].

Efficient query processing is, in general, a critical issue for Web data management. AXML, when properly extended, becomes an algebraic language that enables query processors installed on different peers to collaborate by exchanging streams of (A)XML data [14]. The crux of the approach is (i) the introduction of generic



Active XML, Fig. 1 An AXML document

services (i.e., services that can be provided by several peers, such as query processing) and (ii) some explicit control of distribution (e.g., to allow delegating part of some work to another peer).

Key Applications

AXML and the AXML algebra target all distributed applications that involve the management of distributed data. AXML is particularly suited for data integration (from databases and other data resources exported as Web services) and for managing (active) views on top of data sources. In particular, AXML can serve as a formal foundation for mash-up systems. Also, the language is useful for (business) applications based on evolving documents in the style of business artifacts and on the exchange of such information. The fact that the exchange is based on flows of XML messages makes it also well adapted to the management of distributed streams of information.

Cross-References

- Active Document
- BPEL
- W3C XML Query Language
- Web Services
- XML
- XML Information Integration
- XML Programming
- XML Publish/Subscribe
- XML Types

Recommended Reading

1. Abiteboul S, Abrams Z, Milo T. Diagnosis of asynchronous discrete event systems – datalog to the rescue! In: Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems; 2005. p. 358–67.
2. Abiteboul S, Benjelloun O, Cautis B, Manolescu I, Milo T, Preda N. Lazy query evaluation for active XML. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2004. p. 227–38.

3. Abiteboul S, Benjelloun O, Milo T. The active XML project, an overview. VLDB J. 2008;17(5):1019–40.
4. Abiteboul S, Bonifati A, Cobena G, Manolescu I, Milo T. Dynamic XML documents with distribution and replication. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2003. p. 527–38.
5. Abiteboul S, Manolescu I, Taropa E. A framework for distributed XML data management. In: Advances in database technology, Proceedings of the 10th International Conference on Extending Database Technology; 2006.
6. Bancilhon F, Maier D, Sagiv Y, Ullman JD. Magic sets and other strange ways to implement logic programs. In: Proceedings of the 5th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems; 1986. p. 1–15.
7. DXQ. Managing distributed system resources with distributed XQuery. http://db.ucsd.edu/dxq/
8. Loo BT, Condie T, Garofalakis M, Gay DE, Hellerstein JM, Maniatis P, Ramakrishnan R, Roscoe T, Stoica I. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2006. p. 97–108.
9. Milo T, Abiteboul S, Amann B, Benjelloun O, Ngoc FD. Exchanging intentional XML data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data; 2003. p. 289–300.
10. Nigam A, Caswell NS. Business artifacts: an approach to operational specification. IBM Syst J. 2003;42(3):428–45.
11. The Active XML homepage. http://www.activexml.net/
12. The Extensible Markup Language (XML) 1.0 (2nd edn). http://www.w3.org/TR/REC-xml
13. The W3C Web Services Activity. http://www.w3.org/2002/ws
14. The XQuery language. http://www.w3.org/TR/xquery