A STUDY OF NATIVE XML DATABASES
Document Update, Querying, Access Control and Application Programming Interfaces in Native XML Databases

M. Mercedes Martínez-González, Miguel A. Martínez-Prieto and María Muñoz-Nieto
Department of Computer Science, University of Valladolid, Campus 'Miguel Delibes' s/n, Valladolid, Spain

Keywords: Native XML databases, NXD, Standard APIs, Access control, Document update, Query language.

Abstract: Native XML databases (NXD) are called to play a crucial role in the near future. The experience with relational databases shows that standard methods for accessing and manipulating databases are necessary if wide acceptance of these database systems is to be expected in information systems whose applications should access databases preferably through standard APIs. In this paper, the current state of APIs that provide standard access to NXD, standard query languages and standard methods for document update and access control are analyzed from the perspective acquired with our experience using NXD in information systems. Our conclusions show the weak points which still need to be improved as compared with relational databases.

1 INTRODUCTION

Native XML databases (NXD) promise to play a key role in the near future as management systems for XML data (Garofalakis et al., 2004). There are several facts that favour these good expectations. XML is the preferred 'format' for storing non-structured data (Bourret, 2007; Bonifati and Cuzzocrea, 2007).

XML and its query language, XQuery (Boag et al., 2007), is a more flexible and less structured way of representing and querying a collection of data than relational databases are. XQuery is a World Wide Web Consortium (W3C) Recommendation since January 2007, which makes it stable, and it is now implemented (with different levels of conformance) in all database management systems that handle XML. Besides their inherent suitability for XML and XQuery features, NXD have the important advantage, over other database systems, that they free users from having to know document schema in advance (prior to designing the database).

However, these systems have still not won the 'XML market' as was predicted, and they still compete with XML-Enabled Databases for this market (Pardede et al., 2008).

An analysis of some of the possible reasons for this situation, resulting from our experience with NXD, is offered in this paper. It does not intend to be an exhautive analysis of NXD, for which a conference paper could not be enough, but to provide an updated position of the considered issues. Some issues, whose current situation is studied, are selected: document update, querying, access control, and the availability of Application Programming Interfaces (APIs) in NXD. Other important issues in databases, such as security or concurrency, are beyond the scope of this paper. Our reflection is guided by the comparison with relational databases, as they are nowadays the referent for database users.

The paper is structured as follows. Section 2 presents the problem we faced, the database issues treated in this paper, and the databases used in the experiments. Section 3 briefly introduces NXD and our findings. Finally, the conclusions are presented in section 4.

2 MOTIVATION

We work with legislative information systems, in which legal documents and additional metadata are represented with XML documents. A large amount of our data are normative texts and legal documents, whose nature implies that the resulting XML representations are document-centric XML documents[1].

[1]Following (Chaudhri et al., 2003) 'a document-centric XML document is one that captures unstructured data as in articles,

MartÃ■nez-GonzÃąlez M., MartÃ■nez-Prieto M. and MuÃśoz-Nieto M.
A STUDY OF NATIVE XML DATABASES - Document Update, Querying, Access Control and Application Programming Interfaces in Native XML Databases.
DOI: 10.5220/0001822300890092
In *Proceedings of the Fifth International Conference on Web Information Systems and Technologies (WEBIST 2009)*, page
ISBN: 978-989-8111-81-4

As NXD are well-suited for this type of XML document, and they are considered the best option as opposed to (relational) XML-enabled databases (Bourret, 2007; Bonifati and Cuzzocrea, 2007; Kolar and Loupal, 2006; Jagadish et al., 2002), we used an NXD to store our XML document collections. Several applications should work on it.

Some of our applications are learning tools, used by Law teachers and students in practical works. As in any other learning tool, a distinction of user roles needs to be managed (at least the roles of 'teacher', 'student' and 'administrator'). Besides, our teachers should be able to organize documents in repertoires (collections) according to their own criteria. Finally, the collections are queried and updated with the introduction or deletion of documents, but also with the modification of some documents as ,for instance, the comments that teachers provide to their students.

Moreover, we wanted the freedom to test different NXD systems without being obliged to modify our applications if the NXD system they accessed changed, or if a different NXD system was used.

Thus we expected to be able to:

- Query the database using a standard query language for XML. We searched a standard which played a role similar to SQL, which is the standard query language for RDBMS.
- Gain standard access to NXD database systems. We expected to be able to use an API which played a role similar to the one ODBC or JDBC play with relational databases.
- Create 'repertoires' of documents, which we would populate with XML documents. The repertoires are equivalent to the 'collection' concept used by NXD.
- Update the database, that is, to insert new XML documents into the repertoires, to delete documents, or to modify them.
- Support multiple users and user roles. Some of these users should be able to create their own repertoires, and to select whether they want to share their collections with other users or not.

These requirements are similar to the requirements we would have if working with relational databases. In relational databases, the creation, population and updates of the database are supported through standard query commands (*create table, update, insert, delete*). Inserting documents in collections is equivalent to inserting tuples in relations. The problem of sharing documents and/or collections with other users is equivalent to the problem of transferring permissions on data objects to other users in relational databases (access control). In this last type of database, permissions can be transferred and revoked with SQL commands, *grant* and *revoke*.

We have used two NXD systems, *eXist* (Meier, 2002; Meier, 2003) and *Sedna* (Fomichev et al., 2006) databases. Other possibilities among the most popular systems at the moment were Timber and XIndice (Apache XIndice). We chose eXists as it is a very popular system, is freeware, and it has better support for XPath 1.0 than XIndice (Kolar and Loupal, 2006). This last factor supposes a bigger variety of queries supported, which is an issue on which we wanted to have good support. On the other hand, Sedna released a new version 3.0 very recently, in April 2008[2], which is supposed to provide better support of client authentication and database users and privileges. Testing this new version that enhanced access control seemed to us an interesting possibility.

3 STANDARDS FOR NATIVE XML DATABASES

Native XML Databases are systems developed purely for storing XML documents. Their data models are flexible, so that documents do not need to be transformed to equivalent tables and columns, which means that they support the complexity of XQuery more efficiently than relational databases can. For document-centric XML documents, these systems are preferable to XML-enabled relational databases (Chaudhri et al., 2003; Bourret, 2007; Bonifati and Cuzzocrea, 2007; Kolar and Loupal, 2006; Jagadish et al., 2002).

Standard connectivity with NXD is achieved with a standard API, XML:DB API, promoted by the XML DB Initiative [3]. This API plays a role similar to JDBC or ODBC in relational databases. It has methods for connecting to the database, exploring metadata, executing queries, and retrieving results (Chaudhri et al., 2003). It uses XQuery as its query language and has been implemented by a number of native XML databases. This API is built around four core concepts: drivers, collections, resources, and services. *Drivers* encapsulate the database access logic. A *resource* can be either an XML resource or a binary large object. Finally, *services* may be requested to

books, or e-mails. This is opposed to data-centric documents, which capture structured data as that pertaining to a product catalog, an order, or an invoice.'

[2]http://www.cs.wisc.edu/dbworld/messages/2008-04/1207738519.html

[3]http://xmldb-org.sourceforge.net/xapi/

perform tasks like querying a *collection* with XPath or managing collections. The API considers that documents are stored in *collections*. This allows collections to be managed (*Get Collection*), documents to be inserted (methods *setContentAsDOM* for DOM documents, and use of a document handler for SAX documents), and a resource to be deleted (*removeResource* method).

XQuery (Boag et al., 2007) is a standard query language for XML documents, which has been a stable W3C recommendation since January 2007. XQuery plays an equivalent role to that played by SQL in relational databases for querying, providing the flexibility and power that the XML data model requires. It uses the structure of XML data in its queries. XQuery embeds XPath (Clark and DeRose, 1999) and includes 'SQL-like' capabilities by means of FLOWR expressions (FOR, LET, WHERE, ORDER BY, RETURN clauses). However, it does not include possibilities to update the database or to manage access control.

XML update is an interesting topic in the XML community. The XML DB Initiative investigated the XML update and released a proposal in 2000 (XUpdate Working Group, 2000). The W3C released the requirements and use cases for XML update as Candidate Recommendations in March 2008 (Robie and Chamberlin, 2008; Robie and Manolescu, 2008). From these, it obtained the updates facilities for XQuery, whose last version was released in August 2008 (Chamberlin et al., 2008). Nevertheless, updates are not yet treated in a standard way.

Following (Pardede et al., 2008), there are four main strategies for updates in NXD. There are NXD products that use their own proprietary language. Other products use a special language called XUpdate. A third strategy is to retrieve the XML document, update it by using an XML API, and to return the document to the database after updating. This is the strategy followed by the XML:DB API, as it follows from its 'Updating a Text XML Document' use case[4], shown in figure 1. The fourth option is to embed the update processes into XML language, as XQuery, within which would fall the XQuery Update Facility (Chamberlin et al., 2008). In fact, the *XQuery Update Facility 1.0* is an extension to XQuery.

The *XQuery Update Facility 1.0* is intended to be used to make persistent changes to instances of the XQuery 1.0 and XPath 2.0 Data Model. The operations it supports are: '*insertion of a node, deletion of a node, modification of a node by changing some of its properties while preserving its node identity, creation of a modified copy of a node with a new node identity*' (Chamberlin et al., 2008).

[4]http://xmldb-org.sourceforge.net/xapi/UseCases.html #N64ab4d

```
String id = "gladiator-2000";
XMLResource resource =
   (XMLResource) collection.getResource(id);
String doc = (String) resource.getContent();
// Change document ...
resource.setContent(doc);
collection.storeResource(resource);
```

Figure 1: Updating an existing Text XML Document stored in the database. Use case from XML:DB API.

Access control has also been the target of some investigation. In (Gabillon, 2004) an authorization model inspired in relational databases is proposed. It supports *read* and *write* privileges and is intended to be used with NXD that support the XML DB *XUpdate* language. However, this model has not been adopted by some popular NXD, such as eXist, which provide access control through proprietary extensions to the XML:DB API. In fact, the NXD used in our experiments, eXists and Sedna, provided their own proprietary methods for access control. That is, a change of the database system that stores the XML documents, would suppose changing the programming code part of any application that manages access control.

4 CONCLUSIONS

The current situation is that there is a standard query language, XQuery, and a standard API, the XML:DB API, but no standards for updates and access control. This situation generates a dependence of software applications on the NXD used, which is clearly a serious drawback. It seems quite reasonable that almost any application needs to update the uderlying database, and access control is a common need as well. For example, if we had used some of the Sedna or eXist database extensions, we would have been forced to renounce to database system's independence, one of our requirements. Our solution was to build a *middleware*, which copes with the update management interoperability and interacts with each database to offer a unified set of methods for access control to our applications. Of course, this supposes a hard overload on the middleware as compared with an equivalent situation with relational databases. This is a drawback of NXD as compared with relational databases. We expect that the stabilization of XUpdate as a W3C Recommendation will suppose its spread and general acceptance by NXD systems. This would eliminate the dependence on proprietary solutions for updates.

Access control is an important issue in a database, for which standard methods are necessary. At the moment of writing this paper, it seems that a standard solution is farther away than it is for updates. However, we are confident that current work on this issue will produce the standard solution needed.

We also found some aspects which could be improved in the future. As with relational databases, users can create their own tables with the names they prefer without risking confusion between tables from two different users (table names avoid ambiguity with a schema name of the form *user.table*), in native XML databases, users should be able to create collections whose names could coincide (which would become something of the style *user.collection*). However, as we worked with eXist, we discovered that such a thing was not possible. The management of collections by users still has to improve to acquire the flexibility it has in relational databases.

The ability of NXD to store schema-less XML documents and their privileged position for implementing complex XQuery queries efficiently (Bonifati and Cuzzocrea, 2007) are important advantages. Standard APIs and query languages are already available. These are positive aspects of NXD, in contrast of access control and document updates, which are the weak aspects of NXD concerning standardization. The situation with updates promises to be standardized soon with the stabilization of the W3C XQuery proposal as a Recommendation. Implementation will probably appear soon after it. As for access control, more work needs to be done. As access control is a basic facility of databases, the effort is necessary if NXD are to play the role they should.

ACKNOWLEDGEMENTS

This work was partially supported by the *Consejera de Educacin de la JCyL* research projects VA010B06, VA012B08, and the project TIN2006-15071-C03-02 from MCyT, Spain (second author). The work of the second author is supported by a fellowship granted by the JCyL and the ESF.

REFERENCES

Boag, S., Chamberlin, D., Fernndez, M. F., Florescu, D., Robie, J., and Simon, J. (2007). XQuery 1.0: An XML Query Language. Technical report, http://www.w3.org/TR/2007/REC-xquery-20070123/.

Bonifati, A. and Cuzzocrea, A. (2007). Synopsis data structures for xml databases: Models, issues, and research perspectives. In *DEXA Workshops, DEXA 2007*, pages 20–24.

Bourret, R. (2007). XML Database Products. http://www.rpbourret.com/xml/XMLAndDatabase-Products.htm. Last upated March 2007. Visited the 2008/10/20.

Chamberlin, D., Florescu, D., Melton, J., Robie, J., and Simon, J. (2008). XQuery Update Facility 1.0. W3C Candidate Recommendation, http://www.w3.org/TR/xquery-update-10-20080801/.

Chaudhri, A. B., Rashid, A., and Zicari, R., editors (2003). *XML Data Management: Native XML and XML-Enabled Database Systems*. Addison-Wesley.

Clark, J. and DeRose, S. (1999). XML Path Language (XPath) version 1.0. W3C Recommendation 16 November 1999, http://www.w3.org/TR/1999/xpath.

Fomichev, A., Grinev, M., and Kuznetsov, S. D. (2006). Sedna: A Native XML DBMS. In *32nd Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2006*, pages 272–281.

Gabillon, A. (2004). An authorization model for XML databases. In *1st ACM Workshop On Secure Web Services, SWS 2004*, pages 16–28.

Garofalakis, M. N., Manolescu, I., Mesiti, M., Mihaila, G. A., Schenkel, R., Thuraisingham, B. M., and Vassalos, V. (2004). What's next in xml and databases? In *Current Trends in Database Technology - EDBT 2004 Workshops*, pages 318–324.

Jagadish, H. V., Al-Khalifa, S., Chapman, A., Lakshmanan, L. V. S., Nierman, A., Paparizos, S., Patel, J. M., Srivastava, D., Wiwatwattana, N., Wu, Y., and Yu, C. (2002). TIMBER: A native XML database. *VLDB J.*, 11(4):274–291.

Kolar, P. and Loupal, P. (2006). Comparison of Native XML Databases and Experimenting with INEX. In *Annual International Workshop on DAtabases, TExts, Specifications and Objects, DATESO6.*

Meier, W. (2002). eXist: An Open Source Native XML Database. In *Web, Web-Services, and Database Systems*, pages 169–183.

Meier, W. (2003). *XML Data Management: Native XML and XML-Enabled Database Systems*, chapter eXist native XML database, pages 43–68. Addison-Wesley.

Pardede, E., Rahayu, J. W., and Taniar, D. (2008). Xml data update management in xml-enabled database. *J. Comput. Syst. Sci.*, 74(2):170–195.

Robie, J. and Chamberlin, D. (2008). XQuery Update Facility 1.0 Requirements. W3C Candidate Recommendation, http://www.w3.org/TR/xquery-update-10-requirements-20080314/.

Robie, J. and Manolescu, I. (2008). XQuery Update Facility 1.0 Use Cases. W3C Candidate Recommendation, http://www.w3.org/TR/xquery-update-10-use-cases-20080314/.

XUpdate Working Group (2000). XUpdate - XML Update Language. http://xmldb-org.sourceforge.net/xupdate/.