Version Management and Historical Queries in Digital Libraries

Fusheng Wang
Siemens Corporate Research
Integrated Data Systems Department
755 College Road East, Princeton, NJ 08540
fusheng.wang@siemens.com

Carlo Zaniolo Xin Zhou Hyun J. Moon
University of California, Los Angeles
Computer Science Department
405 Hilgard Ave, Los Angeles, CA 90095
{zaniolo, xinzhou, hjmoon}@cs.ucla.edu

Abstract

Historical information can be effectively preserved using XML and searched through powerful historical queries written in XQuery. Indeed, by storing the successive versions of a document in an incremental fashion, XML repositories and data warehouses can achieve (i) the efficient preservation of critical information, (ii) its representation using a temporally grouped data model and (iii) the ability of supporting historical queries on the evolution of documents and their contents using XQuery. The proposed approach can be applied uniformly to (a) XML document archives and (b) transaction-time databases published in XML and queried in XQuery. Our case studies include the UCLA course catalog, W3C Xlink standards, and the CIA WorldFact Book, besides relational databases. The experience described here and in [15, 16] suggests that current standards provide reasonable support for temporal applications at the logical level, whereas many challenges remain at the physical level.

1. Introduction

The computer technology that makes digital libraries possible also makes it all too easy to revise documents and publish new versions. Thus version management has become a critical issue characterized by significant new challenges and opportunities [1]. Indeed we are faced with the problem of how to organize, search, and query effectively multi-version documents. On the other hand, we now have a unique opportunity of querying the evolution history of documents and their contents, and learning information that would have been hard to acquire from separate snapshots. Historical queries represent an excellent example of the ability of digital libraries to enhance content delivery services well beyond those of traditional libraries. Our ICAP project proves this point via three interesting case studies on the successive version history of various XML documents, including the UCLA course catalog [3], W3C Xlink [5] standards, and the CIA World FactBook [4].

2. Version Management in XML

Our technical approach is based on the observation that simply storing the successive snapshots of a document is insufficient because of (i) storage inefficiency, (ii) explosion of hits on content-based searches, and (iii) difficulty of answering queries on the evolution of a document and its content. Therefore, we developed XML-based techniques whereby a multiversion document is managed as a unit where its successive versions are represented by the delta changes between versions [12, 6]. This approach optimizes storage utilization, and makes it possible to support complex historical queries on the evolution of the document and its elements (e.g., abstract, figures, bibliography, etc.). For instance in our ICAP project [2], we store the successive versions of the UCLA course catalog as one document, where each element in the document is time-stamped with its period of validity. This representation makes it easy to ask queries such as: "When was course CS143 introduced?" and "How many years did it take for 'nanotechnology' topics to migrate from graduate-level courses to undergraduate ones?" In the spirit of 'e-permanence' standards for Web sites of public interest [9], we can now preserve the whole history of normative documents, city plans, street maps, and natural preserves. Using XML annotation capability we can also annotate each change with an explanation for its cause—in ICAP we annotate each revision of W3C Xlink standards with references to memos that led to the changes. We have automated the process of building the version history of a document (called the V-Document) from its successive versions by (i) first computing the diff between the last version and the new version [12, 6] and (ii) using the results of this process to update the time- stamped V-Document. The V-Document so obtained supports services, such as color-marking the changes between any two versions of the document (not just successive ones) and ad-

vanced features inspired by the Version Machine [7].

3 MultiVersion XML Documents

Our basic approach to represent multi-version XML documents is demonstrated by the ICAP project [2], where we:

(i) use structured diff algorithms [6, 12] to compute the validity periods of the elements in the document,

(ii) use the output generated by the diff algorithm, to represent concisely the history of the documents with a temporally grouped data model. Then, we

(iii) use XQuery, to formulate temporal queries on the evolution of these documents and their contents.

For instance consider a very simple document in three successive versions, as shown in Figure 1.

```
<document>      <!--version 1 -->
  <chapter no="1">
    <title>Introduction</title>
    <section>Background</section>
    <section>Motivation</section>
  </chapter>
</document>
<document>      <!--version 2 -->
  <chapter no="1">
    <title>Overview</title>
    <section>Background</section>
    <section>History</section>
  </chapter>
  <chapter no="2">
    <title>Related Work</title>
    <section>XML Storage</section>
  </chapter>
</document>
<document>      <!--version 3 -->
  <chapter no="1">
    <title>Overview</title>
    <section>Background</section>
    <section>History</section>
  </chapter>
  <chapter no="2">
    <title>Related Work</title>
    <section>XML Indexing</section>
  </chapter>
</document>
```

Figure 1. Snapshot-based representation of XML document versions

To store and query efficiently the history of this evolving document, we compute the differences between its successive versions, using a structure diff algorithm such as those described in [6, 12]. Then, we represent the history of the document by time-stamping and temporally grouping these deltas as shown in Figure 2.

We obtain a temporally grouped representation – V-Document. V-Documents can be easily queried using XQuery.

4. Historical Queries

A unique benefit offered by V-Documents is that they support complex historical queries, using the standard query languages of XML. Indeed, in XML an expressive temporal representation can be obtained by simply adding temporal attributes to the elements [15]. Furthermore, XML's query language, XQuery, achieves native extensibility via user-defined functions. Therefore, temporal extensions can be defined in XQuery, without requiring changes in the standards (which proved unattainable for SQL). In the ICAP project [2], we have defined a temporal library of XQuery functions to facilitate the formulation of historical queries and isolate the user from lower-level details, such as the internal representation of the 'now' timestamp. The complete gamut of historical queries, including snapshot and time-slice queries, element-history queries, 'since' and 'until' queries, can now be expressed in standard XQuery.

This approach can also be applied to transaction-time relational databases, and it requires no change in existing standards. This is significant, given that support for temporal information and historical queries proved to be difficult in standard SQL. Such difficulties led to a number of proposed temporal extensions for SQL [14]; but these have not been incorporated into commercial DBMS. However, historical information can be managed and queried in XML, without requiring extensions to the current standards, since:

- XML provides a richer data model, whose structured hierarchies can be used to support temporally grouped data models by simply adding temporal attributes to the elements. Temporally grouped representations have long been recognized to provide a very natural data model for historical information [10, 11], and
- XML provides query languages with higher expressive power than SQL; in particular, XQuery [8] achieves native extensibility and Turing completeness via user-defined functions [13]. Thus, the constructs needed for temporal queries can be introduced as user-defined libraries, without requiring extensions to existing standards.

5. Testbeds and Demos

Interesting case studies have been implemented and are available for demoing [2], including the UCLA course catalog [3], the W3C Xlink standards [5], and the CIA World FactBook [4]. These studies lead to interesting historical findings that would have been difficult to obtain from the original documents. Also available for demoing there is the ArchIS system [16], which demonstrates that significant performance improvements can be obtained by mapping temporal XML views and queries into relational tables

```
<document vstart="2002-01-01" vend="now">
  <chapter vstart="2002-01-01" vend="now">
    <no isAttr="yes" vstart="2002-01-01" vend="now">1</no>
    <title vstart="2002-01-01" vend="2002-01-01">Introduction</title>
    <title vstart="2002-01-02" vend="now">Overview</title>
    <section vstart="2002-01-01" vend="2002-01-01">Motivation</section>
    <section vstart="2002-01-01" vend="now">Background</section>
    <section vstart="2002-01-02" vend="now">History</section>
  </chapter>
  <chapter vstart="2002-01-02" vend="now">
    <no isAttr="yes" vstart="2002-01-02" vend="now">2</no>
    <title vstart="2002-01-02" vend="now">Related Work</title>
    <section vstart="2002-01-02" vend="2002-01-02">XML Storage</section>
    <section vstart="2002-01-03" vend="now">XML Indexing</section>
  </chapter>
</document>
```

Figure 2. Temporally grouped representation of XML document versions

and SQL/XML. Compression and temporal clustering techniques can also be used to further improve the performance of this approach [16], although much more work is needed in that area. The ArchIS experience also underscored that the performance of native XML database systems might not scale up as well as relational DBMS, and simple techniques are not at hand for achieving effective temporal indexing and clustering in this context.

6. Conclusions

As discussed in a companion paper [15], when dealing with the history of relational databases, SQL:2003 can be used as effectively as XML/XQuery in managing temporal information. This suggests that the approach of carefully grafting temporal extensions standards might no longer be practical or desirable for either SQL or XML. While SQL:2003 can be used for database histories, XML/XQuery can provide a uniform solution to the logical problems involved in publishing and querying the histories of databases and documents. However, achieving scalability and performance for these queries poses major problems, and temporal database researchers should focus on physical issues with renewed energy.

Acknowledgment

The authors would like to thank Vassilis Tsotras for many illuminating discussions.

References

[1] A.R. Kenney, et. al., *"Preservation Risk Management for Web Resources - Virtual Remote Control in Cornell's Project Prism"*, D-Lib Magazine, Jan 2002, 8(1).

[2] The ICAP Project. wis.cs.ucla.edu/projects/icap/.

[3] UCLA Catalog. www.registrar.ucla.edu/catalog/.

[4] CIA: *The World Factbook*. http://www.cia.gov/cia/publications/factbook/.

[5] XML Linking Language (XLink). http://www.w3.org/TR/xlink/.

[6] Microsoft XML Diff. http://apps.gotdotnet.com/ xmltools/xmldiff/.

[7] The Versioning Machine. http://mith2.umd.edu/products/ver-mach/.

[8] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/.

[9] National Archives of Australia: Policy Statement Archiving Web Resources. http://www.naa.gov.au/.

[10] J. Clifford. *"Formal Semantics and Pragmatics for Natural Language Querying"*. Cambridge University Press, 1990.

[11] J. Clifford, A. Croker, F. Grandi,and A. Tuzhilin, *"On Temporal Grouping"*, in Proc. of the Intl. Workshop on Temporal Databases, 1995.

[12] Gregory Cobena, Serge Abiteboul, Amelie Marian, *"Detecting Changes in XML Documents"*, in ICDE 2002.

[13] S. Kepser. *"A Simple Proof for the Turing-Completeness of XSLT and XQuery"*. In *Extreme Markup Languages*, 2004.

[14] G. Ozsoyoglu and R.T. Snodgrass, *"Temporal and Real-time Databases: A Survey"*. in TKDE, 7(4):513–532, 1995.

[15] F. Wang, C. Zaniolo and X. Zhou, *"Temporal XML? SQL Strikes Back!"*, Time 2005.

[16] F. Wang and C. Zaniolo: *"Publishing and Querying the Histories of Archived Relational Databases in XML"*, in WISE 2003.