Managing Multiversion Documents & Historical Databases: a Unified Solution Based on XML

Fusheng Wang
Siemens Corporate Research
fusheng.wang@siemens.com

Carlo Zaniolo Xin Zhou Hyun J. Moon
University of California, Los Angeles
{zaniolo, xinzhou, hjmoon}@cs.ucla.edu

ABSTRACT XML can provide a very effective environment for the preservation of digital information whereby historical information can be easily preserved and searched through powerful historical queries. We propose a unified approach to represent multiversion XML documents and transaction-time databases in XML, and show that temporal queries can then be expressed in standard XQuery. In our demo we demonstrate the benefits of this approach on several examples, including the UCLA course catalog, W3C XLink standards, the CIA WorldFact Book, and a database of company employees. We will also demonstrate the ICAP and ArchIS system that have explored two alternative implementation architectures, one based on native XML DBMS, and the other on mapping the historical XML views back into a relational DBMS.

1. INTRODUCTION

Preservation of digital artifacts represents a critical issue for our web-based society [10, 14]. Web documents are frequently revised, and this creates the problem of how to organize, search, and query effectively multiversion documents. When presented with multiversion documents, users will want to pose queries on the evolution history of the documents and their contents, in addition to searching and retrieving specific versions of such documents. Users would also like to view and query the history of the content of a relational database in a similar fashion.

Our demo will illustrate that the technology is at hand for supporting the preservation and queries of multiversion documents, since XML and its query languages can manage and search effectively the history of relational databases and XML documents. Our approach consists on viewing their history as one integrated document, whereby each element is timestamped with its period of validity, by its `vstart` and `vend` attributes. The advantage of this approach is that it makes it possible to support powerful historical queries using standard XQuery.

Copyright is held by the author/owner.
Eighth International Workshop on the Web and Databases (WebDB 2005), June 16-17, 2005, Baltimore, Maryland.

2. VERSION MANAGEMENT IN XML

Our demo will cover three case studies on evolving XML documents and several examples involving the transaction-time history of relational databases.

A first case study is the UCLA course catalog that is published anew every two years. By integrating successive versions into a time-stamped history, we can express basic snapshot queries such as "Find all the graduate courses offered by the History Department in 1998", and "When was course CS143 introduced?". But we can also pose more complex queries, such as "Show me the growth of the courses offered by the CS department in the last 20 years (a temporal aggregate query)", or "How many years did it take for 'nanotechnology' topics to migrate from graduate-level courses to undergraduate ones?".

A second interesting example involves the successive versions of the W3C XLink standards [4] that are regularly published as new XML documents. Temporal queries bring out changes between the successive versions and also changes on meta-level information. For instance, a query that proved to have unexpected ramifications is "Where authors ever dropped from one version of XLink to the next?".

While XLink and the UCLA course catalog were originally created as XML documents, the CIA World FactBook [3] was instead converted from HTML. This is an interesting publication that describes the economy, government, population, resources, etc., of every country in the world. A new version of the WorldFact Book is published each year; this creates an opportunity to search the political/economic history of the globe, and ask very specific queries on the evolution of particular countries and regions, using XML.

These case studies reveal that an integrated history document can lead to interesting findings and unexpected discoveries that would have been difficult to obtain from the separate versions of the document. These examples will be discussed in the course of our demo.

Historical queries represent an excellent example of the ability of digital libraries to enhance content delivery services well beyond those provided by traditional libraries. Our ICAP system also supports services, such as color-marking the changes between any two versions of the document (not just successive ones) and advanced features inspired by the version machine [9].

3. HISTORICAL QUERIES

An important advantage of our approach is that it can be applied to both XML documents and 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 to SQL [16]; 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 [11, 12], and
- XML provides more powerful query languages, such as XQuery, that achieves native extensibility and Turing completeness via user-defined functions [15]. Thus, the constructs needed for temporal queries can be introduced as user-defined libraries, without requiring extensions to existing standards.

In the ICAP project [1], 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-slicing queries, element-history queries, *since* and *until* queries—can be expressed in standard XQuery [1].

4. MULTIVERSION XML DOCUMENTS

In the ICAP project [1],

(i) we use structured diff algorithms [6, 13] to compute the validity periods of the elements in the document,

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

(iii) we use XQuery, enhanced with the library of temporal functions discussed above, to formulate temporal queries on the evolution of these documents and their contents.

For instance consider a very simple document in three successive versions:

```
<document>                <!--This is version 1 -->
    <chapter no="1">
        <title>Introduction</title>
        <section>Background</section>
        <section>Motivation</section>
    </chapter>
</document>
<document>                <!--This is 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>                <!--This is 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>
```

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, 13]. Then, we represent the history of the document by time-stamping and temporally grouping these deltas as shown below. We call this history-grouped document *V-Document*.

```
<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="now">Background</section>
      <section vstart="2002-01-01"
             vend="2002-01-01">Motivation</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>
```

We obtain a temporally grouped representation (similar to that of SCCS [17]) that can be easily queried using XQuery.

4.1 Complex Queries

Using XQuery [5], complex temporal queries on V-Documents can be easily expressed as described next.

QUERY 1. Evolutionary queries: find the history of titles for Chapter 1:

```
for $title in
doc("V-Document.xml")/document/chapter[no="1"]/title
return $title
```

This query returns the list of the titles of chapter 1, each with the time periods in which those titles were used. Thus for the example at hand it will return:

```
<title vstart="2002-01-01"
      vend="2002-01-01">Introduction</title>
<title vstart="2002-01-02"
      vend="now">Overview</title>
```

The next query shows an example of duration query.

QUERY 2. Duration queries: find titles that didn't change for more than 2 consecutive years.

```
for $title in doc("V-Document.xml")/document/chapter/title
let $dur:=substract-dates($title/@vend, $title/@vstart)
where dayTimeDuration-greater-than($dur, "P730D")
return  $title
```

The next two examples demonstrate how the connectives **since** and **until** from first-order temporal logic can be expressed using XQuery on V-Documents:

QUERY 3. A Since B: find chapters in which section "History" remains unchanged since the version when the title was changed to "Introduction and Overview".

```
for $ch in doc("V-Document.xml") /document/chapter
let $title := $ch/title[.="Introduction and Overview"]
let $sec := $ch/section[.="History"]
where not empty($title) and not empty($sec)
 and $sec/@vstart = $title/@vend and $sec/@vend ="now"
return  $ch
```

QUERY 4. A Until B: find chapters in which the title didn't change until a new section "History" was added.

```
for $ch in doc("V-Document.xml")/document/chapter
let $title := $ch/title[1]
let $sec := $ch/section[.="History"]
where not empty($title) and not empty($sec)
 and  $title/@vend = $sec/@vstart
return  $ch
```

ICAP provides several temporal functions, including a function called *snapshot($node, $versionTS)* that only returns the element and its descendants where *vstart* $\leq$ versionTS $\leq$ *vend*.

QUERY 5. Snapshot queries: retrieve the version of the document on 2002-01-03:

```
for $e in doc("V-Document.xml")/document
return snapshot( $e,"2002-01-03")
```

Other functions written in XQuery hide the user from the implementation details of 'now'. There is also a recursive *diff-identical($node, $version1TS, $version2TS)* function that returns the elements that have changed between $version1TS$ and $version2TS$. This function can, e.g., be used to show and/or color-code the differences between two arbitrary document versions.

QUERY 6. Change queries: retrieve the elements that have changed from 2002-01-01 to 2002-01-03:

```
for $e in doc("V-Document.xml")/document
return alldiff($e,"2002-01-01","2002-01-03")
```

5. ARCHITECTURE & PERFORMANCE

In the ICAP system, history documents are stored and managed using a native XML systems: in our current implementation we use Tamino [7] and X-Hive [8]. In all three case-study examined (i.e, UCLA course catalog, W3C XLink, and the CIA World FactBook), we are dealing with sizes that do not exceed a few megabytes, and can be effectively supported by native XML DBMS. However, the performance of this approach can be unsatisfactory when dealing with the history of database relations. The current contents of typical databases can exceed the gigabyte size—and the size of their transaction time history collected over several years can be significantly larger. These sizes are easily handled by relational database systems whose users have also learned to expect that their DBMS performs well for large data sets. For instance, users of transaction time databases are likely to require that their queries on past snapshots are not much slower than those on their current databases. Our tests [18] indicate that native XML databases do not scale up to this task, and their performance are likely to disappoint the users of relational databases. As described in [18], these performance problems can effectively be addressed by using a relational DBMS to support historical (virtual) XML views and queries on these views. In our ArchIS system, therefore, we decompose the historical views into individual tables that contain the history of each attribute in the relation. Then, we transform and execute the XQuery statements expressed against the views into equivalent SQL/XML statements against the stored tables. This approach assures a satisfactory level of performance [18].

6. TESTBED AND DEMO

Several interesting test cases will be demonstrated in our demo, including the UCLA course catalog, W3C XLink standards [4], the CIA World FactBook [3], and a database of company employees. These examples reveal that V-Documents often lead to interesting findings that cannot be easily inferred from the snapshots of the original documents. Alternative system architectures and their performance [18] will also be covered in the demo.

Acknowledgment

The authors would like to thank Bertram Ludäscher and Richard Marciano for illuminating discussions on the preservation of documents.

7. REFERENCES

[1] The ICAP Project. http://wis.cs.ucla.edu/projects/icap/.
[2] UCLA Catalog. http://www.registrar.ucla.edu/catalog/.
[3] CIA: *The World Factbook.* http://www.cia.gov/cia/publications/factbook/
[4] XML Linking Language (XLink). http://www.w3.org/TR/XLink/.
[5] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/.
[6] Microsoft XML Diff. http://apps.gotdotnet.com/xmltools/xmldiff/.
[7] Software AG: Tamino XML Server, http://www.softwareag.com/tamino.
[8] X-Hive/DB. http://www.x-hive.com.
[9] The Versioning Machine. http://mith2.umd.edu/products/ver-mach/
[10] Library of Congress. Displays for Multiple Versions from MARC 21 and FRBR. http://www.loc.gov/marc/marc-functional-analysis/multiple-versions.html
[11] J. Clifford. *"Formal Semantics and Pragmatics for Natural Language Querying"*. Cambridge University Press, 1990.
[12] J. Clifford, A. Croker, F. Grandi,and A. Tuzhilin, *"On Temporal Grouping"*, in Proc. of the Intl. Workshop on Temporal Databases, 1995.
[13] Gregory Cobena, Serge Abiteboul, Amelie Marian, *"Detecting Changes in XML Documents"*, in ICDE 2002.
[14] 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).
[15] S. Kepser. *"A Simple Proof for the Turing-Completeness of XSLT and XQuery"*. In *Extreme Markup Languages*, 2004.
[16] G. Ozsoyoglu and R.T. Snodgrass, *"Temporal and real-time databases: A survey"*. in TKDE, 7(4):513–532, 1995.
[17] M. J. Rochkind, *"The Source Code Control System"*, IEEE Transactions on Software Engineering, SE-1, 4, Dec. 1975, p. 364-370.
[18] F. Wang, X. Zhou and C. Zaniolo, *"Efficient XML-based Techniques for Archiving Querying and Publishing the History of Relational Databases"*, Submitted for Publication.
[19] F. Wang and C. Zaniolo. *"XBiT: An XML-based Bitemporal Data Model"*, in ER 2004.
[20] F. Wang and C. Zaniolo, *"Publishing and Querying the Histories of Archived Relational Databases in XML"*, in WISE 2003.