

MIX: A Meta-data Indexing System for XML

SungRan Cho
L3S, University of Hannover
scho@l3s.de

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Abstract

We present a system for efficient *meta-data indexed* querying of XML documents. Given the diversity of the information available in XML, it is very useful to annotate XML data with a wide variety of meta-data, such as quality and security assessments. We address the *meta-data indexing* problem of efficiently identifying the XML elements along a location step in an XPath query, that satisfy meta-data range constraints. Our system, named MIX, incorporates query processing on all XPath axes suitably enhanced with *meta-data* features offering not only query answering but also dynamic maintenance of meta-data levels for XML documents.

1 Introduction

Companies are using the Web as the main means of information dissemination, and XML has become the de facto standard for data representation and exchange over the Web. The information available over the Web is extremely diverse, and of varying degrees of quality (e.g., accuracy, recency, etc.) and security. When querying such data, these quality and security assessments are often as useful as the data, and should be queryable as well (see, e.g., [5, 1, 6]). This suggests that it is very useful to annotate XML data with meta-data, and to efficiently identify the data that meet desired constraints on the meta-data.

Our meta-data model for XML documents is based on *meta-data levels*, assigned to elements by associating an attribute called `MetadataLevel`. We use positive integers as meta-data levels, and the one-sided range constraint “ $\leq \text{uml}$ ” to identify desirable data, i.e., those that satisfy the meta-data constraint; `uml` is referred to as the meta-data query threshold. For example, a user can ask a query to find all sales numbers with approximation error $\leq 5\%$.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005

Given the importance of XPath-based query access to XML, and the ubiquity of meta-data that can be modeled as values from an ordered domain, XPath-based query evaluation engines need to be able to efficiently identify the elements that satisfy meta-data constraints along each location step in the XPath query. We have developed a family of indexing structures to facilitate meta-data based access to XML documents. We refer to this family as *meta-data indexes*. A meta-data index can quickly identify all XML elements that are (i) reachable from a specified element using a specified XPath axis, and (ii) satisfy the meta-data range constraint.

Meta-data indexes can be easily realized using popular multi-dimensional index structures such as R-trees [2]. Meta-data indexes incorporate the following features: (i) actual meta-data levels are associated only with elements for which this value is explicitly specified, and (ii) inherited meta-data levels and inheritance source nodes are associated with some non-leaf nodes of the index structure, to enable efficient lookup in the index structure, while keeping the update cost manageable. A meta-data index can be easily utilized by a variety of XPath evaluation algorithms, based on an iterator-style interface, including those in Timber [4] and Galax, an implementation of XQuery [7].

We describe MIX (Meta-data Indexing for XML), a system for meta-data indexed querying, and dynamic maintenance of meta-data for XML documents. MIX implements efficient novel algorithms for querying, focusing on XPath axes, and update of element meta-data levels providing a complete solution for the entire set of XPath query axes.

In Section 2, we introduce the structure of our meta-data indexes. Sections 3 and 4 describe the architecture of MIX and outline the demonstration.

2 Meta-data Indexes

In this section, we describe a family of *meta-data indexes*, which can quickly identify all XML elements reachable from a given element using a specified XPath axis, and have a meta-data level no larger than a specified threshold.

2.1 Matching XPath Location Steps

We use an encoding scheme for nodes in XML documents to reconstruct the XML document unambiguously, which

has the same effect as preorder and postorder. L_n is the rank at which the node is encountered in a *left to right* depth first search (DFS) of the XML data tree, and R_n is the rank at which the node is encountered in a *right to left* DFS. L_n and R_n suffice for matching along ordered and structural axes, such as ancestor, descendant, following, and preceding, but are insufficient for level sensitive matching, such as child and parent axes (matching nodes one level apart), and following-sibling and preceding-sibling axes (matching nodes with the same parent). Associating an additional number with each node, e.g., the parent node's L_n , written as PL_n , permits level sensitive matching as well. A multi-dimensional index structure, such as an R-tree, on the $L_n/R_n/PL_n$ dimensions can be constructed to quickly locate desired result elements [3].

2.2 Incorporating Meta-data Levels

Our meta-data model for XML documents assigns meta-data levels, which are positive integers, to a subset of XML elements. If an element does not have a meta-data level explicitly assigned to it, it inherits the value from its closest ancestor (in the XML tree) that has an explicitly assigned value. We allow the meta-data level of a child element to be larger or smaller than that of its parent element.

To support meta-data indexed querying of XML, the R-tree on the $L_n/R_n/PL_n$ dimensions can be augmented to include the meta-data level. We describe our proposal for such a meta-data index next.

We use the term *bounding box* of a non-leaf index page entry p to denote the set of leaf pages rooted at p . An element node in a bounding box (i.e., in one of its leaf pages) is defined as a *boundary node* of the bounding box if no ancestor element of that node is also in the bounding box.

Actual/Inherited Meta-data Levels: In the *meta-data index* (MI), actual meta-data levels are associated *only* with elements for which this value is explicitly specified (as the value of attribute `ActualML`). Non-leaf page entries in the MI additionally maintain information about meta-data levels that are *inherited* by elements in the (index) subtree of that non-leaf page entry (as the value of attribute `InheritedML`). More precisely, the meta-data levels inherited by the boundary nodes of a bounding box, that do not have an `ActualML` attribute defined, influence the `InheritedML` values of the bounding box. The element nodes in leaf pages entries of the MI *do not* store `InheritedML` values; only the non-leaf page entries maintain ranges of these values.

Inheritance Source Nodes: Although `ActualML` and `InheritedML` in page entries identify which leaf pages contain potentially desirable elements, those values, however, do not suffice for efficient maintenance of the meta-data index under updates to the actual meta-data level of an element. So far, there is no way to efficiently make this decision, based on the `ActualML` and `InheritedML` values. Our solution to enable effective support for maintenance of `InheritedML` values is to store the identities of

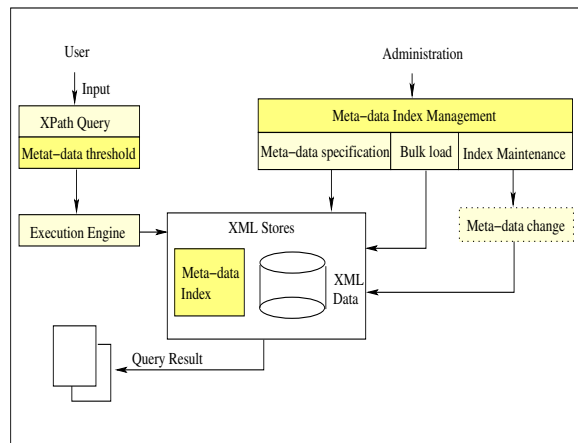


Figure 1: Architecture of MIX

XML element nodes from where these low and high meta-data levels are inherited. This information is maintained in the non-leaf page entries as values for the `InheritList` attribute.

Lastly, we discuss some conventions and an optimization used in the meta-data index. A value of 0 for low/high actual meta-data levels in a bounding box means that no element node in the bounding box has an `ActualML` defined. A value of 0 for low/high inherited meta-data levels in a bounding box means that no node in the bounding box inherits its meta-data level from outside the box. As an optimization to the MI, whenever the value of a meta-data related attribute in a non-leaf page entry is the same as that in its parent non-leaf page entry, the child entry records this information using a special symbol “-”. This optimization is very useful to bound the propagation of updates to `InheritedML` and `InheritList` values, under meta-data level updates.

3 System Architecture

MIX is a Java-based prototype. Its architecture is depicted in Figure 1. It consists of two main components: the *XPath querying* module and the *meta-data index management* module. The first component incorporates a query processor on all XPath axes, using meta-data aware indexing, that accepts queries and returns elements satisfying the meta-data constraint. The second component incorporates maintenance of meta-data information for XML elements and incremental maintenance of the index under meta-data update.

The MIX system in its current version encompasses the following functionality and features that will be demonstrated.

Dynamic visualization: MIX facilitates browsing of different XML documents, display of queries and query results, and visualization of the meta-data index through a flexible and interactive graphical interface.

Aids to adjusting parameters: MIX provides support for adjusting various parameters of the method. The spe-

cific operations taking place when meta-data information or page size on the index is adjusted can be viewed in an interactive mode.

Meta-data access and manipulation: MIX implements novel algorithms that can not only dynamically incorporate meta-data specification changes but also bulk load meta-data efficiently.

4 Demonstration Outline

4.1 XPath Querying

MIX implements the index lookup algorithm for the meta-data index, which uses the `ActualML` and `InheritedML` ranges to return the set of nodes with meta-data level $\leq \text{uml}$ along an XPath location step. Once an XML document to be queried is selected, users can specify XPath queries along with a meta-data threshold through a flexible graphical interface. Then the system uses the meta-data index to return all XML data through the meta-data index. MIX system supports all XPath axes and provides various ways to conveniently visualize query results. In Figure 2(b), for example, a user issues a query, “retrieve all descendants of `all-accounts` with meta-data level ≤ 2 ”, and as a result, `cart` and `item` elements are highlighted in the graphical interface, along with the associated meta-data index information.

4.2 Meta-data Index Management

Meta-data Specification: XML documents can be selected and browsed in graphical form. XML element nodes are labeled with element tags or string values; edges are either between elements or between an element and a string value. In Figure 2(a), an example XML database represents an `online-seller` that contains information about items and accounts. XML elements are encoded with `Ln`, `Rn`, and `PLn` and the `MetadataLevel` attribute specified on elements is displayed in a circle. Meta-data information for XML elements can be dynamically modified on demand through a graphical interface.

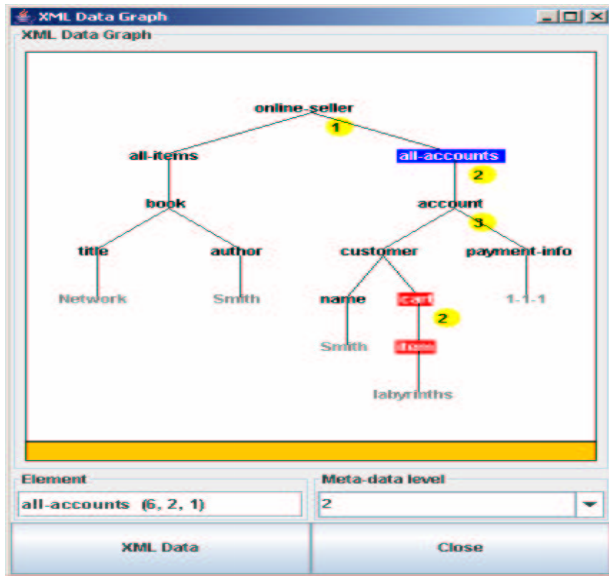
Bulk Loading: The system incorporates efficient support for loading new documents, via a bulk loading algorithm that facilitates index construction and efficient representation of element meta-data information in the hierarchical structure. Each element is represented in the index using the “coordinates”: `Ln`, `Rn`, `PLn`, `ActualML`, `InheritedML`, `InheritList`. Meta-data index contents as well as index operations can be efficiently visualized in the system. Figure 2(b), for example, shows a meta-data index over the `online-seller` dataset of Figure 2(a) with a page capacity of 2. The leaf pages in the index contain both leaf and non-leaf XML elements, and non-leaf index pages indicate page boundaries by the smallest and the largest values occurring in the page.

Index Maintenance: Maintenance under updates is an essential task for the meta-data index, simply because changes occur in the real world. A specified meta-data level

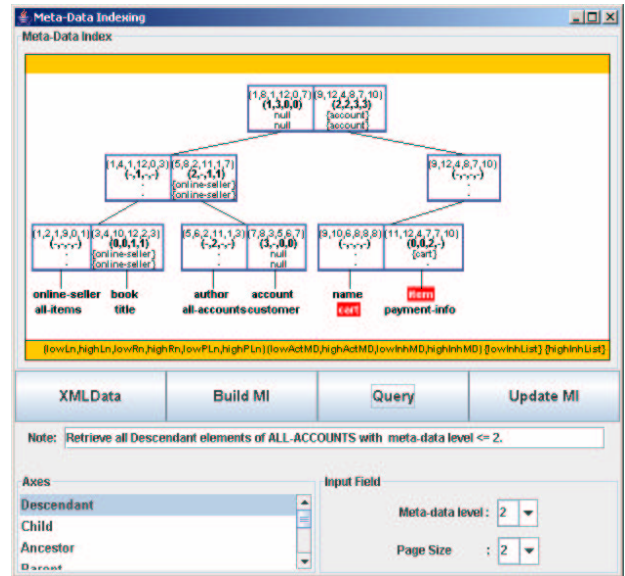
may be changed to a different value, may be removed, or an inherited meta-data level may be replaced by an explicit value. Whenever the meta-data level information for XML elements is modified, such changes should be propagated to the meta-data index, so that they are taken into account during query processing. MIX incorporates novel algorithms for such propagation, in particular such propagation is performed in a top-down and bottom-up fashion in the index.

- **Top-down update for actual meta-data levels:** The top-down update for actual meta-data levels navigates the index tree down from the root page and makes the implicit values (“-”) of `ActualML` explicit during index tree traversal. Once the leaf page is reached, it modifies the `ActualML` of the node, and then invokes bottom-up update.
- **Bottom-up update for actual meta-data levels:** The goal of the bottom-up update is to change back the index page entries correctly. We use a lock-step approach to make sure that the implicit (“-”) values are properly recorded. More specifically, if the value `ActualML` of the child page entry is the same as that of the parent page entry, the value at the child page entry is changed back to “-”. If the value at the child page entry is “-”, then the old value at the parent page is propagated down *only* to child page, instead of propagating all the way down to the leaves.
- **Top-down update for inherited meta-data levels:** The top-down update for inherited meta-data levels first traverses down the index tree to identify pages with the given element node or its descendants, and makes explicit the values of `InheritedML` and `InheritList` during traversal.
- **Bottom-up update for inherited meta-data levels:** The bottom-up update propagates `InheritedML` and `InheritList`, back up the index tree, modifying and making implicit values and lists during traversal. The approach to propagate values and lists to child pages and to revert to “-” is the same as the bottom-up update for actual meta-data levels. The bottom-up update works with a global stack to keep track of how inheritance works for the data items in the leaf page, to deal with a nearest ancestor with an explicit meta-data level.

The system includes support for effective visualization of meta-data level propagation operations in the index. Consider that a meta-data level of 2 is specified for the `all-items` element in Figure 2(a). The top-down update for the actual meta-data level makes the “-” values explicit on the paths where the `all-items` is present. The bottom-up update for actual meta-data level changes back index pages. If a meta-data level for element `all-items` is added, this would result in a change of the meta-data level inherited by its descendants. The top-down update for inherited meta-data level makes the “-” values explicit



(a) XML data



(b) Meta-data Index

Figure 2: XML data and meta-data index

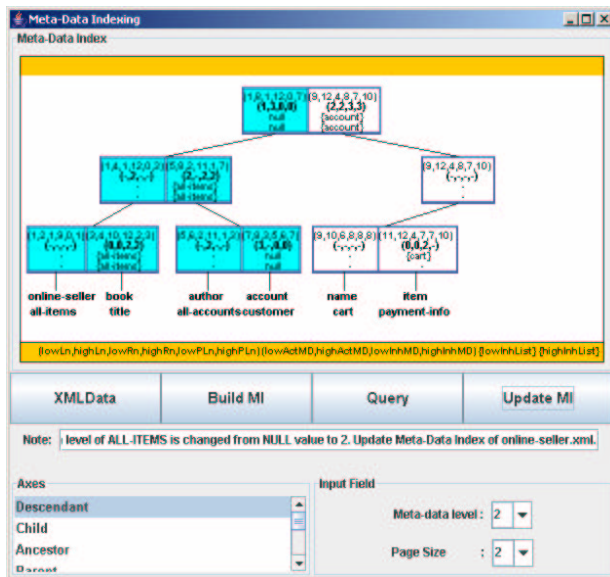


Figure 3: Update of meta-data level

on the paths where book, title, and author elements are present. The bottom-up update for inherited meta-data level changes the smallest or the largest InheritedML value in the index pages in a way identical to what the bottom-up update for actual meta-data level has done. Additionally, the list InheritList is modified to contain a reference to the all-items node. The final meta-data index is shown in Figure 3.

5 Conclusion

The proposed system, MIX, provides meta-data access and manipulation of XML documents. It incorporates novel meta-data indexing methods to facilitate meta-data operations on XML documents as well as novel algorithms to reflect dynamic changes of meta-data level information on the index. Through an easy to use graphical user interface, it accepts queries, requests for element meta-data level modifications, and displays and visualizes query results. Moreover, it facilitates understanding of the novel meta-data index operations incorporating support for dynamic visualization of the index.

References

- [1] D. Bhagwat, L. Chiticariu, W.C. Tan, and G. Vijayvargiya. An annotation management system for relational databases. In *Proc. of VLDB*, 2004.
- [2] V. Gaede and O. Gunther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2), 170–231, 1998.
- [3] T. Grust. Accelerating XPath location steps. In *Proc. of SIGMOD*, 2002.
- [4] H.V. Jagadish, S. Al-Khalifa, A. Chapman, L.V.S. Lakshmanan, A. Nierman, S. Pappas, J.M. Patel, D. Srivastava, N. Wiwatwattana, Y. Wu, and C. Yu. TIMBER: A native XML database, *VLDB Journal* 11(4): 274–291, 2002.
- [5] S. Murthy, D. Maier, and L. Delcambre. Querying bi-level information. In *Proc. of WebDB*, 2004.
- [6] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *Proc. of CIDR*, 2005.
- [7] S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML query language. W3C Working Draft. <http://www.w3.org/TR/xquery/>