ASIC: Algebra-based Structural Index Comparison

Yuqing Wu, Sofía Brenes, Tejas Totade, Shijin Joshua, Dhaval Damani, Michel Salim
Indiana University, Bloomington, USA
{yuqwu, sbrenesb, ttotade, sjoshua, ddamani, msalim}@cs.indiana.edu

ABSTRACT

Structural indices play a significant role in improving the efficiency of XML query evaluation. Being able to compare various structural indexing techniques is critical for a DBMS to select which indices to support, for the query optimizer to choose an index to use in query evaluation, and for DBAs to configure a database application. We present ASIC, an Algebra-based Structural Index Comparison framework that aids users in understanding the ability of different types of structural indices in answering XPath queries which have been characterized using the XPath algebra. ASIC allows users to select, configure and construct structural indices for comparison, guides users to compare the selected indices by evaluating queries of a particular XPath sub-algebra, and visually displays the index structures, query evaluation plans, and performance results for analysis and comparison.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Index

Keywords

Structural index, XPath algebra

1. INTRODUCTION

The structural nature of XML data and the tree-pattern matching features of XML queries require the existence of good structural indices that assist in the efficient evaluation of XML queries, particularly XPath [7] queries. Being able to compare various structural indexing techniques is critical for a DBMS to select the indices to support, for the query optimizer to choose an index to use in query evaluation, and for DBAs to configure a database application. In particular, it is important to determine: (1) for each type of structural index, what are the queries it is suitable for; (2) for each class of queries, what are the structural indices that are suitable to answer them; (3) how do the indices scale with respect to the data size and data distribution; and (4) which index can be updated efficiently when the data, workload and available space change.

Over twenty major structural indices have been proposed, including DataGuides [4], $\mathcal{A}[k]$-index [6], and APEX [2]. However, it is not clear how these indices fare when directly compared to each other. More specifically, there lacks a framework in which structural indices can be compared with respect to the aforementioned key questions.

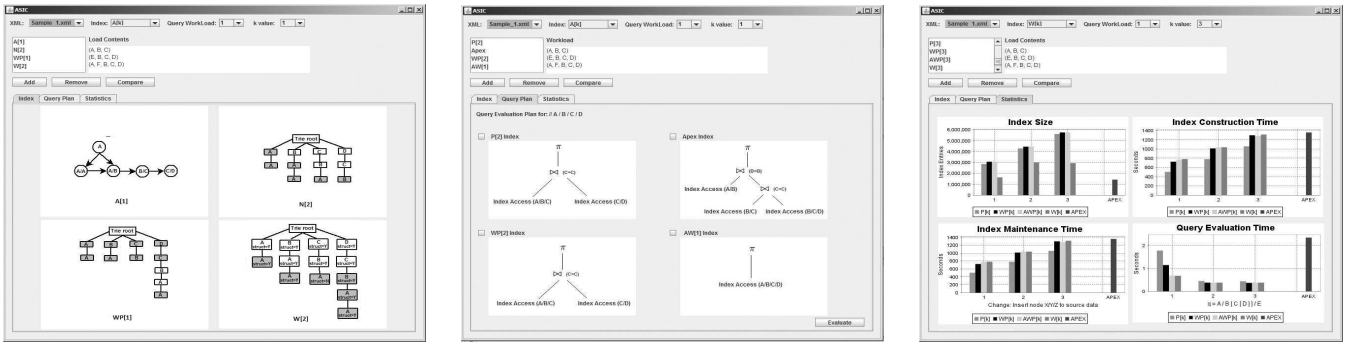
Our recent research [3] on XML query languages and their correlation with data partitions sheds light on this problem. In particular, we studied a set of sub-algebras of XPath and identified the coupling between the partition induced by the structure of an XML document and the partition induced by certain sub-algebras. The results led to the introduction of a family of Trie indices for XML, including the $\mathcal{N}[k]$-Trie and $\mathcal{P}[k]$-Trie indices [1], and the more recent workload-aware Trie indices [8]. This study opens the door for using sub-classes of queries to understand the performance of structural indices for XML. In this demo, we present ASIC - an Algebra-based Structural Index Comparison framework that interprets the organization of the structural indices of an XML document, as well as their ability to answer different types of queries, in terms of XPath algebras.

2. XPATH ALGEBRA AS THE INDEX COMPARISON TOOL

An XML document $\mathcal{X}$ is a node-labeled tree, formally, a 4-tuple (V, Ed, r, λ). We define the *label-path* of nodes m and n as the unique path between m and n, denoted as $LP(m, n)$. We also define $DownPaths(\mathcal{X}, k) = \{LP(m, n) \mid \text{m is an ancestor of } n \text{ and } length(m, n) \leq k\}$. We can define the label-path based equivalence relationship ($\equiv_{\mathcal{N}[k]}$) among data nodes and say that $n \equiv_{\mathcal{N}[k]} m$ iff the downward label-path of length k into nodes n and m is exactly the same. Similarly, we can define the label-path based equivalence relationship ($\equiv_{\mathcal{P}[k]}$) among node pairs. Node (node pair) partitions can be induced by such equivalence relationships, and we call them the $\mathcal{N}[k]$-partition ($\mathcal{P}[k]$-partition). A partition class in the $\mathcal{N}[k]$-partition ($\mathcal{P}[k]$-partition) can be identified by the label-path shared by the nodes (node pairs) in the class.

XPath [7], which is at the core of other more complicated XML query languages, relies on node and path navigation to retrieve results, allowing for path expressions to contain branching conditions. The XPath algebra, as presented

CIKM'09, November 2–6, 2009, Hong Kong, China.



(a) Structural Index Selection (b) Query Evaluation Plan Comparison (c) Performance Comparison

Figure 1: ASIC User Interface

in [3], expresses these navigation expressions as:

$$E := \epsilon \mid \phi \mid \widehat{l} \mid \uparrow \mid \downarrow \mid E_1 \circ E_2 \mid E_1[E_2] \mid E_1 * E_2 \ ^1$$

In [3] we identified a few sub-algebras of the XPath algebra, including the $\mathcal{D}^{[]}$ algebra which consists of the expressions in the XPath algebra without occurrences of the set operators and the $\uparrow$ primitive; and the $\mathcal{D}$ algebra, which further removes the predicate ([ ]) operator from $\mathcal{D}^{[]}$. We also singled out the length-restricted version of the $\mathcal{D}^{[]}$ and $\mathcal{D}$ algebras - the $\mathcal{D}^{[]}[k]$ algebra and the $\mathcal{D}[k]$ algebra. Following the concept of distinguishability of a query language, we say that two pairs of nodes are $\mathcal{D}$ equivalent ($\equiv_{\mathcal{D}}$) if they can not be distinguished by any query in $\mathcal{D}$. Similarly, we can define equivalence relationships $\equiv_{\mathcal{D}^{[]}}$, $\equiv_{\mathcal{D}[k]}$ and $\equiv_{\mathcal{D}^{[]}[k]}$.

Coupling a partition A of an XML document induced by its label-paths and a partition B induced by an XML query language L, we claim in [3] that if A is a refinement of B, any query expression in L can be answered using index-only plans when an index based on A is available. Armed with this coupling theorem, we can analyze structural indices for XML by examining the classes of queries they can evaluate using index-only plans. We summarize the analysis we can perform on a combination of structural indices and XPath sub-algebras in the table below. A ✓ indicates that the index is capable of answering all queries in the query class with an index only plan, and × indicates the lack of such an ability.

	$\mathcal{D}[1]$	$\mathcal{D}[3]$	$\mathcal{D}$	$\mathcal{D}^{[]}[1]$	$\mathcal{D}^{[]}$
DataGuides	✓	✓	✓	×	×
$\mathcal{A}[2]$-index	✓	×	×	×	×
$\mathcal{A}[3]$-index	✓	✓	×	×	×
$\mathcal{P}[1]$-index	✓	✓	✓	✓	✓
$\mathcal{P}[3]$-index	✓	✓	✓	✓	✓

3. DEMONSTRATION PROPOSAL

We will demonstrate the tight coupling of label-path based partitions and XPath algebra based partitions of XML data and how it is reflected on various XML indexing techniques. We will also demonstrate how such data partitions and the organization of the partition classes in the index structures affect the query evaluation plan and the type of queries an index is suitable for. ASIC was implemented on top of the TIMBER [5] native XML database system, focusing on the following functionalities:

Index Structures The index configuration interface allows users to select among all structural indices implemented in ASIC. Users can further configure the indices by providing the degree of local similarity, e.g. the k value for bi-similarity based indices, (i.e. the $\mathcal{A}[k]$-index or the $\mathcal{P}[k]$-Trie index) or a workload to further impact the workload-aware indices (i.e. APEX or the $\mathcal{AW}[k]$-Trie index). The structure of the selected indices will be displayed side by side, as shown in Figure 1(a). Additionally, users can click on the nodes in the index to see their label-paths and extents.

Query Evaluation The core functionality of ASIC is its comparison of structural indices based on their capability in assisting certain types of queries. ASIC classifies queries into algebraic classes to assist the users in understanding the comparison results. When users select a group of indices to compare and a query to be evaluated in these indices, ASIC will generate and display physical evaluation plans for the query and for each selected index configuration, as shown in Figure 1(b). Users can select the plans they want to execute and compare their performance.

Performance Comparison Users can compare the indices based on a rich set of criteria supported by ASIC including index size, construction time, maintenance time, and query evaluation time. They may select the indices they want to compare, the criteria on which they should be compared, and how the figures are plotted. They may also select a comprehensive view of the comparison, in which the indices are compared on all important criteria, as shown in Figure 1(c).

[1] Where $*$ is $\cap$, $\cup$ or $-$

4. REFERENCES

[1] S. Brenes, *et al.* Trie Indexes for Efficient XML Query Evaluation. In *WebDB*, 2008.

[2] C.-W. Chung, *et al.* APEX: An Adaptive Path Index for XML Data. In *SIGMOD*, 2002.

[3] G. H. L. Fletcher, *et al.* A Methodology for Coupling Fragments of XPath with Structural Indexes for XML Documents. In *DBPL*, 2007.

[4] R. Goldman, *et al.* DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *VLDB*, 1997.

[5] H. Jagadish, *et al.* TIMBER: A Native XML Database. *VLDB J.* 11(4): 274-291 (2002).

[6] R. Kaushik, *et al.* Exploiting Local Similarity for Indexing Paths in Graph-Structured Data. In *ICDE*, 2002.

[7] W3C Consortium. XML Path Language (XPath) 2.0. http://www.w3.org/TR/xpath20, 2007.

[8] Y. Wu, *et al.* Workload-aware Trie Indices for XML. In *CIKM*, 2009.