Evaluating Structural Summaries as Access Methods for XML

Mirella M. Moro
UC Riverside
mirella@cs.ucr.edu

Zografoula Vagena
IBM Almaden Research Center
zovagena@us.ibm.com

Vassilis J. Tsotras
UC Riverside
tsotras@cs.ucr.edu

ABSTRACT

Structural summaries are data structures that preserve all structural features of XML documents in a compact form. We investigate the applicability of the most popular summaries as *access methods* within XML query processing. In this context, issues like space and false positives introduced by the summaries need to be examined. Our evaluation reveals that the additional space required by the more precise structures is usually small and justified by the considerable performance gains that they achieve.

Categories and Subject Descriptors: H.2.3 [**Database Management**]: Languages

General Terms: Query Processing, Performance

Keywords: Structural Summaries, Precision

1. INTRODUCTION

The lack of schema on XML documents has originally motivated the research for *Structural Summaries* [2, 5]. Such summaries are compact, dynamically maintained graph structures that preserve all structural characteristics of XML documents. Their effectiveness as path indexes has already been widely accepted. Moreover, these structures are also finding their way in the web environment, as enhancements to traditional full-text indexes, enabling querying and retrieval of XML documents at different granularities [8].

The versatility of structural summaries has led to the proposal of many variations. Current work has mainly explored such structures as secondary indexes that can identify document nodes reachable from specific path patterns. Recently, summaries have also been explored in a different way, namely as *access methods* (i.e. on-disk data organizations, coupled with specialized retrieval operations) [6]. Nevertheless, their relative behavior within this context has not yet been fully understood.

In this work, we attempt to fill this gap and provide the first empirical study for the applicability of those summaries as alternative access methods within XML query processing. We experimentally compare their behavior while processing *path expression queries*. Answers to such queries are document path instances which satisfy all the constraints imposed by the associated path expressions. Our main contributions are summarized as follows: (1) We explore the behavior of the summaries as access methods for XML query processing. (2) Within this context, we identify the cases where the summaries provide false positives and show how such results impact the size of intermediate results as well as the additional processing time required to retrieve exact answers. (3) We complete our study by exploring the behavior of the summaries for heterogeneous XML document collections. To the best of our knowledge, this is the first study that considers this situation, which is very common, especially in an environment such as the world wide web.

We proceed with section 2 that describes the structural summaries under comparison and the processing of the considered queries. Section 3 summarizes the most important results of our study, and section 4 concludes the paper.

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-332-9/06/0005.

2. STRUCTURAL SUMMARIES

A structural summary groups document nodes into classes based on their structural characteristics. Each node in the summary represents a group of document nodes that belong to the same *equivalence class*. This group of nodes associated with a summary (index) node is called its *extent*. Next, we provide a brief description of the summaries we compare.

DataGuides. Starting from a database D, a `DataGuide` keeps a unique copy of *all* and *only* the paths that exist within D [2]. Among the different `DataGuides` that can be generated from a database D, in our experiments, we employed the `Strong DataGuide`, in which nodes with the same path from the root are grouped together.

A(k)-indexes. The `A(k)-indexes` [3] correspond to a family of approximate structural summaries that use bisimilarity to identify the groups of equivalent data nodes. Each index node represents only bisimilar elements from the document and the index extents do not overlap. In our experiments, we considered `A(`k`)` for $k = 1$, 2 and 3.

Suffix Trees. A `Suffix Tree` [7, 4] is a trie-based data structure that stores all set of strings S, and all suffixes of each string s in S. The structure has already been employed in several works for XML query processing. Another trie-based data structure, namely the `prefix tree`, has also been employed. We omit the `prefix tree` from our discussion since its performance as access method is similar to that of the suffix tree.

Query Processing. With any of the previous summaries, processing of path expression queries is performed in *two* phases. In the *first* phase, the structural summary is probed to identify the index nodes (and respective extents) that satisfy the query. Since all query nodes (as opposed to only those nodes reachable by a path) are involved in the results, a post-processing matching step is needed over the extents identified in the first phase. Thus, a *second* phase filters out any false positives and produces the actual results.

False Positives. While computing the query result, all extents that satisfy the query are considered. In this context, there are three main situations where summaries may give false positives. First, the summary may not cover the query. This may happen when the summary is compacted to fit in memory. Second, when a query includes an ancestor (parent) whose descendant (child) is an optional subelement. The summary then returns a superset of the result, where many ancestor (parent) elements do not have matching descendants (children). Third, the original dataset may present recursive paths, aggravating the previous situation.

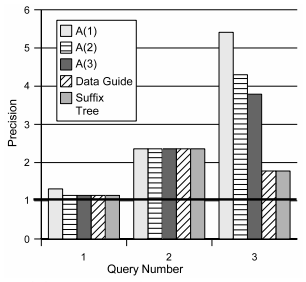
3. EXPERIMENTAL EVALUATION

In our evaluation of the summaries as access methods, we considered the following aspects that characterize the structures: space requirements, precision (impact of false positives), and time to evaluate a query.

Space Requirements. We considered heterogeneous datasets and evaluated the impact of diversity on the size of the summaries. Each dataset comprises of 1000 XML documents generated from different DTDs. The dataset heterogeneity is defined by varying two parameters: the number of DTDs from which the documents are generated, and the rate in which one of the DTDs appears in the document.

	Size	Size Structural Summary in %				
Dataset	Dataset	A(1)	A(2)	A(3)	DG	Suffix
5 dtds	1253332	0.014	0.019	0.022	0.027	0.115
10 dtds	753118	0.029	0.039	0.044	0.054	0.221
20 dtds	579183	0.067	0.088	0.098	0.156	0.549
10%	733042	0.061	0.077	0.086	0.154	0.515
50%	1236341	0.036	0.046	0.051	0.088	0.298
80%	1632115	0.027	0.035	0.039	0.061	0.215
100%	1819657	0.002	0.003	0.004	0.004	0.013

(a) Size for heterogeneous datasets



(b) Precision for Nasa and XMark

Figure 1: Experimental results

Figure 1(a) depicts the size of each summary as a percentage of the dataset size. In the top half, the number of DTDs varies from 5 to 20. In the bottom half, the documents are created from 25 DTDs, and one single DTD is in all datasets on different rates: 10% means that ten percent of the data was generated from that specific DTD, the remaining 90% from other DTDs. This number is gradually increased to 100%, meaning that the whole dataset was generated from one DTD. A clear observation is that the summaries save substantial space. Even with heterogeneous data, the summary sizes are still smaller than 1% of the document. Likewise, the more homogeneous the dataset, the smaller the resulting summaries. We also considered other real and benchmark datasets (DBLP, NASA, Reed University, Shakespeare's plays, Swiss-Prot, TreeBank, and Xmark) and obtained similar results.

Precision of Structural Summaries. Here, we focus on the *first* phase of the query processing, where the extents of all nodes required by the query are evaluated. We considered path queries with 3 to 10 nodes, and with both parent-child and ancestor-descendant axes. The performance measure is the ratio of the total number of document nodes that are returned by the structural summary (through the returned extents) and the number of nodes that compose the actual results. We call this ratio the *precision* of the structural summary. Precision ratio equal to 1 means the summary returns exactly the query answer, whereas higher values imply larger number of false positives. This measure is justified because the work necessary in the post-processing step is proportional to the size of the returned extents.

We performed experiments over heterogeneous, real and benchmark data. Figure 1(b) shows results for three typical queries that were performed on real dataset NASA and benchmark dataset XMark (results for other queries follow similar patterns). In queries 1 and 2, all summaries had similar relative performance because the indexes present similar features. Query 2 requires nodes that are more selective than those in query 1 (the second case for false positives) thus the precision ratio practically doubles. Query 3 shows how precision deteriorates when the value of k (in A(k) indexes) is still not stabilized. These experiments show that more precise structures (like Dataguides) consistently give better results. These three queries illustrate the most common scenarios in our experiments, but there were also much higher reported precision ratios (worst cases around 100).

Query Evaluation Performance. The first phase identifies the query extents by traversing the summaries. Since the size of the summaries is minimal, the time for processing it is minimal as well. Here, we evaluate the second phase of the query processing by employing the summaries as access method to a state of the art algorithm for pattern query processing, the `TwigStack` [1]. We compared the summaries performance against two benchmarks, (i) when evaluating the query over the whole document (upper bound) and (ii) when the actual query results are present (a hypothetical lower bound, equivalent to precision ratio 1). As a typical example, in query 3, the results were: 345ms for the whole document, 279ms for `A(1)`, 108ms for Dataguide, and 80ms for actual results. As expected, the overall query time of the summaries is proportional to the size of the extents, in which the false positives make a considerable influence. Nevertheless, summaries with good precision (i.e., Dataguides) gave query time performance very close to the optimal.

4. CONCLUSION

Current work on structural summaries covers the problem of identifying nodes reachable from specific path patterns. We focus on a broader perspective, where not only the reachable final nodes are important, but also intermediate path nodes are needed to construct the result of a query. Our results show that, for tree-structured XML data, performance improves when using more precise structures, like `DataGuides` (or `1-indexes`) instead of approximate versions.

5. REFERENCES

[1] N. Bruno et.al. Holistic Twig Joins: Optimal XML Pattern Matching. In *SIGMOD*, 2002.

[2] R. Goldman and J. Widom. DataGuides: Enabling Formulation and Optimization in Semistructured Databases. In *VLDB*, 1997.

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

[4] E. M. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *Journal of ACM*, 23(2), April 1976.

[5] T. Milo and D. Suciu. Index Structures for Path Expressions. In *ICDT*, 1999.

[6] M. M. Moro, Z. Vagena, and V. Tsotras. Tree-Pattern Queries on a Lightweight XML Processor. In *VLDB*, 2005.

[7] P. Weiner. Linear Pattern Matching Algorithms. In *Annual Symp. on Switching and Automata Theory*, 1973.

[8] B. Yang et.al. Virtual cursors for XML joins. In *CIKM*, 2005.