Synopsis Data Structures for XML Databases: Models, Issues, and Research Perspectives

Angela Bonifati
Icar CNR, Italian National Research Council
Via P. Bucci 41C, I-87036 Rende, Italy
bonifati@icar.cnr.it

Alfredo Cuzzocrea
Icar CNR, Italian National Research Council &
DEIS, University of Calabria
Via P. Bucci 41C, I-87036 Rende, Italy
cuzzocrea@si.deis.unical.it

Abstract

Due to the lack of efficient native XML database management systems, XML data manipulation and query evaluation may be resource-consuming, and represent a bottleneck for several computationally intensive applications. To overcome the above limitations, a possible solution consists in computing synopsis data structures from XML databases, i.e. compressed representations providing a "succinct" description of the original databases while ensuring low computational overhead and high accuracy for many XML processing tasks. Specifically, these data structures are very useful for both selectivity estimation and approximate query answering purposes. On the other hand, while synopsis data structures have been widely applied to relational as well as multidimensional data, a full usage for XML data is still lacking. Inspired by these considerations, in this paper we discuss the models and issues of synopsis data structures for XML databases, and we complete our analysis by selecting and discussing future perspectives for this research field.

1. Introduction

XML processing is gaining momentum as it affects the reliability and performance of many computationally intensive applications, ranging from the recent Web-based and Grid-based infrastructures to the more traditional integrated and cooperative information systems. In all such scenarios, efficiently querying native XML databases[1] is a critical issue, due to the evidence that non-native databases may be too inefficient. This is due to several motivations, among which we recall: (*i*) the XML data model is hierarchical in nature and not prone to be represented as a set of relations or objects; (*ii*) quite often, XML documents appear in a *schema-less* fashion (e.g., like those of corporate B2B and B2C *e*-commerce Web systems), thus making the relational translation more difficult; (*iii*) the inherent "richness" of the standard XML query language, which defines a comprehensive class of queries with possibly complex syntax and predicates (e.g., `for` clause of *XQuery* queries [6], *twig* XML queries [7], partial- and exact-match *XPath* queries [8] etc.); (*iv*) the ambiguity of the XML semantics during query evaluation [20]; (*v*) problematic update management issues posed by processing XML data [2, 3, 31].

A possible solution to the above issues consists in computing and packaging *synopsis data structures* against the original XML documents. The aim of this approach is to obtain a small-size XML document $\widetilde{X}$ (i.e., a proper synopsis data structure) starting from the input XML document X, in order to reduce the computational overhead due to processing data in X. The intuition behind such data structures is that the information content carried out by $\widetilde{X}$ is "very similar" to the information content carried out by X. The measure of this similarity can be defined according to different alternatives depending on the target application. As an example, looking at the structure of documents (e.g., [19]) is useful in the context of algorithms for clustering XML documents, and algorithms for detecting similarities among XML documents. In our approach, we focus on XML query processing aspects, thus we define, given a query Q, the similarity measure in terms of the *approximation error* of Q, denoted by $E(Q)$. $E(Q)$ is quantified as the relative difference between Q *exact answer* (i.e., the answer to Q evaluated against X), denoted by $A(Q)$, and Q *approximate answer* (i.e., the answer to Q evaluated against $\widetilde{X}$), denoted by $\widetilde{A}(Q)$. $E(Q)$ is thus defined as follows: $E(Q) = \frac{|A(Q)-\widetilde{A}(Q)|}{A(Q)}$. The goal of any query-centric technique for computing synopsis data structures from XML documents is that of making $E(Q)$ as minimal as possible.

[1] In the rest of the paper, we mutually refer to XML documents as the components of a native XML database, i.e. a database that stores such components natively.

1529-4188/07 $25.00 © 2007 IEEE
DOI 10.1109/DEXA.2007.100

Lossless compression techniques ensures that, for any arbitrary query Q, $E(Q) = 0$, whereas lossy compression techniques do not guarantee this condition.

Synopsis data structures allow to achieve two main goals. The first goal is to enable *selectivity estimation* for XML queries (e.g., [1]). In our context, given an XML query Q, the selectivity of Q, denoted by $\gamma(Q)$, is intended as the number of XML elements/nodes visited during the evaluation of Q. Similarly to conventional relational database systems [16], the *Selectivity Estimator* is a fundamental component of the *Query Optimizer*, which, based on the selectivity of queries, determines the optimal query plan according to which the input query is evaluated. In turn, the latter plan represents the input of the *XML Query Engine*, which eventually provides the answer to the query. The second goal of synopsis data structures consists in efficiently supporting *approximate query answering* over XML documents [25], with is similar in spirit to what done in the context of databases and data cubes [11]. This means to evaluate input queries against the synopsis data structure $\widetilde{X}$ rather than on the original XML document X, by allowing an approximation error that is meant to be negligible for the target application. Such an approximation, although generally applicable, makes more sense in particular application fields such as XML-OLAP[15], due to the fact that imprecise query evaluations are perfectly tolerable in OLAP systems[9].

Starting from these considerations, in this paper we discuss the models, issues, and research perspectives that concern the problem of efficiently computing synopsis data structures over XML databases, with particular emphasis on query processing aspects. The paper is organized as follows. In Section 2, we provide an overview of the state-of-the-art techniques for computing synopsis data structures for XML databases. In Section 3, we discuss the research challenges and novel perspectives of our investigation on this research field. Finally, in Section 4, we present the concluding remarks.

2. Synopsis Data Structures for XML Databases: An Overview

Data compression/reduction techniques for massive data sets have a long history. Among proposals appearing in literature, *sampling* [13], *histograms* [14] and *wavelets* [5] are the widely-accepted methods for obtaining synopsis data structures from massive amount of data sets ranging from conventional relational databases to data warehouses and OLAP data cubes.

In the context of XML, there has been some preliminary work related to the problem of compressing large XML documents, and efficiently using such compressed representations to improve the performance of mining and query algorithms over XML data. Also, previous solutions for computing synopses from relational data sources are ineffective for XML data sets, as also recognized in [25]. A first study on the problem of optimizing XML queries via computing so-called *data guides* having a statistical nature appears in [18]. Data guides can be viewed as a form of synopsis data structures. This solution has limited applicability, as it only supports approximate query answering for path queries of fixed length, which makes it not extensible in such a way to cover more complex kinds of XML queries.

[1] investigates the problem of estimating the selectivity of path expression in an Internet-like applications scenario. The solution consists in two specialized data structures that, in total, form a synopsis data structure: (*i*) the *Summarized Path Tree* that, given the input XML document X, collapses its structure in a tree where low-frequency nodes are deleted or coalesced in summarizing nodes; (*ii*) the *Summarized Path Table* that stores the count of all frequent paths in X having length less than a parameter $k > 0$. Similarly to [18], this proposal has limited applicability and, in addition to this, it does not consider at all the problem of maintaining compressed information related to the structure of the input XML document, which should not be instead ignored.

[22, 23] move the problem from tree-structured XML databases to the more general case of handling *graph-structured XML databases* like those one can find in large-scale, complex Web systems, modeled by means of `IDREF` constructs. The solution in [22, 23] consists in computing a statistical summary of the input graph-structured XML database X within a given space bound, thus devising a *graph-synopsis* called XSKETCH, which, starting from a *partitioned representation* of nodes in X, is obtained by means of successive refinements of the so-called *label-split graph*, the coarsest summary of X, which summarizes the path and branching distribution in X. To efficiently support the construction of XSKETCH, what is an $\mathcal{NP}$-hard problem as recognized in [22, 23], an heuristic algorithm based on greedy selection is proposed. Specifically, to this end, authors introduce the definitions of *backward-stability* (B-stability) and *forward-stability* (F-stability), which both model two meaningful *equivalence classes* for nodes (see [22, 23] for further details). The XSKETCH idea is further exploited in [26] in order to achieve the definition of the `twig-`XSKETCH synopsis data structure, which is tailored to provide selectivity estimation and approximate query answering to complex twig XML queries.

StatiX [10] is a framework for computing statistical summaries from *XMLSchema*-aware XML documents with the purpose of efficiently supporting the design of cost-based XML storage layers of native and relational-based XML databases. The main idea of *StatiX* is that of decomposing input schemas in order to obtain statistics at different granularity, and then gathering these statistics when input

documents are validated against reference schemas. Also, this method is able to find *structural skews* that can appear in an XML document, which is an interesting aspect quite neglected in literature.

Contrarily to [10], [30] does not take into account schema information of XML documents. [30] proposes estimating the size of answers to XML queries by means of construction the so-called *position histograms*, in a similar spirit to what done with relational and multidimensional databases, where a wide histogram literature exists. Position histograms are able to capture ancestor-descendant relationships among nodes of the input document, and encode occurrences of positions of the corresponding XML elements. Position histograms provide an efficient support for evaluating both simple and complex path queries, but they are not able to capture skews that can be originated by other important structural constraints (e.g., union, repetition) present in typical XML documents.

XPATHLEARNER [17] is an on-line learning method for estimating the selectivity of XML path expressions by means of statistical summaries used to build a *Markov histogram* on path selectivities gathered from the target XML engine. The novelty of XPATHLEARNER is represented by the fact that it collects the required statistics from the *query feedback* in an online manner, without accessing and scanning the original XML data, which is in general resource-consuming. These statistics are used to learn both tag and value distributions of input queries, and, when needed, to change the actual configuration of the underlying Markov histogram in order to improve the accuracy of approximate answers. In this vest, XPATHLEARNER can be intended as a *workload-aware method*. As stated in [17], XPATHLEARNER is able to provide support for the accurate estimation of the selectivity of most popular XML path expressions.

[4] has studied the problem of finding a compact tree representation for large XML documents, which allows their evaluation in main memory. It uses techniques derived from symbolic model checking to compress and manipulate the tree structure. Specifically, succinct representations of document tree structures based on sharing subtrees are highly effective. The ability of handling large portions of the structure of very large XML documents in main memory is crucial to the development of efficient, scalable native XML databases and query engines. However, the compact tree representation of [4] is different from a synopsis, as it is an exact, albeit succinct, representation of an XML document. Thus, an exact query answer is guaranteed instead of a partial or approximate query answer.

TREESKETCH [25] is a complex synopsis data structure for graph-structured XML databases which extends the capabilities of both the XSKETCH [22, 23] and `twig-XSketch` [26] proposals. TREESKETCH still founds on a partitioned representation of nodes of the input graph-structured XML database X, and, with respect to previous results [22, 23, 26], introduces the amenity of capturing the *similarity of sub-structures* that one can find in X. To this end, authors introduce the novel concept of *count-stability* (C-stability), which is a meaningful refinement of the previous F-stability concept [22, 23]. In more detail, C-stability defines a further partition of equivalence classes given by the F-stability (see [25] for further details), thus achieving a better performance in the compression of the input graph-structured XML database as well as a better support for a larger set of complex XML queries.

Finally, XCLUSTER [24] is a recent proposal that aims at computing a synopsis $\widetilde{X}$ from a given XML document X, via summarizing both the structure and the content of X. Specifically, the XCLUSTER-based synopsis data structure is a node- and edge-labeled graph, where each node represents a sub-set of elements with the same tag, and an edge connects two nodes if an element of the source node is the parent of elements of the target node. Nodes and edges of this graph are then equipped with aggregate statistical information concerning: (*i*) for nodes, tuples $\langle Tag, \texttt{COUNT}, Val \rangle$, such that Tag is the tag of elements for which the statistics is computed, `COUNT` is the count of elements having Tag as tag, and Val is an optional summary value that captures the distribution of elements having Tag as tag in that statistics; (*ii*) for edges, the average child count between source and target elements, which allows us to model the connectivity between elements and sub-elements. XCLUSTER is also the underlying framework of the system AQAX (*Approximate Query Answering for XML*) [27], which enables rapid, interactive exploration of large XML databases.

3. Novel Research Perspectives

Several open issues are still left that concern the problem of fully using synopsis data structures in native and non-native XML databases. In this Section, we outline them by highlighting the future directions of this research field.

We are in the process of investigating some of these issues and we provide an intuition of what will be the results of our investigation.

Probabilistic guarantees of synopses A first topic that is interesting and not yet considered is the possibility of obtaining *probabilistic guarantees of XML synopses*. Whereas the optimal condition $E(Q) = 0$ is seldom reachable in practice, the goal is to keep this error as lower as possible. In principle, we can think of providing a probabilistic bound for such value. For instance, in many applications it is difficult to exactly quantify $E(Q)$, due to the fact that the exact query answer $A(Q)$ may not be available to external users/applications. By opposite, when dealing with proba-

bilistic data,i.e. data that is not certain, query processing on this data carries out a probability value [29] for both exact and approximate query answers. In such cases, $A(Q)$, albeit available, is inherently probabilistic. These are only a few examples of cases in which we need to compute a *probabilistic bound* to $E(Q)$. Similar work has been done in the context of data cube compression via wavelets [12]. We are studying these issues when considering XML databases.

Full-fledged XML databases and synopses As a further issue, the usage of synopsis data structures in both native and non-native XML databases has not been investigated yet. Indeed, these databases are still in their infancy, whereas the relational technology and data warehousing is mature enough to integrate such data structures in their query engines. We feel that the *integration of such data structures in XML databases* and their usage for query processing is not trivial. In native XML databases, when storage is not based on an existing technology, storing and using such structures is challenging. In non-native XML databases, these structures may be stored as their relational counterpart, which is not always what we want to use when evaluating an XML query. Thus, translation mechanisms for such synopsis data structures are to be devised in order to enable their transparent use.

Clustering techniques for synopses Computation of synopses has been based on statistical measures on the XML documents, whereas such data structures could be built with alternative methods. For instance, statistics on an XML document may not be available or difficult to compute. In such cases, we can think of applying *clustering algorithms* to an XML document and deriving the synopsis thereof. We need efficient techniques to analyze the output of a clustering algorithm and translate this output into a synopsis.

Workload-driven synopses Moreover, the presence of *query workloads* can simplify the job of an XML query engine using such data structures. We have observed that[17] has studied how to exploit such workloads for estimating the selectivity of XPath expressions. Whereas this is a first attempt in this direction, we argue that more work can be done to take full advantage of query workloads for XML query processing, i.e. with queries expressed in XQuery and when structural and value joins are to be evaluated.

Application scenarios Many applications may benefit from such data structures. As an example, we discuss here two of them: *XML data streams* and *P2P data management*.

Data stream processing brings up other issues, e.g. the fact that the data stream cannot be buffered and stored and needs to be processed on the fly. Streaming processing for XML has been considered in [28, 21]. XML stream processing takes advantage of special-purpose indexes, that are different from indexes on stored data. We think that suitable synopsis data structures can be built for XML streaming data, and these are different from those used so far. Such data structures have to be lightweight and easily adaptable to the changes in the stream.

Another application that we envision for synopses are P2P XML databases. In such databases, peers are dynamic and there is no assumption of a global schema. Thus, a global synopsis data structure is unfeasible. A problem that arises in such context is how to use the synopsis for distributed XML query processing.

4. Conclusions

In this paper, we have discussed the current models and issues for XML synopsis data structures, and given perspectives for future work. We have highlighted that a lot of work needs to be done in this direction, and have identified a list of open research problems.

References

[1] A. Aboulnaga, A.R. Alameldeen, and J.F. Naughton, "Estimating the Selectivity of XML Path Expressions for Internet Scale Applications", *Proc. of VLDB*, 2001.

[2] M. Benedikt, A. Bonifati, S. Flesca and A. Vyas, "Verification of Tree Updates for Optimization" *Proc. of CAV*, 2005.

[3] P. A. Boncz, J. Flokstra, T. Grust, M. van Keulen, S. Manegold, K. S. Mullender, J. Rittinger and J. Teubner, "MonetDB/XQuery-Consistent and Efficient Updates on the Pre/Post Plane", *Proc. of EDBT*, 2006.

[4] P. Buneman, M. Grohe and C. Koch, "Path Queries on Compressed XML", *Proc. of VLDB*, 2003.

[5] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim, "Approximate Query Processing Using Wavelets", *Proc. of VLDB*, 2000.

[6] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu, "XQuery 1.0: An XML Query Language", *W3C Working Draft 07*, available at `www.w3.org/TR/xquery/`, 2001.

[7] Z. Chen, H.V. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. Ng, and D. Srivastava, "Counting Twig Matches in a Tree", *Proc. of IEEE ICDE*, 2001.

[8] J. Clark, and S. DeRose, "XML Path Language (XPath), Version 1.0", *W3C Recommendation*, available at `http://www.w3.org/TR/xpath`, 1999.

[9] A. Cuzzocrea, "Overcoming Limitations of Approximate Query Answering in OLAP", *Proc. of IEEE IDEAS*, 2005.

[10] J. Freire, J.R. Haritsa, M. Ramanath, P. Roy, and J. Simeon, "StatiX: Making XML Count", *Proc. of ACM SIGMOD*, 2002.

[11] M. Garofalakis, and P.B. Gibbons, "Approximate Query Processing: Taming the TeraBytes", *Proc. of VLDB*, 2001.

[12] M.N. Garofalakis, and P.B. Gibbons, "Wavelet Synopses with Error +Guarantees", *Proc. of ACM SIGMOD*, 2002.

[13] P.B. Gibbons, and Y. Matias, "New Sampling-Based Summary Statistics for Improving Approximate Query Answers", *Proc. of ACM SIGMOD*, 1998.

[14] Y. Ioannidis, "The History of Histograms (abridged)", *Proc. of VLDB*, 2003.

[15] M.R. Jensen, T.H. Moller, and T.B. Pedersen, "Specifying OLAP Cubes On XML Data", *Proc. of IEEE SSDBM*, 2001.

[16] R.P. Kooi, *The Optimization of Queries in Relational Databases*, PhD Thesis, CWR University, 1980.

[17] L. Lim, M. Wang, S. Padmanabhan, J.S. Vitter, and R. Parr, "XPathLearner: An On-Line Self-Tuning Markov Histogram for XML Path Selectivity Estimation", *Proc. of VLDB*, 2002.

[18] J. McHugh, and J. Widom, "Query Optimization for XML", *Proc. of VLDB*, 1999.

[19] A. Nierman, and H.V. Jagadish, "Evaluating Structural Similarity in XML Documents", *Proc. of WebDB*, 2002.

[20] M. Gyssens, J. Paredaens, D. Van Gucht and G.H. L. Fletcher: "Structural characterizations of the semantics of XPath as navigation tool on a document", *Proc. of PODS*, 2006.

[21] F. Peng and S. S. Chawathe. "XPath queries on streaming data", *Proc. of ACM SIGMOD*, 2003.

[22] N. Polyzotis, and M. Garofalakis, "Statistical Synopses for Graph Structured XML Databases", *Proc. of ACM SIGMOD*, 2002.

[23] N. Polyzotis, and M. Garofalakis, "Structure and Value Synopses for XML Data Graphs", *Proc. of VLDB*, 2002.

[24] N. Polyzotis, and M. Garofalakis, "XCluster Synopses for Structured XML Content", *Proc. of IEEE ICDE*, 2006.

[25] N. Polyzotis, M. Garofalakis, and Y. Ioannidis, "Approximate XML Query Answers", *Proc. of ACM SIGMOD*, 2004.

[26] N. Polyzotis, M. Garofalakis, and Y. Ioannidis, "Selectivity Estimation for XML Twigs", *Proc. of IEEE ICDE*, 2004.

[27] J. Spiegel, E. Pontikakis, S. Budalakoti, and N. Polyzotis, "AQAX: A System for Approximate XML Query Answers", *Proc. of VLDB*, 2006.

[28] T. J. Green, G. Miklau, M. Onizuka and D. Suciu, "Processing XML Streams with Deterministic Automata", *Proc. of ICDT*, 2003.

[29] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases", *Proc. of VLDB*, 2004.

[30] Y. Wu, J.M. Patel, and H.V. Jagadish, "Estimating Answer Sizes for XML Queries", *Proc. of EDBT*, 2002.

[31] J. Xu Yu, D. Luo, X. Meng and H. Lu, "Dynamically Updating XML Data: Numbering Scheme Revisited", *World Wide Web 8(1)*, 2005.