Towards Cost-based Optimizations of Twig Content-based Queries

Michal Krátký and Radim Bača

Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
michal.kratky@vsb.cz

Extended Abstract

In recent years, many approaches to indexing XML data have appeared. These approaches attempt to process XML queries efficiently and sufficient query plans are built for this purpose. Some effort has been expended in the optimization of XML query processing [20].

There are not many works that take cost-based query optimizations into account. In work [20], we find some cost-based considerations, however, they work only with one type of structural join and one type of underlying index. There are works depicted two types of query processing as well [10, 17]. The first type applies an element-based index, the second type applies a navigation in a persistent DOM-like structure. In our work, we propose usage of two path-based indices that provide significant potential for a query optimization based on a cost-based join selection.

We can identify some classes of approaches for efficient processing of XML queries. The first class includes approaches based on *shredding* [18] (storing an XML document in many relations, where each element name has its own relation). These approaches work well only on specific documents with a defined XML schema. The second class of approaches [15, 21, 1] provides an *element-based* decomposition of an XML document. These methods usually work with a labeling scheme and they differ mainly in various join algorithms.

We identify two main types of join algorithms. The first join type works with sorted node sets merged by a type of holistic join [3, 4]. This kind of structural join is optimal when a small number of nodes is rejected during the structural join. In this case, nodes are retrieved in a very efficient way with a small number of I/O. In our article, this kind of join operation is called a *merge join*. Another type of join algorithm is based on context nodes [9], where each location step is processed using context nodes from the previous step. We say that this approach utilizes a previously processed location step and uses the nodes for an efficient search of nodes in the next step. If there is a large number of rejected nodes in the join operation, this kind of join is efficient. In our article, this kind of join operation is called a *progressive join*.

There are approaches that decompose XML documents according to the node path, as opposed to the node name. Handling the paths during a query processing

V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 71–73, ISBN 978-80-248-1746-0.

provides some advantages. If the `a/b/c//e/f/g` simple-path query is considered, then the query is evaluated by five structural joins. The evaluation may be very inefficient compared to *path-based* approaches [16, 19, 6, 14, 13, 11] and summarizing approaches [8, 7], even when an additional optimization of structural joins is used [4, 3, 12]. The path-based approaches perform these queries by finding matched labelled paths – a relatively simple task – and then finding relevant nodes with a single index search [16]. Since there is a significantly lower number of labelled paths than nodes in an XML document, the search can be performed very quickly. Supposing there is a set of matched labelled paths, we finish the simple-path query process by finding nodes corresponding to those labelled paths in an inverted list. In work [5], we find a comparison of different decomposition approaches and a labelled-path approach (Prefix-Path streaming in this case) has good experimental results.

In the case of path-based approaches, we can observe the same issue as in the case of element-based approaches. When we join two path results, we can perform it with two different types of joins. One join is based on an inverted index and the second one utilizes the previous query path result.

Chen et al. in [6] compares two path-based approaches to processing twig queries. Whereas the first index, *ROOTPATHS* index, is able to process twig queries only with the merge join, the second index, *DATAPATHS* index, is able to process a query path by utilizing previous query path results. Consequently, DATAPATHS index applies a progressive join algorithm. These indices can outperform each other depending on the query. However, there is no general proposal that can help to decide that the index should be used to achieve the best query evaluation performance. In works [13, 14], we have introduced an index to provide these join operations.

In work [2], we introduce a simple, cost-based, optimization technique for a join selection during a query evaluation. This technique joins the advantages of a simple path query processing based on inverted lists and usage of previous results for a twig query processing. We show that the knowledge of the result size can help to choose a good query evaluation strategy. We utilize advantages of existing, state-of-the-art, path-based approaches, such as [16, 19, 6], to achieve an optimal query performance.

References

1. S. Al-Khalifa, H. V. Jagadish, and N. Koudas. Structural Joins: A Primitive for Efficient XML Query Pattern Matching. In *Proceedings of International Conference on Data Engineering, ICDE 2002*. IEEE Computer Society, 2002.
2. R. Bača and M. Krátký. A Cost-based Join Selection for XML Twig Content-based Queries. In *Proceedings of the Third International Workshop on Database Technologies for Handling XML Information on the Web, DataX 2008, EDBT, Nantes, France*. Accepted, to appear in ACM DL, 2008.
3. N. Bruno, D. Srivastava, and N. Koudas. Holistic Twig Joins: Optimal XML Pattern Matching. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2002*, pages 310–321. ACM Press, 2002.

4. S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S. Candan. Twig2Stack: Bottom-up Processing of Generalized-tree-pattern Queries Over XML documents. In *Proceedings of International Conference on Very Large Databases, VLDB 2006*, pages 283–294. VLDB Endowment, 2006.
5. T. Chen, J. Lu, and T. Ling. On Boosting Holism in XML Twig Pattern Matching Using Structural Indexing Techniques. *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2005*, pages 455–466, 2005.
6. Z. Chen, G. Korn, F. Koudas, N. Shanmugasundaram, and J. Srivastava. Index Structures for Matching XML Twigs Using Relational Query Processors. In *Proceedings of ICDE 2005*, pages 1273–1273. IEEE Computer Society, 2005.
7. C.-W. Chung, J.-K. Min, and K. Shim. APEX: an Adaptive Path Index for XML Data. In *Proceedings of the ACM International Conference on Management of Data, SIGMOD 2002*, pages 121–132, New York, NY, USA, 2002. ACM Press.
8. B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon. A Fast Index for Semistructured Data. In *Proceedings of the 27th International Conference on Very Large Databases, VLDB 2001*, pages 341–350, 2001.
9. T. Grust, M. van Keulen, and J. Teubner. Staircase Join: Teach a Relational DBMS to Watch Its (Axis) Steps. In *Proceedings of the 29th, International Conference on Very Large Databases, VLDB 2003*, pages 524–535. VLDB Endowment, 2003.
10. A. Halverson and et al. Mixed Mode XML Query Processing. In *Proceedings of VLDB 2003*, pages 225–236. VLDB Endowment, 2003.
11. W. H. Hanyu Li, Mong Li Lee. A Path-Based Labeling Scheme for Efficient Structural Join. In *Proceedings of XSym 2005*, pages 34 – 48. Springer–Verlag, 2005.
12. H. Jiang, W. Wang, H. Lu, and J. Yu. Holistic twig joins on indexed XML documents. *Proceedings of VLDB 2003*, pages 273–284, 2003.
13. M. Krátký, R. Bača, and V. Snášel. Implementation of XPath Axes in the Multidimensional Approach to Indexing XML Data. In *Proceedings of DEXA 2007*, volume LNCS 4653/2007. Springer–Verlag, 2007.
14. M. Krátký, J. Pokorný, and V. Snášel. Implementation of XPath Axes in the Multidimensional Approach to Indexing XML Data. In *Current Trends in Database Technology, EDBT 2004*, volume LNCS 3268/2004. Springer–Verlag, 2004.
15. Q. Li and B. Moon. Indexing and Querying XML Data for Regular Path Expressions. In *Proceedings of VLDB 2001*, 2001.
16. T. S. M. Yoshikawa, T.Amagasa and S. Uemura. XRel: a Path-based Approach to Storage and Retrieval of XML Documents Using Relational Databases. *ACM Trans. Inter. Tech.*, 1(1):110–141, 2001.
17. N. May, M. Brantner, A. Böhm, C.-C. Kanne, and G. Moerkotte. Index vs. Navigation in XPath Evaluation. In *Proceedings of Database and XML Technologies, XSym 2006*, volume LNCS 4156/2006, pages 16–30. Springer–Verlag, 2006.
18. J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. DeWitt, and J. Naughton. Relational Databases for Querying XML Documents: Limitations and Opportunities. In *Proceedings of the 25th International Conference on Very Large Databases, VLDB 1999. Edinburgh, Scotland, UK*, pages 302–314. Morgan Kaufmann, 1999.
19. S. S.Prakas and S.Madria. SUCXENT: An Efficient Path-Based Approach to Store and Query XML Documents. In *Proceedings of DEXA 2004*, volume LNCS 3180/2004, pages 285–295. Springer-Verlag, 2004.
20. Y. Wu, J. M. Patel, and H. Jagadish. Structural Join Order Selection for XML Query Optimization. In *Proceedings of ICDE 2003*. IEEE CS, 2003.
21. C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings of ACM SIGMOD 2001*, pages 425–436, New York, USA, 2001. ACM Press.