On the Integration of Structure Indexes and Inverted Lists

Raghav Kaushik Rajasekar Krishnamurthy Jeffrey F Naughton Raghu Ramakrishnan
University of Wisconsin, Madison
{raghav,sekar,naughton,raghu}@cs.wisc.edu

Recently, there has been a great deal of interest in the development of techniques to evaluate path expressions over collections of XML documents. In general, these path expressions contain both structural and keyword components. Several methods have been proposed for processing path expressions over graph/tree-structured XML data. These methods can be classified into two broad classes. The first involves graph traversal where the input query is evaluated by traversing the data graph or some compressed representation. The other class involves information-retrieval style processing using inverted lists. In this framework, *structure indexes* have been proposed to be used as a substitute for graph traversal. These structure indexes are proven to be very effective when applied to queries that examine the "coarse" structure of documents. For example, for many documents, a query //section/figure/title would be evaluated very efficiently by a structure index. Unfortunately, the structure indexing approach is much less successful when we consider queries on "values" or text words in the documents. This is roughly because any summary that retains enough detail to answer such queries has to be big (it has to encode a lot of details about specific values), so running queries over the summary will be no more efficient than running them over the original data. On the other hand, while inverted list processing has proven very effective for keyword searches in the information retrieval (IR) community, when applied to path expression queries over XML documents they are less universally effective. The problem is that evaluating a path may require many joins over large inverted lists, and these joins can be expensive. To the best of our knowledge, no published literature addresses the problem of combining these two forms of auxiliary indexes.

This paper bridges this gap by proposing a strategy that combines structure indexes and inverted lists and a query evaluation algorithm for branching path expressions based on this strategy. Our algorithm does not assume any specific property of these indexes and is applicable for a wide range of structure indexes and inverted list join algorithms. We have implemented our approach in the Niagara native XML data management system [3] and our experiments demonstrate that we can derive substantial benefits by integrating the two forms of indexes.

While finding all documents or elements that satisfy a given path expression is a common use of path expression querying, users who specify keyword-based IR queries typically want just the k most relevant answers. Several proposals have been made to incorporate the IR notion of *relevance* to XML queries. As described in [2], XML search tasks can be divided into Content-Only (CO) tasks where XML documents are searched only using keywords, and Content-and-Structure (CAS) tasks where both structure and content is queried.

In this paper, we focus on a subclass of CAS queries consisting of simple path expressions. We study algorithmic issues in integrating structure indexes with inverted lists for the evaluation of these queries, where we rank all documents that match the query and return the top k documents in order of relevance. We allow a broad class of relevance functions that covers the standard tf-idf notion of ranking and propose instance-optimal methods of pushing down top k computation by combining the forms of indexes. Our approach is based on Fagin et al.'s Threshold Algorithm (TA) [1]. Our setting poses novel challenges, since the ranking function we allow is not necessarily monotonic [1]. Also, unlike TA which is a middleware algorithm, our focus is on the database *server* where additional access paths are available. This violates the assumptions under which TA is shown to be instance optimal. We show that a structure index can be used in conjunction with the ranked inverted lists to design a new algorithm that is instance optimal even in the presence of these access paths.

References

[1] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of PODS*, 2001.

[2] N. Fuhr, M. Lalmas, and S. Malik. INEX: initiative for evaluation of XML retrieval. `http://inex.is.informatik.uni-duisburg.de:2003`.

[3] Niagara query engine. `http://www.cs.wisc.edu/niagara`.

Proceedings of the 20th International Conference on Data Engineering (ICDE'04)
1063-6382/04 $ 20.00 © 2004 IEEE