

XJoin Index: Indexing XML Data for Efficient Handling of Branching Path Expressions

Elisa Bertino
University of Milano, Italy
bertino@dico.unimi.it

Barbara Catania
University of Genova, Italy
catania@disi.unige.it

Wen Qiang Wang
National University of Singapore
wangwq@comp.nus.edu.sg

1 Introduction

Queries for XML documents usually specify selection predicates for multiple elements, related by some tree structured relationships. Such queries are called *branching path expressions* because their graphical representation contains branches and corresponds to a small tree (a *twig*). Solving a branching path expression requires to find all occurrences of the twig inside a set of XML documents. To this purpose, one possible approach consists in first determining elements or attributes matching each node and then applying a *structural join* algorithm to element/attribute sets associated with nodes connected by an edge.

Most structural join algorithms rely on the usage of some indexing technique to more efficiently perform structural join. These techniques usually provide a support for efficiently executing each element-attribute, parent-child or ancestor-descendant relationship but they do not usually vary the number of joins to be performed. Rather, they provide a support for efficiently executing *each* join, required by the query. On the other hand, in the relational context, the join index has been proposed as an index structure to reduce the join overhead [2]. As far as we know, no similar approach has been proposed for XML yet.

In this paper, we consider the problem of indexing XML data for solving branching path expressions with the aim of reducing the number of joins to be executed and we propose a simple yet efficient join indexing approach to shrink the twig before applying any structural join algorithm.

2 XJoin Index

The indexing technique we propose, that we call XJoin Index, precomputes some structural (semi-)join results thus reducing the number of joins to be computed [1]. Precomputed (semi-)joins support the following operations: (i) attribute selections, possibly involving several attributes; (ii) detection of parent-child relationships; (ii) counting selections, like *Find all books with at least 3 authors*. Unlike other approaches, based on specialized data structures,

XJoin Index is entirely based on B⁺-trees and can be coupled with any structural join algorithm proposed so far.

Besides simple indexes supporting a quick retrieval of all elements and attributes with a certain tag name, the XJoin index consists of two additional structures: (i) *counting join index*, that allows one to quickly retrieve all elements with a given tag name having a certain number of children with a certain tag name; (ii) *attribute join index*, that allows one to quickly retrieve all elements with a given tag name having an attribute with a certain tag name.

Even if in the XJoin Index some element information is replicated, the space is still linear in the number of elements and attributes appearing in the XML data set. Moreover, differently from other approaches, any additional attribute or counting predicate inside a query condition does not correspond to a new application of a structural join algorithm but corresponds to a simple set intersection.

The XJoin index is flexible enough to support a large variety of query processing strategies, differing in the number of joins to be executed to solve the original branching query. In order to evaluate the impact of such different query strategies, we have performed several experiments for search and update operations against both synthetic and real datasets. The obtained results show that the XJoin Index effectively reduces the number of structural joins to be executed, processing twig queries by up to an order of magnitude faster than traditional indexing approaches.

Acknowledgements. We would like to thank Professor Beng Chin Ooi for his valuable comments on this paper.

References

- [1] E. Bertino, B. Catania, and W.Q. Wang. XJoin Index: Indexing XML Data for Efficient Handling of Branching Path Expressions. At <ftp://ftp.disi.unige.it/person/CataniaB/xjoin.pdf>.
- [2] P. Valduriez. Join Indices. *ACM Transactions on Database Systems*, 12(2):218-246, 1987.