CCIT Report #693 March 2008

The Space Complexity of Processing XML Twig Queries Over Indexed Documents[*]

Mirit Shalem
Department of Computer Science
Technion, Haifa, Israel
mirit2s@cs.technion.ac.il

Ziv Bar-Yossef
Department of Electrical Engineering
Technion, Haifa, Israel
and Google Haifa Engineering Center
Haifa, Israel
zivby@ee.technion.ac.il

March 27, 2008

Abstract

Current twig join algorithms incur high memory costs on queries that involve child-axis nodes. In this paper we provide an analytical explanation for this phenomenon. In a first large-scale study of the space complexity of evaluating XPath queries over indexed XML documents we show the space to depend on three factors: (1) whether the query is a path or a tree; (2) the types of axes occurring in the query and their occurrence pattern; and (3) the mode of query evaluation (filtering, full-fledged, or "pattern matching"). Our lower bounds imply that evaluation of a large class of queries that have child-axis nodes indeed requires large space.

Our study also reveals that on some queries there is a large gap between the space needed for pattern matching and the space needed for full-fledged evaluation or filtering. This implies that many existing twig join algorithms, which work in the pattern matching mode, incur significant space overhead. We present a new twig join algorithm that avoids this overhead. On certain queries our algorithm is exceedingly more space-efficient than existing algorithms, sometimes bringing the space down from linear in the document size to constant.

1 Introduction

XQuery and XPath [9] queries are typically represented as node-labeled *twig patterns* (i.e., small trees). Evaluating a twig pattern over an XML document is therefore a core database operation. As with relational databases, creating an index over the XML document at a pre-processing step can significantly reduce the costs (time, space) of query evaluation. Similarly to text search, an index for an XML document consists of *posting lists* or *streams*, one for each XML label that occurs in the document. The stream consists of positional encodings of all the elements that have this label, in document order. In this paper we focus on the most popular encoding scheme, the *BEL encoding* [6], in which each element is encoded as a (Begin,End,Level) tuple. The BEL encoding, although being compact, enables simple testing of structural relationships between elements.

Over the past decade, many algorithms for evaluating twig queries over indexed XML documents have been proposed (e.g., [6, 7, 10, 23, 22, 25, 18]). Much progress has been made in supporting wider fragments

[*]Supported by the European Commission Marie Curie International Re-integration Grant.