R-Programs: A Framework for Distributing XML Structural Joins across Function Calls

David Bednárek⋆
Department of Software Engineering, Faculty of Mathematics and Physics
Charles University in Prague
bednarek@ksi.mff.cuni.cz

Abstract. Structural joins and, in particular, twig joins are essential operations in XML query processing. Algorithms presented so far treat a twig join as a single operator with multiple inputs. However, in XQuery and XSLT, a twig pattern may be scattered across several functions (templates); thus, function integration is required before the application of a twig join operator. This paper presents R-programs – a novel evaluation framework based on an expanding network of operators. In this environment, a function may repeatedly and bidirectionally interact with its caller; consequently, a structural join algorithm may be distributed across the boundary of a function. Given this ability, function integration is no longer required and twig join algorithms become applicable even in the presence of recursive functions.

1 Introduction

The most successful XML query-processing methods are based on the architecture inherited from relational database systems [1]. The XML-specific part of query processing is concentrated mainly in various versions of structural join operators. A number of methods that process more than one structural relationship at a time were developed; both theory and experiments show that such holistic approach is advantageous because it avoids unnecessary intermediate results. In the query, twig patterns are detected [2], containing ancestor-descendant or parent-child relationships, OR operators, and, in the most advanced methods, negation. For example, a twig pattern, corresponding to the XPath expression a/b[c]/d/e is shown in Fig. 2a.

In the physical plan, a holistic twig join is represented by an n-ary operator that consumes n streams assigned to the nodes of the twig pattern, as shown in Fig. 2b. Each stream carries XML elements that meet the local condition of the corresponding twig node (in our example, the condition on their element names).

This approach works well when a twig pattern is located inside an XPath expression. However, functions may be defined in XQuery and, thus, a twig

* Supported by the Project GA201/09/0983 – Agile systems and service-oriented software of the Czech Science Foundation.
pattern may be scattered across function boundaries. An example, containing the same pattern as in Fig. 2b, is shown in Fig. 1.

In such a case, integration of the bodies of the functions into the main expression is required because the twig-join algorithm is represented as an atomic physical operator that can not be spanned across function calls. Such integration is expensive (as it may lead to exponential expansion of the program size) and generally impossible in the presence of recursive functions.

In this paper, we suggest that centralized implementations of twig joins be replaced by distributed versions as shown in the example in Fig. 2c. Note that the word "distributed" does not necessarily mean that the algorithm is run on more than one computing node; nevertheless, the original centralized algorithm is divided into smaller blocks that run conceptually in parallel, communicating using pipes.

Our approach allows a twig algorithm to run on the whole twig pattern scattered across several functions, even though there is no inter-procedural detection of twig patterns. Each function may be compiled independently, producing a partial network; at run-time, the connected networks will behave exactly like a twig join algorithm.

Although this idea is quite simple, it requires significant changes in the architecture of the query evaluation system. At run time, function evaluation may no longer follow the usual call-return scheme. Instead, function bodies must run in parallel with their callers and they must be able to exchange data in both directions.

In this paper, we present a mathematical model of such a system, named R-program. R-programs may be used as an intermediate language for both logical and physical plans, depending on the set of operators used.

An R-program is conceptually a network of computing nodes, connected by pipes carrying relations (more exactly, sequences of tuples). Each node performs a relational or XML-specific operation. Instead of calling, functions are expanded, i.e. added to the existing network at run-time so that they can run in parallel with their callers, exchanging data in both directions. While there is one-to-one correspondence between R-program functions and XQuery functions, a single expansion of an R-program function corresponds to multiple calls to the corresponding XQuery function. This fact ensures that the cost of the expansion is relatively small with respect to the amount of data processed.

The rest of this paper is organized as follows: The principle of twig join and the common skeleton found in most existing twig join algorithms is reviewed in Sect. 2. In Sect. 3 we will show how a twig-join algorithm may be rewritten in