A flexible extension of XQuery Full-Text

Emanuele Panzeri and Gabriella Pasi

University of Milano-Bicocca
Viale Sarca 336, 20126 Milano, Italy
{panzeri,pasi}@disco.unimib.it

Abstract. This paper presents the implementation of an extension of the XQuery Full-Text language on top of the BaseX query engine. The proposed extension adds to the language two new flexible axes that allow users to express structural constraints that are evaluated in an approximate way with respect to a considered path; the constraints evaluation produces a scored set of elements. The implementation and the efficiency evaluations of the constraints are reported in this paper.

1 Introduction

Recent works have been dedicated at improving standard XML query languages, such as XQuery and XPath, by enriching their expressiveness in both content constraints [1, 8] and structural constraints [2, 6] evaluation. While the work reported in [1] has been adopted by W3C in the XQuery Full-Text extension[7], no approximate matching for structural-based constraints has been standardized by W3C yet. The adoption of structured query models (such as XQuery) to inquiry highly structured document repositories or XML databases forces users to be well aware of the underlying structure; non of the previous approaches allows users to directly specify flexible structural constraints the evaluation of which produces weighted fragments. The XQuery Full-Text language extension proposed in [5] was the first proposal to introduce a set of flexible constraints with an approximate matching. The extension allows users to formulate queries where the relative position of important nodes can be specified independently from an exact knowledge of the underlying structure. The extension gives to the user the ability to express structural constraints with approximate matching and to obtain a weighted set of fragments; users can also define a score combination using standard XQuery operators and obtain a customized element ranking.

In this work we present the implementation of the flexible constraints, as defined and motivated in [5], named `Near` and `Below`, that allow users to explicitly specify their tolerance to an approximate structural matching. The implementation, performed on top of the BaseX query engine [4], integrates and extends the fragment scoring introduced by the FullText extension by taking into account also the structural scores computed by the approximate constraint evaluation.

2 The XQuery Full-Text extension in a nutshell

For each element matched by the `Below` and `Near`, a score is computed by the approximate matching; the score is in the interval $]0, 1]$ where 1 represents a full

satisfaction of the constraint evaluation, while values less than 1 are assigned to target nodes *far* from the context node.

The constraint `Below` is defined as an XPath axis (like, for example, the `children`, `self`, etc axes) the evaluation of which is aimed at identifying elements that are direct descendants of a node. The `Below` constraint is specified as: `c/below::t`, where `c` is the *context* node, and `t` is the target node. The score computed by the `Below` axis evaluation, is computed by the formula: $w_{below}(c,t) = \frac{1}{|desc\_arcs(c,t)|}$. Where $desc\_arcs(c,t)$ is a function that returns the set of unique descending arcs from c to t.

The constraint `Near` is specified as a flexible axis of a path expression; it allows to identify XML elements connected through *any path* to the *context node*. The axis allows to define a maximum distance n that acts as a threshold on the number of arcs between the context node and the target node; nodes the distance of which is more then n arcs are filtered out from the possible results. The `Near` syntax is: `c/near(n)::t` and the score for its evaluation is computed as: $w_{near}(c,t,n) = \begin{cases} \frac{1}{|arcs(c,t)|} & \text{if } |arcs(c,t)| \leq n \\ 0 & \text{else.} \end{cases}$ where c is the context node, t is the current target node, n is the maximum allowed distance and $arcs(c,e)$ returns the set of arcs in the shortest path between c and t.

3 Implementation

The new axes have been integrated into the BaseX XQuery engine by extending both its language interpreter and its XQuery evaluation processor to include a new `score-structure` Score Variable definition. BaseX has been chosen for being the first system (and the only one, to the best of our knowledge) to implement the full XQuery Full-Text language. As described in [3], BaseX adopts an efficient indexing schema for XML documents. The XQuery FLOWR clauses have been made capable to identify the new structural score variable, and to allow its usage in sorting, ordering and results display. As an example the XQuery `for` clause has been extended as follows:

```
ForClause ::="for" "$" VarName TypeDeclaration? PositionalVar? FTScoreVar?
  StructScoreVar? "in" ExprSingle ("," "$" VarName TypeDeclaration?
  PositionalVar? FTScoreVar? "in" ExprSingle)*
StructScoreVar ::= "score-structure" "$" VarName
```

where `Varname` is a valid variable name; `TypeDeclaration` is a variable type declaration; and `ExprSingle` is the actual query for node selection as defined in the XQuery language. From the user point of view this approach offers unlimited possibilities of the usage of the new Structure Score Variable: the user can define aggregation functions using the default XQuery constructs.

Fig. 1a shows an example of the `Near` constraint application: the query `person//act/near::title` is evaluated and the three gray `title` nodes are matched with a score of 0.3, for the `act/movie/title` node, and 0.25 for the other two nodes. In Fig. 1b the evaluation of the query `person/below::name` is shown: three `name` nodes are retrieved; `person/name` node with a score of 1 and 0.3 for the other nodes.

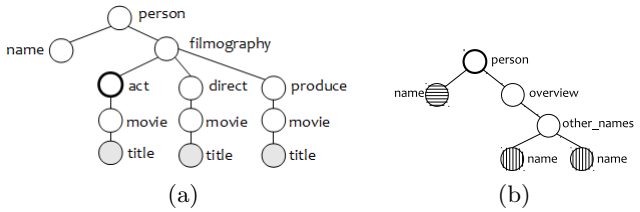

Fig. 1: Example of (a) `Near` and (b) `Below` constraint evaluation.

4 Evaluation

The performed evaluations compare the efficiency of all the new axes constraints with the *standard* XPath/XQuery counterparts (if available): in particular for the `Below` axis evaluation we executed each query using both the `Below` and the `descendant` axes. Concerning the `Near` axis evaluation, instead, no counterpart could be identified due to the innovative nature of the proposed axis.

The axes evaluations have been performed by using the IMDB INEX Data-Centric collection. Performance tests have been executed with an increasing size of the evaluated collection to verify the overhead introduced by the flexible axis evaluation in comparison with standard (if applicable) XPath axes constraints. Due to the nature of the BaseX indexing system that caches queries, result set, and opened databases, the evaluations have been performed by unloading the BaseX system between each run. All evaluation tests have been executed 5 times, and the average timings (removing the worst and the best results) are presented.

`Below` axis evaluation: The `Below` axis has been compared with the standard `descendant` axis: both axes have been evaluated by executing the test without any query optimization introduced by BaseX. Five queries containing the `Below` axis have been evaluated against each collection by measuring its execution time. The same query, with the `Below` axis replaced by the `descendant` axis has then been executed and its timings compared. In Fig. 2 the evaluation results are sketched: not surprisingly the `Below` axis evaluation takes more time than the equivalent `descendant` axis to obtain the query results, due to the computation of the structural score. The `Below` axis evaluation takes in average 36% more time than the execution of the `descendant` counterpart.

`Near` axis evaluation: The `Near` axis evaluation has been performed by using the same IMDb collection used for the evaluation of the `Below` axis. The queries used during the evaluation process have been defined so as to require the BaseX engine to retrieve all the XML elements without neither adopting any optimization strategy nor any query re-writing; this aspect forced the BaseX system to perform a sequential analysis of the target nodes, and thus to provide a complete execution of the `Near` axis evaluation. Furthermore the BaseX Full-Text index has been avoided, further enforcing the complete iteration over any target node without using any BaseX pre-pruning strategy. These aspects allowed to measure the efficiency of the `Near` axis evaluation implementation.

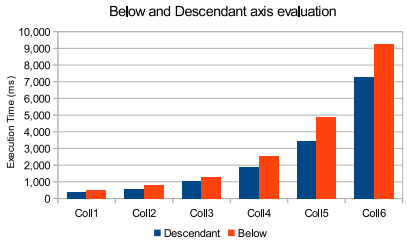

Fig. 2: Comparison between `Below` and `descendant` axis evaluation.

5 Conclusions and Future Work

The `Below` and `Near` axes, semantically and syntactically defined in [5] have been implemented and evaluated on top of the BaseX system, where both the query interpreter and the evaluation engine have been extended to identify and evaluate the new axes. The obtained results confirm that, although the flexible evaluation of both axes requires relatively longer times, the proposed flexible evaluation and the subsequent XML element ranking based on both textual and structural constraints can be successfully introduced into the XQuery language. Ongoing works are being conducted related to the definition, alongside the BaseX data structures, of ad-hoc indexes to better evaluate the new flexible constraints by adopting efficient pruning techniques during target node identification, thus further improving the axis evaluation performance.

References

1. S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. In *WWW '04*, pages 583–594. ACM, 2004.
2. S. S. Bhowmick, C. Dyreson, E. Leonardi, and Z. Ng. Towards non-directional Xpath evaluation in a RDBMS. In *CIKM '09*, pages 1501–1504, 2009.
3. C. Grün. *Storing and Querying Large XML Instances*. PhD thesis, Universität Konstanz, December 2010.
4. C. Grün, S. Gath, A. Holupirek, and M. H. Scholl. XQuery Full Text Implementation in BaseX. In *XSym '09*, pages 114–128, 2009.
5. E. Panzeri and G. Pasi. An Approach to Define Flexible Structural Constraints in XQuery. In *AMT*, pages 307–317, 2012.
6. B. Truong, S. Bhowmick, and C. Dyreson. Sinbad:towards structure-independent querying of common neighbors xml databases. In *DASFAA'12*, pages 156–171. 2012.
7. W3C. XQuery/XPath FullText. www.w3.org/TR/xpath-full-text-10, March 2011.
8. C. Yu and H. V. Jagadish. Querying Complex Structured Databases. In *VLDB '07*, pages 1010–1021, 2007.