Query optimization in XML structured-document databases

Abstract While the information published in the form of XML-compliant documents keeps fast mounting up, efficient and effective query processing and optimization for XML have now become more important than ever. This article reports our recent advances in XML structured-document query optimization. In this article, we elaborate on a novel approach and the techniques developed for XML query optimization. Our approach performs heuristic-based algebraic transformations on XPath queries, represented as PAT algebraic expressions, to achieve query optimization. This article first presents a comprehensive set of general equivalences with regard to XML documents and XML queries. Based on these equivalences, we developed a large set of deterministic algebraic transformation rules for XML query optimization. Our approach is unique, in that it performs exclusively deterministic transformations on queries for fast optimization. The deterministic nature of the proposed approach straightforwardly renders high optimization efficiency and simplicity in implementation. Our approach is a logical-level one, which is independent of any particular storage model. Therefore, the optimizers developed based on our approach can be easily adapted to a broad range of XML data/information servers to achieve fast query optimization. Experimental study confirms the validity and effectiveness of the proposed approach.

Keywords XML query optimization · XML query processing · XML database · Query transformation · Deterministic query optimization

1 Introduction

XML (eXtensible Markup Language) [44] has become the de facto standard for information/data representation and exchange on the Internet and elsewhere. As a consequence, more and more data sources switch over to XML and express their contents using XML or an XML dialect, e.g., the familiar HTML, NITF (in the news industry), WeatherML (for weather information), CellML (in bioinformatics), and XMLPay (for Internet-based payments). The rapidly growing XML data sources call for commensurate management solutions that are XML-aware. It has been a wide consensus that XML documents/data should obtain the same type of management functionalities as conventional data received from RDBMSs, and the database community is well underway towards this destination.

In recent years, many storage schemes for XML data have been proposed, e.g., mapping XML data to relational [5, 18, 21, 38] or object-relational models [28, 41], using special-purpose databases such as semistructured databases [32, 33], or developing native XML databases [19]. A key issue that faces every XML data management system is the optimization of XML queries.

Query optimization in the context of XML databases is extremely challenging. The main reason for this is the high complexity of the XML data model, as compared with other data models, e.g., relational models. This high complexity renders a much enlarged search space for XML query optimization. Furthermore, XML applications are typically Web-hooked and have many simultaneous, interactive users. This dynamic nature requires highly efficient XML query processing and optimization. The classical cost-based and heuristic-based approaches yield unacceptably low efficiency when applied to XML data—query optimization itself becomes very time-consuming because of the huge
search space for optimization caused by the high complexity of the XML data model. Lots of work related to XML query processing has been done, but the majority is focused on investigation for efficient supporting algorithms [13, 17, 24, 26, 31, 32, 40, 47] and indexing schemes [9, 10, 25, 31, 35]. Complete and systematical work on XML query optimization has hardly been reported. We will discuss related work in detail in Sect. 7.

1.1 Motivation

The essential difference between XML data and traditional data (e.g., relational data) is the extra structural relationships between the various elements in an XML data source. On the one hand, these structural relationships render high complexity for XML data modeling and query optimization; on the other hand, they imply invaluable opportunities—source of semantic knowledge for efficient XML query optimization—which is, however, often overlooked or inadequately exploited. For example, assume a query asks for the instances of element type $t_1$ that are subelements of the instances of element type $t_2$ and if we know that every $t_1$ element has to be contained in a $t_2$ element according to the data sources DTD or XSD (XML Schema Definition), then we can simply return all instances of type $t_1$ without checking the containment relationship between the elements of the two types. This is a trivial example of using the structure knowledge of XML data to conduct a deterministic transformation on an input query for better evaluation efficiency. Here, by deterministic, we mean that the transformation once carried out will bring in definite improvement (simplification in this case) on the input query expression. More complicated cases exist that yield plenty of opportunities for efficient optimization of XML queries, e.g., exploiting an available structure index which is otherwise inapplicable to an input query (A structure index for now can be thought of as a shortcut between two distant elements within the structure of the XML data source). Obviously, using a structure index can significantly reduce the cost of evaluating a covered containment operation. For example, let $t_1$, $t_2$, and $t_3$ be three element types; $t_1$ is a distant descendent of $t_2$, and $t_2$ is a direct child of $t_3$ (see Fig. 1); if the evaluation of a query needs to check the containment relationship between $t_1$ and $t_2$, a straightforward but costly way is to traverse all the intermediate “generations” between $t_1$ and $t_2$; however, if a structure index between $t_1$ and $t_3$ is available, we can bypass the long path (from $t_1$ to $t_2$) by using the structure index to reach $t_2$’s parent, $t_3$, first, and then get to the target $t_2$.

The above examples indicate a huge realm where efficient XML query optimization can be pursued by exploiting the structure knowledge of XML data and structure-related semantics carried on by an XML query expression. This observation motivated the work reported in this writing and set the keynote of this article.

XML query optimization is one of the most challenging issues facing the database research community. It is only obtainable through a systematically and carefully worked-out approach. To this end, we first need to study the equivalence issue related to XML queries and XML data because query transformation has to be based on query equivalence. Secondly, we need to work out a good strategy to efficiently and effectively accomplish XML query optimization by using these equivalences. The strategy turns out to be a real challenge because, as mentioned before, we are now confronting a much enlarged search space, from which equivalent transformations are to be conducted in order to locate an optimal evaluation plan. A pretty straightforward guideline in our mind is to find a way to radically prune the search space during query optimization but still be able to obtain a sufficiently good plan though it may not be an optimal one.

The solution we come up with is the so-called deterministic optimization approach. By this approach, every transformation applied to a query has to be deterministic, in the sense that it is bound to produce nontrivial improvement on the input query in terms of evaluation efficiency. This type of transformation, by its nature, is heuristic-based and relies on proper exploitation of the structure knowledge of XML data. Often in a scenario, a structure index is available but not applicable because of the particular, unfavorable structure pattern of a query. However, the hidden opportunity of eventually applying the index to the query may exist and can be identified by guided transformations that exploit particular structural properties of the source XML data. Deterministic transformations are achieved in our approach via invocation of heuristic transformation rules, which are based on the more general equivalences of XML queries. Thirdly, we need a group of supporting functions to facilitate the implementation of our approach. All these issues are addressed systematically in this article.

1.2 Scope of the article

In this article we address the issues related to efficient XML query optimization at the logical level. Instead of directly targeting the full-fledged XQuery [2], which is emerging as the standard query language for XML, this article focuses on its core sublanguage, XPath [14], which is small and can