Information Systems 35 (2010) 315–334

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Query rewritings using views for XPath queries, framework, and methodologies

Jian Tang[a,*], Ada Waichee Fu[b,1]

[a] Department of Computer Science, Memorial University of Newfoundland, Elizabeth Ave, St. John's NL, Canada A1B 3X5
[b] Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong

ARTICLE INFO

Article history:
Received 9 September 2006
Received in revised form
26 January 2009
Accepted 26 October 2009
Recommended by: Y. Ioannidis

Keywords:
XML
Query
View
Rewriting
Pattern
Containment
Embedding
Maximality

ABSTRACT

Query rewriting using views is a technique that allows a query to be answered efficiently by using pre-computed materialized views. It has many applications, such as data caching, query optimization, schema integration, etc. This issue has been studied extensively for relational databases and, as a result, the technology is maturing. For XML data, however, the work is inadequate. Recently, several frameworks have been proposed for query rewriting using views for XPath queries, with the requirement that a rewriting must be complete. In this paper, we study the problem of query rewriting using views for XPath queries without requiring that the rewriting be complete. This will increase its applicability since in many cases, complete rewritings using views do not exist. We give formal definitions for various concepts to formulate the problem, and then propose solutions. Our solutions are built under the framework for query containment. We look into the problem from both theoretic perspectives, and algorithmic approaches. Two methods to generate rewritings using views are proposed, with different characteristics in terms of generalities and efficiencies. The maximality properties of the rewritings generated by these methods are discussed.

1. Introduction

Since it emerged as a language for information transfer and storage late last century, XML has caught increasing attentions from the research communities across different disciplines for its flexible encoding schemes and expressive power. Recently, due to the near completion of the standardization of XQuery language [25], the usage of XML has reached far beyond the simple information-encoding domain. Such a broad adoption has led not only to the developments of new paradigms, but also the reformulations of some existing theories and methodologies in relational databases. One of the areas in which such a reformulation is in a pressing need is in query processing. Due to the relatively complex structure of XML documents compared with relation tables, efficiency in querying XML documents has become one of the most widely investigated topics in recent years.

XQuery employs XPath as its core sub-language for navigating XML documents. In the query processing literature for XML documents, therefore, a lot of attentions have been in XPath processing. One approach is query rewriting. It is a technique that allows a query to be answered efficiently by using pre-computed materialized views. It has many applications, such as data caching, query optimization, schema integration, etc. This issue has been studied extensively, in both theoretical and algorithmic aspects, for relational databases. As a result, sound theories and methodologies have been proposed in that context. For XML data, however, the work is inadequate. Recently, several frameworks have been proposed for query rewriting using views for XPath queries, with the requirement that a

* Corresponding author. Tel.: +1 7097374580; fax: +1 7097372009.
E-mail addresses: jian@mun.ca (J. Tang), adafu@cse.cuhk.edu.hk (A.W. Fu).
[1] Tel.: +852 26098432; fax: +852 26035024.

rewriting must be complete, being that *all* the answers to the query running on the original database must be generated by the rewriting running on the materialized views [2,3,22]. Requiring a complete rewriting using views, however, is not always realistic. The most prominent scenario is in the area of schema integration. Local-as-view (LAV) is an important strategy for schema integration in database systems [8]. In the LAV model, queries are written over a global schema (also called a mediated schema). Views are queries that describe the contents of the local data sources, and are also written over the global schema. The model does not assume the existence of a separate 'original' data source. The direct goal of a query is to retrieve information from the local data sources. To accomplish this, it must be rewritten into some query that can run on the local data sources and produce the answer it needs. Since there is no original data source, 'completeness' of the query results generated from the local data sources is not required. Incomplete rewriting may also be useful in the case where the original data are not conveniently accessible but a materialized view is available. In this case, a user has an option to use the materialized view to obtain an answer which is only a proper subset of the answer he/she has expected from the original data.

It is well known that if a rewriting using view is complete, then the query and its rewritten version are equivalent. Such an equivalence provides valuable information to guide an algorithm to generate the rewriting. If the completeness requirement is removed, however, the aforementioned information will not be available. This makes generating a rewriting using views with desirable properties a challenging task. In this paper, we study the problem of query rewriting using views for XPath queries without requiring the rewriting be equivalent to the original query. We give formal definitions for various concepts to formulate the problem. We look into the problem from both theoretic perspectives, and algorithmic approaches, and then propose solutions. Our solution is built on top of the theories for query containment. We introduce the concept of 'trap', based on which two methods to generate rewritings using views are proposed. These methods have different characteristics in terms of generalities and efficiencies. We describe conditions under which our generated rewritings are optimal. The class of the XPath queries that we consider in this paper uses four kinds of symbols, /, //, [·], and *, which, respectively, denote child axis, descendant axis, branches, and wildcard. We denote this class by $XP^{[/,\ //,\ [\ ],\ *]}$. The query in this class can be described in a condensed grammar as follows:

$$X = X/X|X\|X|X[X]|L|^*$$

where L denotes labels from an infinite symbol set. We will omit tagging templates that normally accompany XPath queries, and concentrate only on the navigation scripts. This is because, technically, tagging templates and navigation scripts are orthogonal, and the navigation scripts are where the most technical sophistications arise in the query rewriting.

The rest of the paper is organized as follows. In Section 2, we propose a model, and introduce related concepts, and then precisely define the problem. In Section 3, we introduce two alternative solutions to the problem, and discuss their strengths and limitations. In Section 4, we describe the conditions under which the rewritings using views generated by our methods are optimal. Section 5 concludes the paper by summarizing the main results, and suggesting some issues for further study.

1.1. A motivating example

If the information required by a query has already been included in the result of a materialized view, and there is a way to retrieve it, then this usually will make the query processing more efficient. Consider the XPath query */publication/book[@review_id]//author/name*. This query requires the names of the authors of the books which belong to the publication category and have a review_id attribute. Let us consider three cases: (1) The view is */publication/book*. In this case the answer of the query is retrievable from the result of the view. It is because, according to the execution semantics, this query will return the entire subtree rooted at each book node in the input document tree. (2) The view is */publication[@permit_no]/book*. In this case, we cannot retrieve a complete answer to the query from the view result. This incompleteness is due to the lack of data, and is intrinsic to the way the queries are formulated. (3) The view is the same as that in the first case, but the query is */publication[@permit_no]/book[@review_id]//author/name*. In this case, the required information is not retrievable. This is not due to the lack of data, but due to the lack of knowledge: we do not know which book in the view result is a child of a publication with permit_no attribute. Obviously, the first case is most preferable. However, given the way the queries are formulated in the second case, if we can get everything contained in the result that fits the user's requirement, it may still be useful if sufficient data is difficult to obtain. Note that the problem arising in the third situation is worse than that in the second case: we cannot retrieve any answer without risking an error.

2. Concepts and definitions

In this paper, we will use the following notations. For any tree or path t, $|t|$ denotes the number of nodes in t. We use $\langle a,\ldots,b\rangle$ to denote a generic path which can be of any length (in nodes), where a and b are the start and the end nodes of the path, respectively, while $\langle a\rangle$ and $\langle a, b\rangle$ denote a path with a length of one and two, respectively. For two graphs, in particular, trees, we use the terms 'isomorphic', 'equal', 'same' interchangeably. Sometimes, for easy presentation and notations, we will allow same nodes (and associated edges) to belong to multiple trees, and therefore avoid using isomorphism symbols on them.

2.1. Pattern tree and input tree

An XPath query can be denoted as a tree, called a *pattern tree* (or simply pattern). Each node is attached with a label, except for the root. The tree may contain