Answering Keyword Queries on XML Using Materialized Views

Ziyang Liu, Yi Chen

Department of Computer Science and Engineering, Arizona State University
P.O. Box 878809, Tempe, AZ 85287 - 8809, USA
{ziyang.liu, yi}@asu.edu

***Abstract*— Answering queries using materialized views has been well studied in the context of structured queries and has shown significant performance benefits. Despite the popularity of keyword search over XML data, it is an open problem whether materialized views can be leveraged for query evaluation. In this paper, we investigate this problem and present techniques for answering keyword queries using a minimal number of materialized views. Experimental evaluation demonstrates the efficiency of the proposed techniques.**

I. Introduction

XML has been used as the standard data representation format for web and scientific applications. Since a typical web user or a scientific user does not know structured query languages (such as XPath, XQuery), and the data schema may be unavailable, complex, or fast-evolving, providing keyword search on XML becomes critical in those applications. Though much work has been done on inferring the semantics of XML keyword search by defining query results appropriately ([6], [4], [5], [1], [2]), the problem of how to efficiently evaluate and optimize XML keyword search has not yet been well addressed.

By avoiding computing query results directly from the source data, exploiting materialized views has been proven crucial for performance optimization in evaluating SQL queries on databases and XPath/XQuery on XML. Caching query results as materialized views in web applications can also reduce the workload of servers and network traffic.

Given the benefits of materialized views in structured query processing, it is a natural idea to leverage them to speed up XML keyword search. However, to the best of our knowledge, there is no research conducted on answering XML keyword search using materialized views.

Many problems need to be addressed in order to answer XML keyword queries using a set of materialized views of previous search results. If the query is the same as a view that has been materialized, we can simply access the materialized view and return it to the user. If the query is similar but not identical to existing materialized views, is it still possible to utilize the views for query evaluation in order to gain performance improvements? The answer actually depends on the definition of query results. Then, are there any existing XML keyword search semantics for which a query can be answered using views? If so, which views are *relevant* to a given query, i.e. they can be used to answer the query?

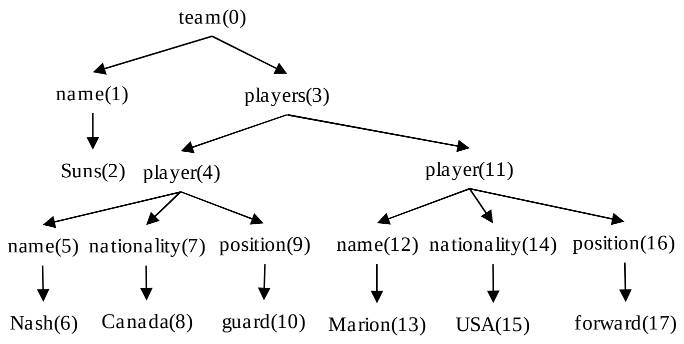


Fig. 1. Sample XML Tree

Furthermore, if there are several sets of views that can answer a query, which set should be used? Finally, how should we use such a set of views to answer the query?

To the best of our knowledge, this is the first work on answering XML keyword searches using materialized views. We find that it is possible to answer queries using views for a widely adopted XML keyword search semantics: *Smallest Lowest Common Ancestor* (SLCA) [6]. While SLCA will be formally defined in Section II, we illustrate it by an example. Consider a query {*Nash*, *position*} and the XML document in Figure 1 where each XML node is associated with a unique identifer. Though there can be many XML data nodes matching keyword *position*, only the one with ID 9 should be considered as closely related to the match of *Nash* since they refer to the same player. This can be detected using the SLCA semantics.

For SLCA semantics, we analyze whether a view is relevant for answering a given query. We then prove that the problem of finding the smallest set of materialized views that can answer a query is NP-hard. A polynomial time algorithm is developed to find such a set with $\ln|Q|$ approximation, where $|Q|$ is the number of keywords in the query. Finally we show how to answer queries using the selected set of views. Experimental evaluation shows significant performance improvements of answering queries using views. The techniques that we propose for exploiting materialized SLCAs of XML keyword searches can be incorporated into existing XML keyword search engines that adopt SLCA semantics for defining query results[1], including XSeek [4], XKSearch [6], [2] and [5].

[1]Some systems perform additional node filtering after SLCA computation is done.

II. Background

We model XML data as a directed tree, such as the one shown in Figure 1. Each internal node represents an element or attribute, and each leaf node represents a data value.

A query is specified as a set of keywords. We use the following notations for keyword search queries.

Definition 2.1: Let Q_1 and Q_2 be keyword queries. We say Q_2 is a subquery of Q_1, or Q_1 is a superquery of Q_2, denoted as $Q_2 \subseteq Q_1$, if every keyword in Q_2 is in Q_1. The size of a query Q, denoted as $|Q|$, is the number of keywords in Q. ■

We define the query result of XML keyword search according to SLCA [6].

Definition 2.2: The *query result*, or *materialized view* of a keyword search Q on XML data $\mathcal{D}$, is the set of SLCA nodes $SLCA(\mathcal{D}, Q)$, which consists of all the nodes s defined as follows:

1) s contains matches to all the keywords in Q in its subtree.
2) There does not exist a descendant of s that contains matches to all the keywords in Q in its subtree.

Query and view are used interchangeably in the paper. ■

Recall query $\{$*Nash, position*$\}$ on the XML tree in Figure 1. Though node *players* contains matches to both keywords in the query in its subtree, it is not considered as an SLCA node since its child *player* (4) already contains matches to both keywords. While *player* (4) is an SLCA node. This reflects that the keyword matches connected through the *player*(4) node have a closer relationship (and therefore should be returned in the query result) compared with the ones connected through the *players* node.

For the convenience of notation, we also refer SLCA with respect to sets of XML nodes instead of keywords.

Definition 2.3: Let $N_1, N_2, \ldots, N_n$ be n sets of nodes in XML data $\mathcal{D}$. $SLCA(\mathcal{D}, N_1, N_2, \ldots, N_n)$ consists of all such nodes s:

1) s contains at least one node in each N_i ($1 \leq i \leq n$) in its subtree.
2) There is no descendant of s that contains at least one node in each N_i ($1 \leq i \leq n$) in its subtree. ■

As we can see, suppose query Q contains keywords $k_1, \ldots, k_n$, and let M_i be the set of matches to k_i ($1 \leq i \leq n$), then $SLCA(\mathcal{D}, Q) = SLCA(\mathcal{D}, M_1, M_2, \ldots, M_n)$.

III. Exploiting Materialized Views to Answer Queries

In this section, we discuss how to leverage materialized views for query evaluation. Obviously if a materialized view is exactly the same as the query, it directly gives the query result. In the following, we focus the discussion on evaluating queries that are different from existing views.

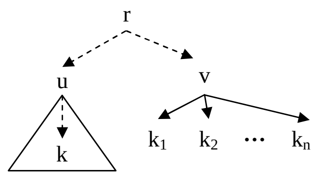


Fig. 2. An Example for Proposition 3.3

A. Identifying Relevant Views

First, we need to find out the inherent relationship among queries in order to identify relevant views for a given query. The following propositions show that the result of a query Q can be computed only from the results of its subqueries.

Proposition 3.1: For two queries Q_1 and Q_2, $SLCA(\mathcal{D}, Q_1 \cup Q_2) = SLCA(\mathcal{D}, SLCA(\mathcal{D}, Q_1), SLCA(\mathcal{D}, Q_2))$ for any data $\mathcal{D}$.

Proof Sketch: According to Definition 2.2, for each $s \in SLCA(\mathcal{D}, Q_1 \cup Q_2)$, s contains all the keywords in both Q_1 and Q_2, therefore it must contain at least one node in $SLCA(\mathcal{D}, Q_1)$ and one node in $SLCA(\mathcal{D}, Q_2)$ in its subtree. Furthermore, none of s's descendants can contain a node in $SLCA(\mathcal{D}, Q_1)$ and a node in $SLCA(\mathcal{D}, Q_2)$ in its subtree. According to Definition 2.3, all such s nodes compose $SLCA(\mathcal{D}, SLCA(\mathcal{D}, Q_1), SLCA(\mathcal{D}, Q_2))$. ■

Corollary 3.2: For query $Q = Q_1 \cup Q_2 \cup \ldots \cup Q_k$, we have $SLCA(\mathcal{D}, Q) = SLCA(\mathcal{D}, SLCA(\mathcal{D}, Q1), \ldots, SLCA(\mathcal{D}, Q_k))$ for any $\mathcal{D}$. ■

Proposition 3.3: For two queries Q_1 and Q_2, if $Q_1 \nsubseteq Q_2$, then $SLCA(\mathcal{D}, Q_1)$ can not provide useful information to compute $SLCA(\mathcal{D}, Q_2)$, for all data $\mathcal{D}$.

Proof Sketch: Let query Q_1, $k \in Q_1$, and $Q_2 = \{k_1, k_2, \ldots, k_n\}$, thus $Q_1 \nsubseteq Q_2$. Suppose the proposition does not hold, that is, $SLCA(\mathcal{D}, Q_2)$ can be obtained from $SLCA(\mathcal{D}, Q_1)$ for any $\mathcal{D}$. We show a counter example. Consider the XML data fragment $\mathcal{D}$ in Figure 2, where a solid line denotes a parent-child edge and a dotted line denotes an ancestor-descendant path, and $SLCA(\mathcal{D}, Q_1) = \{u\}$. As we can see, $v \in SLCA(\mathcal{D}, Q_2)$, $v \notin SLCA(\mathcal{D}, Q_1)$. $SLCA(\mathcal{D}, Q_1)$ does not provide information about the location of node v, which essentially could be anywhere in $\mathcal{D}$, and therefore is not helpful for computing $SLCA(\mathcal{D}, Q_2)$. ■

In summary, to compute the results of a query Q, a materialized view that is a subquery of Q is *relevant*. If there is a keyword $k \in Q$, such that none of the subqueries of Q containing k is materialized, then we need to access the original data $\mathcal{D}$ and find the matches to k to compute $SLCA(\mathcal{D}, Q)$.

Example 3.4: Consider evaluating $Q = \{A, B, C, D, E\}$ given a set of materialized views $\mathcal{V}$: $Q_1 = \{A, B\}$, $Q_2 = \{A, B, C\}$, $Q_3 = \{D\}$, $Q_4 = \{B, D\}$, $Q_5 = \{E, F\}$. Q_1, $\ldots$, Q_4 are subqueries of Q and therefore are relevant to Q. Since keyword E is not in any view that is a subquery of Q, we need to access the matches to E in the source data. ■

B. Evaluating Queries

Although any subquery of a query Q is relevant for computing the result of Q, we should find the smallest set of materialized views that can maximally cover the keywords in Q in order to reduce the size of intermediate results and achieve efficiency. The following proposition shows that this problem is NP-hard.

Theorem 3.5: Consider a set of materialized views $\mathcal{V}$ and a query Q, the problem of selecting the smallest set $\mathcal{V}'$, such that $\mathcal{V}' \subseteq \mathcal{V}$, and $\bigcup(V \mid V \in \mathcal{V}') = \bigcup(V \mid V \in \mathcal{V}) \cap Q$, is NP-hard. ■

The proof is omitted due to space limitation [2]. The main idea is to prove that its corresponding decision problem is NP-hard by reduction from the set cover problem.

We propose a polynomial time greedy algorithm to select relevant materialized views for a given query Q from a set of materialized views $\mathcal{V}$. Initially, all keywords in Q are not covered. For each view $V \in \mathcal{V}$, $V \subseteq Q$, we record the number of uncovered keywords in Q that can be covered by V as $V.cover$, which is initialized to be $|V|$. At each step, the algorithm chooses a materialized subquery of Q that contains the largest number of uncovered keywords in Q, that is, $V \in \mathcal{V}, V \subseteq Q$, and $V.cover$ is the maximal. Then for each previously uncovered keyword k that is now covered by V, we find each view that contains k and decrease its cover by one, that is, $\forall V' \in \mathcal{V}$, such that $k \in V'$, we have $V'.cover--$. The procedure continues till all the keywords in Q are covered, or none of the views in $\mathcal{V}$ can provide additional cover. Then we compute the results of Q based on Corollary 3.2.

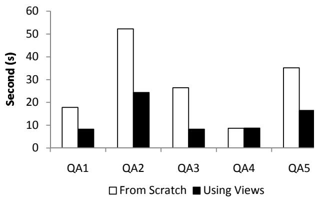
During each step for view selection, we may find a query and its subqueries, all of which are relevant views and have the same $cover$ value. We show that choosing superqueries can reduce the size of intermediate results.

Proposition 3.6: Consider Q and its subquery Q', $Q' \subseteq Q$. We have $|SLCA(\mathcal{D}, Q)| \leq |SLCA(\mathcal{D}, Q')|$ for any data $\mathcal{D}$. ■

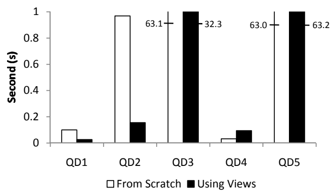
This algorithm has time complexity $O(|\mathcal{V}||Q|^2)$ and approximation ratio ln$|Q|$.

Example 3.7: Continuing Example 3.4, Initially, the relevant views of Q are found to be Q_1, Q_2, Q_3 and Q_4. Since Q_2 covers 3 keywords which is the largest among all of them, we first choose Q_2. Now the remaining keywords are D and E; Q_1 covers neither, each of Q_3 and Q_4 covers one keyword: D. Although Q_3 and Q_4 cover the same number of keywords, since Q_3 is a subquery of Q_4, we choose Q_4 for smaller intermediate results. Now, E is the only uncovered keyword, and no view covers it, so we stop. To compute $SLCA(\mathcal{D}, Q)$, we first have $SLCA(\mathcal{D}, Q) = SLCA(\mathcal{D}, SLCA(\mathcal{D}, Q_2), SLCA(\mathcal{D}, Q_4))$. Then we access the data for the matches to keyword E, and update $SLCA(\mathcal{D}$, $Q)$ = $SLCA(\mathcal{D}$, $SLCA(\mathcal{D}$, $Q)$, matches to $E)$. ■

[2]All the proofs and detailed algorithms can be found in [3].



(a) Auction (160MB)



(b) DBLP (66MB)

Fig. 3. Processing Time of Exploiting Views to Evaluate New Queries

IV. Experiments

To evaluate our approach, we compare the processing time of answering queries using materialized views and possibly a small portion of the data with that of answering queries from scratch. Some experiment results are shown in Figure 3. Two data sets, Auction generated by XMark, and a part of DBLP, are tested. Each data set has five materialized views generated. We test five keyword queries for a data set.

We can see that in general leveraging materialized views achieves significant benefits in processing time, compared with computing query results from scratch. For $QA4$, $QD4$ and $QD5$, the two approaches have roughly the same processing time, as there is no materialized view that is a subquery of $QA4$, $QD4$ and $QD5$, and we have to compute them from scratch. The extra processing time in our approach for identifying a small subset of relevant materialized views is very small.

V. Conclusions

This paper addresses an open problem of answering XML keyword search using materialized views. We adopt the query result definition using the concept of SLCA proposed in the literature [6]. We identify the relevant materialized views for a given query, and develop an algorithm to find a small set of relevant views that can answer a query. Finally, we present how to answer the query using such a set of views. Experimental evaluation shows significant performance improvements of our approach over computing query results from scratch. Our techniques can be incorporated into any XML keyword search system that uses SLCA semantics ([4], [6], [5], [2]).

Acknowledgment

This research was supported in part by the NSF grant IIS-0740129.

References

[1] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *SIGMOD*, 2003.
[2] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava. Keyword Proximity Search in XML Trees. *TKDE*, 18(4), 2006.
[3] Z. Liu and Y. Chen. Exploiting and Maintaining Materialized Views for XML Keyword Search. Technical Report TR-07-010, Arizona State University, 2007.
[4] Z. Liu and Y. Chen. Identifying Meaningful Return Information for XML Keyword Search. In *SIGMOD*, 2007.
[5] C. Sun, C.-Y. Chan, and A. Goenka. Multiway SLCA-based Keyword Search in XML Data. In *WWW*, 2007.
[6] Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. In *SIGMOD*, 2005.