IJDCST @ Dec-2017, Issue- V-5, I-8, SW-05
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

Top without Keyword included Query Processor

B. Swathi[1], S.Suvarna[2],
M.Tech, Dept of CSE, Nova College of Engineering & Technology Hyderabad.
Assistant Professor, Dept of CSE, Nova College of Engineering & Technology Hyderabad.

Abstract— Profitably noticing XML catchphrase request has pulled in much research effort in the latest decade. The key parts realizing the inefficiency of existing procedures are the general ancestor emphasis (CAR) and going to trivial centers (VUN) issues. To address the CAR issue, we propose a nonexclusive best down getting ready strategy to answer a given watchword request w.r.t. LCA/SLCA/ELCA semantics. By "top-down", we suggest that we visit all typical forerunner (CA) centers in a significance in the first place, left-to-redress ask for; by "non particular", we infer that our technique is free of the inquiry semantics. To address the VUN issue, we propose to use youth center points, instead of relative center points to test the satisfiability of a center v w.r.t. the given semantics. We propose two estimations that rely upon either ordinary modified records or our as of late proposed LLists to improve the general execution. We furthermore propose a couple of computations that rely upon hash interest to enhance the operation of finding CA center points from all included LLists. The exploratory results affirm the upsides of our systems as demonstrated by various evaluation estimations.

1 Introduction

XML has been effectively utilized as a part of numerous applications, for example, that in logical and business areas, as the standard organization for putting away, distributing and trading information. Contrasted and organized question dialects, for example, XPath and XQuery, watchword look is likewise picked up notoriety on XML information as it soothes clients from understanding the mind boggling inquiry dialects and the structure of the fundamental information, and has gotten much consideration because of that outcomes are not the whole archives any longer but rather settled parts. Normally, a XML archive can be displayed as a hub named tree T. For a given catchphrase inquiry Q, a few semantics have been proposed to characterize significant outcomes, for which the essential semantics is Lowest Common Ancestor. In view of LCA, the most broadly embraced question semantics are Exclusive LCA (ELCA) [2], and Smallest LCA (SLCA) [5], [7], [8], [9], [11]. SLCA characterizes a subset of LCA hubs, of which no LCA is the progenitor of some other LCA. As a correlation, ELCA tries to catch more significant outcomes, it might take some LCAs that are not SLCAs as important outcomes. Expect that for a given question Q ¼ fk1; k2… ..kmg, every watchword shows up at any rate once in the given XML report. Instinctively, to get all CA hubs of Q, our strategy takes all hubs in the arrangement of rearranged IDDewey mark records as leaf hubs of a XML tree Tv established at hub v, and checks whether every hub of Tv contains all watchwords of Q in a "best down" manner. The "best down" implies that if Tv contains all watchwords of Q, at that point v must be a CA hub. We at that point expel v and get a woodland Fv ¼ fTv1; Tv2 ; . . . ; Tvng of subtrees established at the n kid hubs of v. In view of Fv, we additionally locate the arrangement of subtrees FCA v Fv, where each subtree Tvi 2 FCA v contains each watchword of Q at any rate once, i.e., hub vi is a CA hub. On the off chance that FCA v ¼ ;, it implies that for Tv, just v is a CA hub, at that point we can securely avoid all hubs of Tv from being

prepared; something else, for each subtree Tvi 2 FCA v , we recursively figure its subtree set FCA vi until FCA vi ¼ ;. Give Si a chance to signify, for v, the arrangement of kid hubs that contain ki, Sca the arrangement of tyke CA hubs of v, and CA the arrangement of CA hubs in Tv. Recipe 2 implies that the arrangement of CA hubs of Q approaches the arrangement of CA hubs in Tr, where r is the report root hub. can be recursively registered by implies that for a given CA hub v, the arrangement of CA hubs in Tv is equivalent to the union of fvg and the arrangement of CA hubs in subtrees established at v's kid CA hubs, which can be additionally figured by Formula**2**

RELATED WORKS

DIL [2] sequentially processes all involved Dewey labels in document order, its performance is linear to the number of involved Dewey labels. IS [3] sequentially processes all Dewey labels of the shortest list L1 one by one. In each iteration, it picks from L1 a Dewey label l and uses it to probe other lists to get a candidate ELCA node. As the basic operations of the two algorithms are OP1 and OP2, they heavily suffer from both the CAR and VUN problems. JDewey-E [7] computes ELCA results by performing set intersection operation on all lists of each tree depth from the leaf to the root. For all lists of each level, after finding the set of common nodes, it needs to recursively delete all ancestor nodes in all lists of higher levels. As a node could be a parent node of many other CA nodes, and the deletion operation needs to process each parent-child relationship separately, JDewey-E suffers from the CAR problem. Meanwhile, as it performs set intersection on all lists of each tree depth fromthe leaf to the root, they will firstly visit nodes of V2 for Q2, thus it also suffers from the VUN problem. As some node IDs appear in many different IDDewey labels of the same inverted list, and HC [4] processes each IDDewey label of the shortest list separately, it still suffers from the CAR problem. Moreover, HC needs to push each component of every IDDewey label of the shortest inverted list into a stack, it also suffers the VUN problem when the pushed components are UNs.

3 THE BASELINE ALGORITHM

Our baseline ELCA algorithm recursively gets all CA nodes in a top-down way, then checks the satisfiability of each CA node, which works on the traditional inverted lists of labels w.r.t. Dewey or one of its variants. To do so, it needs to solve two problems: ðP1Þ identify the set of child CA nodes for each CA node v, check v's satisfiability w.r.t. ELCA semantics. For P1, given a query Q with m keywords, we know that 8i 2 ½1;m_; Sca. Thus given a CA node v and its subtree set Fv ¼ fTv1; Tv2 ; . . . ; Tvng, to get Sc, we do not need to check whether each subtree contains all query keywords; instead, we just need to check whether each node in SminvÞ, which contains least number of child nodes of v w. r.t. kmin, appears in Si 2 ½1;m_ ^ i 6¼ minÞ. Even if we know the lengths of all child lists, it's difficult to know which one is. Fortunately, as all node IDs in each child list of v are sorted in ascending order, our newly proposed set intersection algorithm guarantees that the number of processed child nodes for each CA node v is bounded by jSm. For P2, we use the following Lemma to check the satisfiability of v, which is similar to .Lemma 1. Given a query Q ¼ fk1; k2; . . . ; kmg and CA node v,

3.1 The Algorithm

Based on the above description, Algorithm 1 recursively gets all CA nodes in a top-down way. For each CA node v, it finds out the number of occurrences of each query keyword in its subtree, i.e., the length of each of its child list, then gets v's child CA nodes by intersecting v's child lists

using binary search operation. After that, it checks the satisfiability of v by Lemma 1. To do so, each inverted list Li is associated with a cursor Ci pointing to some IDDewey label of Li, Ci½x_ denotes the xth component

of the IDDewey label that Ci points to, and posðCiÞ is used to denote Ci's position in Li. Given a node v, we use lðvÞ to denote the IDDewey label of v, v:Ni denotes the number of keyword occurrences w.r.t. ki in the subtree rooted v. As shown in Algorithm 1, it firstly initializes the subtree rooted at the root node of the given XML tree in line

Algorithm 1. TDELCA($Q = \{k_1, k_2, \ldots, k_m\}$)

```
1 initialize v as the root, L_i(v) = L_i, v.N_i = |L_i(v)|,
  and C_i points to the first IDDewey label of L_i(v).
2 processCANode(v)
Procedure processCANode(v)
1 chL ← |l(v)| + 1
2 while(¬ eof(v))do
3   u ← getNextChildCA(v, chL)
4   if (u = −1) then break
5   initializeChildCA(v, chL, u)
6   v.[N_1, ..., N_m] ← v.[N_1, ..., N_m] − u.[N_1, ..., N_m]
7   processCANode(u)
8 if (∀i ∈ [1,m], v.N_i > 0) then
9   output v as an ELCA node
Function getNextChildCA(v, chL)
1 j ← 1; n ← 1; x ← argmax_i{C_i[chL]}
2 while (n < m) do
3   if (j = x) then j ← j + 1
4   use C_x[chL] as the eliminator to do binary
    search on the chL^th level of L_j(v)
5   if (pos(C_j) is out of L_j(v)) then return −1
6   if (C_x[chL] = C_j[chL]) then j ← j + 1; n ← n + 1
7   else x ← j; j ← 1; n ← 1
8 return C_x[chL]
Procedure initializeChildCA(v, chL, u)
1 for each (i ∈ [1,m]) do
2   set the start position of L_i(u) as pos(C_i)
3   binary search the end position of L_i(u) by using
    u + 1 to probe the chL^th level of L_i(v)
4   u.N_i ← |L_i(u)|
Function eof(v)
1 if (exists L_i(v), such that all nodes are processed) then
2   return TRUE
3 return FALSE
```

The procedure processCANode works as follows. Itfirstly gets the depth of v's child nodes in line 1. In lines 2-7, it repeatedly gets all child CA nodes of v. For each child CA node u got in line 3, it firstly gets the values of variables associated with v in line 5; in line 6, it excludes the occurrences of all query keywords under u from that under v. In line 7, it calls processCAN to recursively process u. After processing all childCA nodes of v, if 8i 2 ½1;m_; v:Ni > 0, according to Lemma 1, v is an ELCA node and outputted in line

4. THE HASH SEARCH BASED ALGORITHMS

Even though TDELCA-L reduces the time complexity compared with TDELCA, it relies on the probe operation (implemented by binary search) to align the cursors of inverted lists. To further improve the overall performance, we consider the existence of additional hash indexes [4], [11], [17] on inverted lists, such that each probe operation takes time without using binary search operation. the first hash table HF records the number of nodes in each Li, which is used to choose the shortest LList. For each Li, another hash table Hi records, for each node of Li, the number of its child nodes that contain ki. Note that Hi in our methods is different with that of [4], [11], [17], where Hi records, for each node v, the number of v's descendant nodes that directly contain ki., we know that the number of nodes of L1 is 11, which can be denoted as HF ½k1_ ¼ 11. According node 1 has three child nodes containing k1("Tom"), which can be denoted as H1½1_ ¼ 3. Node 5 does not have child nodes containing k1, thus H1½5_ ¼ 0. Similarly, node 3 does not contain k1, which is denoted as 3 62 H1.

4.1 The Baseline Hash Search Algorithm

Assume that jL1j _ jL2j _ __ _ _ jLmj, the main idea of our baseline hash search algorithm is: take the shortest LList L1 as the working list and recursively process all CA nodes in top-down way. For each CA node v, sequentially check whether each of its child nodes in L1 is a CA node, then output v if it is an ELCA result.

Algorithm 2. TDELCA-H($Q = \{k_1, k_2, \ldots, k_m\}$)

```
1 initialize v as the root, L_1(v) = L_1^2, and C_1 points to the first
  node of L_1(v)
2 v.[N_1, N_2, ..., N_m] ← [H_1[v], H_2[v], ..., H_k[v]]
3 processCANode(v)
Procedure processCANode(v)
1 chL ← |l(v)| + 1; N_chCA ← 0
2 while (C_1 ∈ L_1(v)) do
3    if (isCA(C_1) = TRUE) then
4       N_chCA ← N_chCA + 1
5       InitializeChildCA(C_1, L_1^chL)
6       processCANode(C_1)
7    advance(C_1)
8 if (N_chCA = 0 or ∀i ∈ [1, m], v.N_i > N_chCA) then
9    output v as an ELCA node
Function is CA(u)
1 for each (i ∈ [2, m]) do
2    if (u ∉ H_i) then return FALSE
3    u.N_i ← H_i[u]
4 return TRUE
Procedure InitializeChildCA (u, L_1^chL)
1 get L_1(u) from L_1^chL and set C_1 to the first node of L_1(u)
2 u.N_1 ← H_1[u]
```

Algorithm 2 shows the detailed description of the TDELCA-H algorithm. Compared with TDELCA and TDELCA-L, for a given query Q, TDELCA-H only needs to process all CA nodes and their child nodes in L1. For each processed node v in L1, TDELCA-H checks whether v is a CA node by hash probe operations, rather than set intersection operations on a set of child lists9 **5 CONCLUSIONS**

Considering that the key factors resulting in the inefficiency for existing XML keyword search algorithms are the CAR and VUN problems, we proposed a generictop-down processing strategy that visits all CA nodes only once, thus avoids the CAR problem. We proved that the satisfiability of a node v w.r.t. the given semantics can be determined by v's child nodes, based on which our methods avoid the VUN problem. Another salient feature is that our approach is independent of query semantics. We proposed two efficient algorithms that are based on either traditional inverted lists or our newly proposed LLists to improve the overall performance. Further, we proposed three hash search-based methods to reduce the time complexity. The experimental results demonstrate the performance advantages of our proposed methods over existing ones. One of our future work is studying disk-based index to facilitate XML keyword query processing when the size of indexes becomes too large to be completely loaded into memory.

REFERENCES

[1] S. Cohen, J.Mamou, Y. Kanza, and Y. Sagiv, "XSEarch: A semantic search engine for XML," in Proc. 29th Int. Conf. Very Large Data Bases, 2003, pp. 45–56.

[2] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "Xrank: Ranked keyword search over XML documents," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2003, pp. 16–27.

[3] Y. Xu and Y. Papakonstantinou, "Efficient LCA based keyword search in XML data," in Proc. 11th Int. Conf. Extending Database Techn.: Adv. Database Technol., 2008, pp. 535–546.

[4] R. Zhou, C. Liu, and J. Li, "Fast ELCA computation for keyword queries on XML data," in Proc. 13th Int. Conf. Extending Database Technol., 2010, pp. 549–560.

[5] Y. Xu and Y. Papakonstantinou, "Efficient keyword search for smallest LCAS in XML databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2005, pp. 537–538.

[6] Y. Li, C. Yu, and H. V. Jagadish, "Schema-free xquery," in Proc. 13th Int. Conf. Very Large Data Bases, 2004, pp. 72–83.

[7] L. J. Chen and Y. Papakonstantinou, "Supporting top-K keyword search in XML databases," in Proc. 26th Int. Conf. Data Eng., 2010, pp. 689–700.

[8] C. Sun, C. Y. Chan, and A. K. Goenka, "Multiway SLCA-based keyword search in XML data," in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 1043–1052.

[9] Z. Liu and Y. Chen, "Reasoning and identifying relevant matches for XML keyword search," J. Proc. Very Large Data Bases Endowment, vol. 1, no. 1, pp. 921–932, 2008.

[10] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective keyword search for valuable LCAS over XML

documents," in Proc. 16th ACM Conf. Conf. Inform. Knowl. Manage., 2007, pp. 31–40.

[11] W. Wang, X. Wang, and A. Zhou, "Hash-search: An efficient SLCA-based keyword search algorithm on XML documents," in Proc. 14th Int. Conf. Database Syst. Adv. Appl., 2009, pp. 496–510.

[12] Y. Chen, W. Wang, and Z. Liu, "Keyword-based search and exploration on databases," in Proc. IEEE 27th Int. Conf. Data Eng., 2011, pp. 1380–1383.

[13] B. Q. Truong, S. S. Bhowmick, C. E. Dyreson, and A. Sun, "MESSIAH: Missing element-conscious SLCA nodes search in XML data," in Proc. SIGMOD, 2013, pp. 37–48.

[14] L. Kong, R. Gilleron, and A. Lemay, "Retrieving meaningful relaxed tightest fragments for XML keyword search," in Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol., 20 pp. 815–826.

[15] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava, "Keyword proximity search in XML trees," IEEE Trans. Knowl. Data Eng., vol. 18, no. 4, pp. 525–539, 2006.

[16] J. Zhou, Z. Bao, W. Wang, T. W. Ling, Z. Chen, X. Lin, and J. Guo, "Fast SLCA and ELCA computation for XML keyword queries based on set intersection," in Proc. 28th Int. Conf. Data Eng., 2012, pp. 905–916.

[17] J. Zhou, Z. Bao, W. Wang, J. Zhao, and X. Meng, "Efficient query processing for XML keyword queries based on the idlist index," Int. J. Very Large Data Bases, vol. 23, no. 1, pp. 25–50, 2014.

AUTHOR DETAILS

STUDENT DETAILS

Miss B.Swathi Studying II M.Tech (CSE) in Nova College of Engineering & Technology Hyderabad. She completed B.Tech (IT) in 2014 in Nalgonda Institute Of Technology And Science

GUIDE DETAILS

S.Suvarna is presently working As.professor & Head, Department of computer science & Engineering in Nova College of Engineering & Technology Hyderabad. She completed M.Tech (CSE) from JNTUH. She is guided many U.G& P.G projects. She has more than 14 years of teaching experience. She published more than 4 International Journals.