International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 7, July 2017)

Top Down Approach for XML Keyword Query Processing using Disk based Index

Nikita R. Alai[1], Archana S. Vaidya[2]

[1]Student, ME(CSE), [2]Professor, Department of Computer Engineering, Gokhale Education Society's, R. H. Sapat College of Engineering Management Studies & Research, Nashik, Savitribai Phule Pune University, Maharashtra, India

***Abstract*— From the past many years there has been much scope in efficiently replying to the Extended Markup Language (XML) keyword queries. Limitations of the existing system can be listed as the common-ancestor-repetition (CAR) and visiting useless-nodes (VUN) issues. In order to solve the CAR problem, we hereby introduce a generic top-down strategy to answer a given keyword query. The meaning of this top-down method is that we will be visiting all the common ancestor (CA) nodes in approaches like depth-wise, left-to-right and by introducing generic approach it can be concluded that our implementation is not related to query semantics. So as to solve the issue of VUN, we implement to make use of child nodes, instead of descendant. We would be showing that for the purpose of faster document retrieval, the usage of tree is better than usage of array. Algorithms like LList based, Hash search based are performed for improved performance. We are using disk based index approach to reduce memory load. The advantage of the tree data structure over traditional array data structure is the searching process is more efficient and time saving.**

***Keywords*— XML, CAR, VUN, keyword query, disk based index, XML keyword.**

I. INTRODUCTION

XML queries are used to extract data and manipulate data from XML documents. By specifying the predicate the related attributes or elements are selected using XML queries. An attribute, a value or an element tag can be represented by node in tree based structure. Edges are used to represent hierarchical relationship like parent-child and ancestor descendant among XML [1]. Conditions are represented by edges and nodes. Conditions must be satisfied in XPath expression as a response of a query. XML query processing is dependent on traditional top-down tree traversals on the XML document which is highly inefficient as a large collection of documents is produced. To solve searching problem and to store XML data various indexing techniques are used. For reduction of memory overload of the processing queries indexing approach is introduced which is use of disk based indexing approach. In this approach a specific data structure is used for storing the index values.

In the XML query processing the common issues which lead to redundancy and inconsistency are CAR and VUN problem.

CAR problem: In graph theory the lowest common ancestor of two nodes v and w in a tree T is the lowest i.e. deepest node that has both v and w as descendants, where each node to be descendant to itself. While multiple operations results in all common ancestors on the path from root to visiting nodes to be repeatedly visited, which is called as common ancestor repetition (CAR).

VUN problem: Given a keyword query Q and XML document D, let v be the set of nodes D that contains one keyword query in their sub trees then we can classify them into following categories:

(1) Common ancestors (CAs)
(2) Useless nodes (UNs)
(3) Auxiliary nodes (AUs).

By considering these problems we demonstrated query semantics with the generic processing strategy for solving above problems more effectively and efficiently as:

In order to solve CAR problem a generic top down approach for XML keyword query processing is demonstrated. To address VUN problem, child nodes are used instead of descendants to test lowest common ancestors (LCA), smallest lowest common ancestors (SLCA), exclusive lowest common ancestor (ELCA) nodes as well as labeling scheme independent inverted list (LList) algorithm is demonstrated. Performance is improved by means of using hash index. We introduced distributed query processing, in which the index structure provides basic means to reconstruct and locate distributed XML fragments.

II. LITERATURE REVIEW

The key factors which results in the inefficiency for the XML keyword search algorithms were CAR and VUN problems. Junfeng Zhou et al. [1] proposed genetic top down processing strategy for visiting all common ancestor nodes only once which avoided CAR problem.

An independent query semantic approach proved satisfiability to avoid VUN problem. They proposed two algorithms namely LList to improve performance and hash search based method for reducing time complexity. The shortcoming in their system was memory overload of the index size while performing query processing on the XML data. In XML document management, XML databases uses specific encoding mechanism which maps hierarchical structure of the document into a flat representation. To support query workload various encoding techniques had been proposed. In XML documents processing an atomic updates is quite costly. To address this issue Lukas Kircher et al. proposed a technique named as structural bulk updates which worked with XQuery Update Facility (XQUF) to support efficient updates. XQUF [2] did not put node to the list. Mukesh. K. Agarwal and K. Ramamritham presented a system known as generic keyword search (GKS) [3] over XML data. With the help of XML data and query most relevant data keywords as well as schema elements are found by using XML node ranking method. GSK did not work on raw XML data.

Keyword search diversification model of contexts were measured by exploring relevance to the original query by J. Li et al. [4]. The problem in this system is effectiveness because the comparison of results became difficult when contents of the results were not informative. The problems of keyword query over error tolerant knowledge bases are solved by Yu-Rong Cheng et al. They proposed a r-clique method [5] which returns reasonable answers to the user. They also provided filtering and verification framework for calculating the answers efficiently. For general purpose query Da Yan et al. [6] developed a distributed system known as Quegel used for big graphs. This was a general purpose system that was applied on graph indexing to speed up query processing in distributed environment. With the help of shortest path queries, graph keyword queries and point to point each-ability queries good performance had been achieved. Jing Wang et al. [7] presented a solution for the graph querying problem. This solution worked in three steps: First, it constructed indexes for queries not to rely on database graph index. Second, it maintained knowledge log of the system produced while executing the queries. Third, it used sub-graph as well as super-graph. They proposed iGQ framework consisting of sub-graph index, super-graph index and developed a method which maintained the index of graph replacement policy.

In distributed database designing two major approaches are used such as:

1) Bottom-up approach
2) Top-down approach

Databases are growing rapidly in size. To design system widely used strategy is top-down approach. Ajay B. Gadicha et al. explained multiple design strategies for distributed database such as Single Query Multiple Database (SQMD) [8]. This architecture performs parallel operations on more than one database by using single query which gives clear idea about design strategy. Jeremy Barbay et al. [9] introduced an algorithm named as small adaptive interpolation algorithm for faster search of results. The experiments showed that the interpolation algorithm performed better than other intersection algorithms such as sequential algorithm. Yi Chen et al. [10] discussed keyword search techniques, query result definition, result generation by using query processing, optimization of performance and quality search evaluation.

To retrieve inverted index which consists of keywords and to identify documents query processing is done. Dimitris Tsirogiannis et al. [11] presented an algorithm to calculate an arbitrary number for both sorted and unsorted list named as intersection algorithm. Vishwakarma Singh et al. studied queries which satisfy given set of keywords of the tightest groups. A novel method known as ProMiSH (Projection and Multiscale Hashing) used for achieving high scalability and speedup the performance using random projection and hash based index structure. An algorithm for finding out top k tightest clusters in subset which retrieves the points from disk using B+ tree for exploration of final set of result. The results on real as well as synthetic data showed that ProMiSH [12] up to 60 times of speed up over tree based techniques. In order to improvise the scalability Evandrino G. Barros et al. introduced PMKStream (Parallel MKStream) for evaluation of multiple keyword queries of multiple parsing stacks. The results showed that PMK [13] Stream was efficient for supporting keyword based search over XML data. Indexing techniques are used for speed up the data retrieval rate. A. John et al. studied various approaches used for minimization of the history data. Update on change and sampling are two approaches which are based on spatio-temporal indexing method. Data is represented in two types: certain data (constant value) and uncertain data (inexact data). Both this data types had its indexing technique based on which tree structure is used. Main indexing techniques are HBase index, Threshold interval index, External interval tree index, U-Grid, PTI index, MON tree, LGU tree, Gauss tree, Segment based index, FUR tree, RUM tree [14].

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 7, July 2017)

Guimei Liu et al. [15] studied three structures for indexing as well as querying frequent item sets:

1) Signature files
2) Inverted files
3) CFP tree.

Experimental result showed that no structure can outperform other structure also CFP tree showed better performance than other two techniques. To address the SLCA computation problem in XML data Ba Quan Truong et al. [17] proposed a property called optionality resilience which specified behaviors of an XKS for queries with missing elements. The experimental results showed quality of search, execution time, scalability, number of missing elements, number of keywords and heuristics for algorithm selection. It also showed that MESSIAH not only produced high quality result but also faster computation speed.

Prefix based numbering (PBN) was proposed by Curtis E. Dyreson et al. [18] which is a popular method for numbering nodes in the hierarchy. They presented a strategy to virtually transform the data without renumbering and instantiating. The result was concise, support efficient querying, updating was efficient and practical. A user query is a set of keywords which match with labels or values of nodes in XML trees. Khanh Nguyen and Jinli Cao [19] introduced novel approach known as Relevant LCA (RLCA) to accurately and efficiently capture relevant fragments to XML keyword search. Experimental results showed the effectiveness of RLCA and carefully measured precision, recall and F-measure which achieved high effectiveness.

Nikita Alai and A. S. Vaidya [16] studied about XML data and keyword, indexing techniques and query processing. In this review various algorithm regarding indexing, XML data and processing of query are studied. Key factors which results in inefficiency of XML keyword search considering algorithms are CAR and VUN problems. These problems are solved by using generic top down strategy and use of child nodes. To reduce memory overload size of index became too large. For which we proposes disk based index approach which can reduce memory overload and improve the performance of the XML keyword search for the query processing. Prefix based numbering (PBN) [18] was proposed by Curtis E. Dyreson et al. which is a popular method for numbering nodes in the hierarchy. They presented a strategy to virtually transform the data without renumbering and instantiating. The result was concise, support efficient querying, updating was efficient and practical.

III. Proposed System

A. Problem Statement

To design a system on XML keyword query processing which is used for faster retrieval of relevant document by using disk based index approach.

B. System Architecture

The following figure represents overall architecture of system.

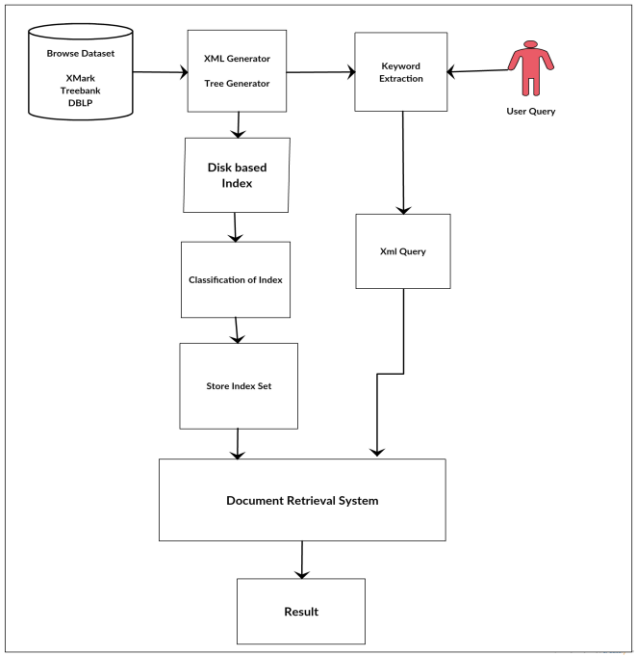

Fig.1. Architecture of Top Down Approach for XML Keyword Query Processing using Disk based Index

In the proposed system architecture, we represents query processing with the help of XML document [20], indexing techniques such as LList based algorithm, hash search algorithm for generating keywords. XML queries are generally used to select relevant elements or attributes by specifying predicate. In a tree-based representation, nodes can represent an element tag, an attribute or a value. Hierarchical relationships such as ancestor descendant and parent-child among XML elements can be represented with the help of edges. XML query processing depends upon the traditional top-down or bottom up traversing of tree on the XML document. It is highly inefficient, because it produces large collection of documents. To reduce the overhead of processing queries, indexing or labeling methods are efficiently used. Our system works as follows:

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 7, July 2017)

User fires a query on specified data-sets, Eg. DBLP [23], XMark [21], TreeBank [22]. By using user query and data-set the XML generator will generate the tree which will be used for keyword extraction. By using keyword extraction module XML query will be generated and this XML query is used for document search. Searching is done by using disk based index. This request is send to the index storage set and with the help of matching keyword the document is retrieved efficiently. Indexes are classified. For searching the document, key value is used which works on disk based index. Index storage set helps to retrieve document efficiently than traditional searching. Hence to reduce memory overload and improve performance disk based index is used.

C. *Algorithms and its analysis*

Algorithm 1: The LLIST-Based Algorithm
Input:
Parsed data-set
Output:
Array of index (key)
Steps:
1. Initialize v as root node
2. Assign id values to nodes
3. Process CANode
4. Call function getNextChild CA(v, chl)
5. Each node and its child list are sorted as follows:
6) Assign j=1, n=1, x=argmax,
7) while(n is less than m) do
8) if(j=x) then j=j+1;
9) use Cx as eleminator to do binary search
10) if (pos(Cj) is out of Lj(v) then return -1
11) if (Cx[child] = Cj[child]) then j=j+1; n=n+1;
12) else x=j; j=1; n=1;
13) return Cx[child]

Analysis:

This algorithm takes parsed data-set as an input. Algorithm needs to perform sorting operation by using inverted list of nodes. Child list is one of the important factor which affects the interval time of the searching. In searching operation, each node needs to be placed into the array of an index. Whether the node is a root node or its child node, it is added to the array of an index which is further used as a key value for the processing. The algorithm guarantees that for any node given as an input, it can be computed, sorted and added to the array of an index. By using LList Based Algorithm we can reduce calling time for sorting and cost of the operation.

For each node from different data-sets or for each node from same data-set the array of an index created is differ from each iteration so the array of index is different for each node. That is why the algorithm is NP-Hard.

Algorithm 2: The Baseline Hash Search Algorithm

Input:
Key (index value of an array)
Output:
Tree (contents of root node)
Steps:
1. Initialize v as root node
2. Assign index value to root node
3. Process CANode
4. Call function CA - addtoroot() or addtoparent() or addtochild()
5. Initialize child CA(u, L_1^{chL})
6) get L1(u) from L1[child] and set C1 to the first node of L1(u)
7) u.N1=H1[u]
Analysis:
To improve overall performance and to reduce memory overload hash index are used. The algorithm is NP-Hard because input given to it is a key value from the array of index. This key is processed by an algorithm and contents of that key are displayed. With the indexing and top down approach retrieval of contents is faster than traditional searching.

D. Data-sets

The experiments were run by using following data-sets. Results can be verified on each data-set.

TABLE I
DATA SET DETAILS

Dataset	Size	Details
XMark	11.3 MB	Total no. of records 1001 Attributes are id, location, name, quantity, payment, description, shipping, to, from, message.
Treebank	85.4 MB	Total no of records 1000 Attributes are NP, VBP, VG, TO, VBG.
DBLP	1.78 GB	Attributes are key, author, title, pages, year, volume, url.

The following table shows the comparison of dataset wrt. index size.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 7, July 2017)

TABLE III
DATA SET DETAILS

Dataset	Without index size	With index size
XMark	11.3 MB	12 MB
Treebank	85.4 MB	87 MB
DBLP	1.78 GB	2 GB

IV. System Analysis

A. Mathematical Modeling

Let S be a XML query processing such that
S = {S0, S5, Fs, D, C, T, Q, K, I, R| ϕs}
Where,
D represents set of Dataset
C represents set of Classes
T represents set of Trees
Q represents set of Queries
K represents set of Keywords
I represent set of disk Index
R represents set of Results
Initial State (S0):
User wants to fire a query.
End State (S5):
User obtains relevant result.
Input:
User Query
Output:
Relevant Document
Functions (Fs): {f1, f2, f3, f4, f5, f6, f7}
f1 = parsing of dataset
f2 = keyword extraction
f3 = generation of tree
f4 = XML query generation
f5 = creation of index
f6 = classification of index
f7 = obtaining result

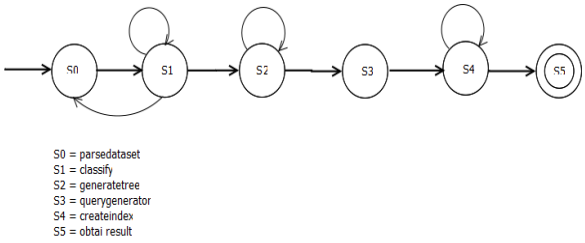

Fig.2. State Transition Diagram

B. Implementation Details

Experiments are carried out on personal computer having minimum 4 GB of memory, 1 TB hard disk, and operating system with web browser such as Firefox or Google Chrome. Software requirements are JAVA (JDK 1.6) for programming, software development tool net beans IDE and Apache Tomcat Server.

V. Result And Discussion

The fig. 3 shows output of Hash search algorithm which is tree structure of given input value i.e. key value. The experiments carried out on XMark dataset. For each key value given input to the algorithm the tree structure generated is different. In fig. 3 we are getting structure for root node which is 'site'. Parent node 'regions' and its child node 'africa' and so on. For each node the structure of tree varies. Documents which are retrieved after experiments are relevant or not has measured. Relevant documents in this scenario are the documents which contain the keyword which are ask by the user in the form of query.

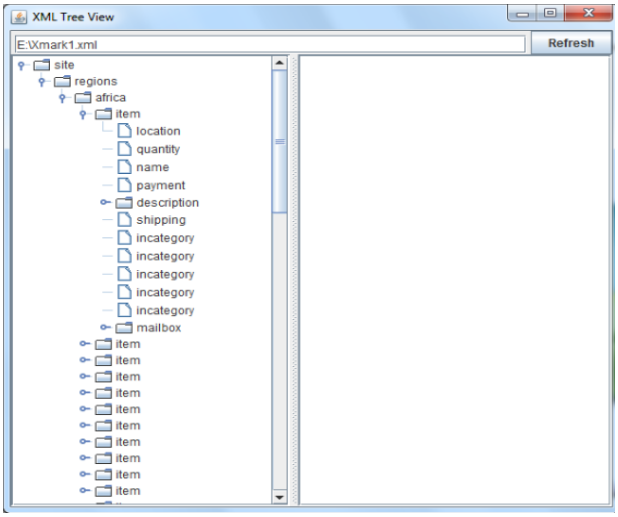

Fig.3. Obtained result for XMark

Performance of the system is measured on the basis of following parameters:

- *Precision:* The number of documents retrieved which are relevant to the query fired by user.
- *Recall:* The number of documents which are relevant to the query that are successfully retrieved.
- *Accuracy:* It is the closeness of a measurement to the true value.
- *Throughput*: It is the maximum rate at which query can be processed.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 7, July 2017)

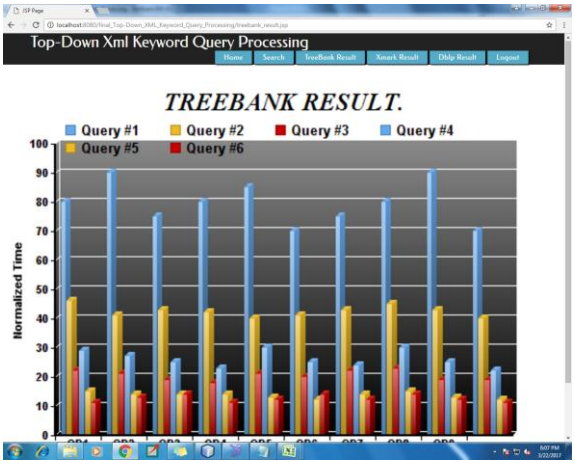

Fig.4. Time Calculation of query processing on Treebank with disk based index

The above result shows graph for Treebank dataset. For different queries that are fired, the time taken for query processing is shown where six different servers are used. For query QD1, the time taken for query processing is 100ms for server 1, 45ms for server 2, 25ms for server 3, 30ms for server 4, 15ms for server 5, 12ms for server 6. Similar results for all the nine queries and their processing time is shown in Fig. 4. The following result shows graph for XMark dataset. For different queries that are fired, the time taken for query processing is shown where six different servers are used. For query QD1, the time taken for query processing is 100ms for server 1, 40ms for server 2, 25ms for server 3, 25ms for server 4, 05ms for server 5, 8ms for server 6. Similar results for all the nine queries and their processing time is shown in Fig. 5.

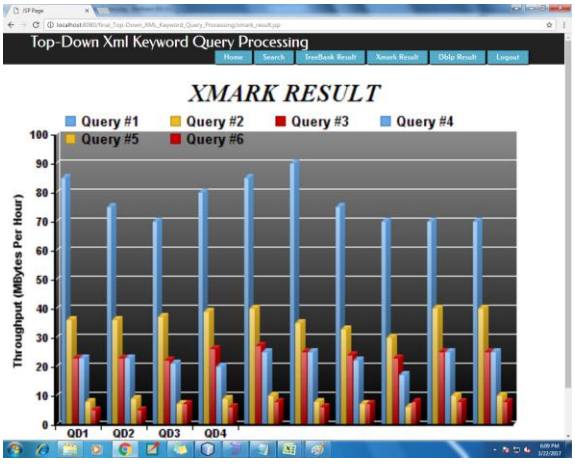

Fig.5. Time Calculation of query processing on XMark with disk based index

The Fig. 6 shows the graphical representation of the execution time required and shows a comparison for existing and proposed system. We have tested the running time required for the process to carry out its execution. It can be seen that XMark takes 30ms time to execute the project by using the existing system while our proposed system takes 15ms for its execution. Similar processing time can be seen for Treebank and DBLP. We can conclude that the time required for executing the process for existing system is comparatively low than the time required for proposed system.

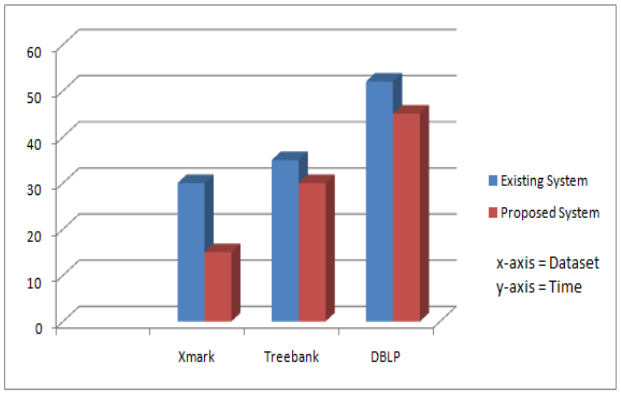

Fig. 6 Existing system vs Proposed system

VI. CONCLUSION

Key factors which results in inefficiency for existing XML keyword search considering algorithms are CAR and VUN problems. These problems are solved by using generic top down processing strategy and use of child nodes provides satisfiability. For query semantics independent approach is used where efficient algorithms are used such as LList index and Hash search based methods which reduce time complexity. To reduce memory overload size of index became too large. For which we proposed disk based index approach which reduces memory overload and improve the performance of the XML keyword search for the query processing. We demonstrated distributed query processing which provide the basic means to locate and reconstruct distributed XML fragments.

Acknowledgment

I wish to express my sincere thanks and deep sense of gratitude to respected guide Prof. A. S. Vaidya in Department of Computer of Gokhale Education Society's R. H. Sapat College of Engineering Management Studies & Research, Nashik for encouragement, technical guideline, advice and constructive criticism which motivated to strive harder for excellence. I also wish to acknowledge to the people who gives their support directly or indirectly to paper writing.

REFERENCES

[1] Junfeng Zhou, Wei Wang, Ziyang Chen and Jeffrey Xu Yu, "Top-Down XML Keyword Query Processing",in IEEE Transactions on Knowledge and Data Engineening,Volume:28,Issue: 5, May 1 2016,pp. 1340-1353.

[2] L. Kircher, M. Grossniklaus, C. Grun, and M. H. Scholl, "Efficient structural bulk updates on the pre/dist/size XML encoding", in Proc. IEEE 31st Int. Conf. Data Eng., 2015, pp. 447-458.

[3] M. K. Agarwal and K. Ramamritham, "Enabling generic keyword search over raw XML data", in Proc. 31st Int. Conf. Data Eng., 2015, pp. 1496-1499.

[4] J. Li, C. Liu, and J. X. Yu, "Context-based diversification for keyword queries over XML data",in IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 3, Mar. 2015, pp. 660-672.

[5] Yu-Rong Cheng, Ye Yuan, Jia-Yu Li, Lei Chen and Guo-Ren Wang, "Keyword Query over Error-Tolerant Knowledge Bases",in Journal of Computer Science and Technology, DOI. 10.1007/s11390-016-1658-y, July 2016, pp. 702-719.

[6] Da Yan, James Cheng, Fan Yang, Yi Lu, John C. S. Lui, Qizhen Zhang and Wilfred Ng, "A General Purpose Query Centric Framework for Querying Big Graphs", in Proceedings of the VLDB Endowment, Vol.9, No. 7, 2016, pp. 564-575.

[7] Jing Wang, Nikos Ntarmos and Peter Triantafillou, "Indexing Query Graphs to Speedup Graph Query Processing", in Proc. 19th International Conference on Extending Database Technology (EDBT), March 15-18, 2016, ISBN 978-3-89318-070-7.

[8] Ajay B. Gadicha, A. S. Alvi, Vijay B. Gadicha and S. M. Zaki, "Top-Down Approach Process Built on Conceptual Design to Physical Design Using LIS, GCS Schema", in International Journal of Engineering Sciences & Emerging Technologies, Volume 3, Issue 1, August 2012, pp. 90-96.

[9] Jeremy Barbay, Alejandro Lopez-Ortiz and Tyler Lu, "Faster Adaptive Set Intersections for Text Searching", in Proceeding WEA'06 Proceedings of the 5th international conference on Experimental Algorithms, May 24-27, 2006, pp. 146-157.

[10] Yi Chen, Wei Wang and Ziyang Liu, "Keyword-based search and exploration on databases", in 2011 IEEE 27th International Conference on Data Engineering, DOI. 10.1109/ICDE.2011.5767958, 16 May 2011, pp. 1380-1383.

[11] Dimitris Tsirogiannis, Sudipto Guha and Nick Koudas, "Improving the Performance of List Intersection", in Journal Proceedings of the VLDB Endowment, Volume 2, Issue 1, August 2009,pp. 838-849.

[12] Vishwakarma Singh, Bo Zong, and Ambuj K. Singh, "Nearest Keyword Set Search in Multi-Dimensional Datasets", in IEEE Transactions On Knowledge And Data Engineering, Vol. 28, No. 3, March 2016, pp. 741-755.

[13] Evandrino G. Barros, Fernando G. D. C. Ferreira and Alberto H. F. Laender, "Parallelizing Multiple Keyword Queries over XML Streams", in Data Engineering Workshops (ICDEW), 2016 IEEE 32nd International Conference, June 2016, pp. 1-4.

[14] A John, M Sugumaran, and RS Rajesh, "Indexing And Query Processing Techniques In Spatio-Temporal Data", in Ictact Journal On Soft Computing, Volume 06, Issue 03, April 2016, pp. 1198-1217.

[15] Guimei Liu, Andre Suchitra, and Limsoon Wong, "A performance study of three disk-based structures for indexing and querying frequent itemsets", in Proceedings of the VLDB Endowment, Vol. 6, No. 7, May 2013, pp. 505-516.

[16] Nikita R. Alai and A.S. Vaidya, "A Survey: Top-Down Xml Keyword Query Processing", in Global Journal of Advanced Engineering Technologies (GJAET), Volume 5, Issue 4- 2016, ISSN (Online): 2277-6370 & ISSN (Print):2394-0921, pp. 426-430.

[17] 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.

[18] C. E. Dyreson, S. S. Bhowmick and R. Grapp, "Querying virtual hierarchies using virtual prefix-based numbers", in Proc. ACM SIGMOD International Conference Manage. Data, 2014, pp. 791-802.

[19] Khanh Nguyen and Jinli Cao, "Exploit Keyword Query Semantics and Structure of Data for Effective XML Keyword Search", in Proc. 21st Australasian Database Conference, Brisbane, Australia, Volume 104, January 2010, pp. 133-140.

[20] J. Lu, T. W. Ling, C. Y. Chan and T. Chen, "From region encoding to extended dewey: On efficient processing of XML twig pattern matching", in Proc. 31st Int. Conf. Very Large Data Base, 2005, pp. 193-204.

[21] http://www.grappa.univlille3.fr/~champavere/Recherche/datasets/

[22] http://www.anc.org/data/masc/downloads/datadownload/

[23] http://dblp.uni-trier.de/xml/

Authors profile

Nikita R. Alai, ME, Second year student, Department of Computer Engineering, GES's, R. H. Sapat College of Engineering Management Studies & Research, Nashik-5.

Prof. Archana S. Vaidya , Assistant Professor, Department of Computer Engineering, GES's, R. H. Sapat College of Engineering Management Studies & Research, Nashik-5.