Journal of Computer Science 8 (4): 545-550, 2012
ISSN 1549-3636

Similarity Based Clustering with Indexing for Semi-Structured Document

[1]Palanisamy, S. and [2]K. Baskaran
[1]Department of Computer Science and Engineering,
Institute of Road and Transport Technology, Erode, India
[2]Department of Computer Science and Engineering,
Government College of Technology, Coimbatore, India

Abstract: Problem statement: To improve the performance of data retrieval in a homogeneous large XML document. **Approach:** Clustering of XML elements based on the content with indexing. The element which is used for clustering has been identified from the document and/or XML schema. This element is used as a parameter for clustering. The suitable index is created after clustering. **Results:** The clustering combined with indexing strategy support the efficient retrieval of XML element from the document. **Conclusion:** The proposed method is used to improve the efficiency of XML data manipulation and comparatively give the better performance rather than clustering or indexing alone.

Key words: Clustering, indexing, XML, query

INTRODUCTION

The evolution of Internet and communication technologies supported the business enterprises to get more benefits. Business partners across the world communicated via Internet and shared their business information rapidly and easily. The complexity of application is increased and data management in Internet becomes a tedious task. To handle these complexities new techniques and methods are constantly proposed and implemented. XML is a simple language to describe and exchange data across Internet. Data in the form of semi-structure become inevitable in business databases. These semi structured data cannot be easily handled in relational database management systems. The ability to manage structured data in a flexible manner is one of the fundamental differences between XML and relational databases. Semi-structured data does not have rigid structure and the structure can be frequently changed. Data with no rigid structure can be easily represented in the form of XML data. Several new technologies such as Xpath, Xquery, Document Type Definition (DTD), XML schema and Xlink are available to manipulate the semi-structured data in the form of XML. Performance is the one of the important criteria for any new technology or methods. In this study the performance of proposed similarity based clustering method is analyzed on SQL Sever2005 database engine and oracle Berkely Database XML (BDXML).

Literature survey: some of the existing works in the area of clustering XML documents are given, stressing the fact that any of the existing work does not deal with efficiently clustering the elements of single large XML document. Clustering is an intelligent technique for mining XML documents. It has been utilized as an excellent way of grouping the documents by their content or structure. A lot of efforts have been taken on how to cluster XML documents effectively with structural (Nierman and Jagadish, 2002; Dalamagas *et al*., 2006; Leung *et al*., 2005a; Hwang and Ryu, 2010) or semantic (Lee *et al*., 2001; Nayak and Iryadi, 2007; Tagarelli and Greco, 2004; Kim *et al*., 2008) information. Hierarchical algorithms (Lian *et al*., 2004) are based on structural information present in the data. The notion structure graph is specified, supporting a computationally efficient distance metric defined between documents and set of documents. The simple metric yields new clustering algorithm, which is efficient and effective, compared to other approaches. A clustering based on path pattern is presented (Leung *et al*., 2005b). It is a method of XML structural representation called Common XPath (CXP), which encodes the frequently occurring elements with the hierarchical information and proposed to take the CXPs mined to form the feature vectors for XML document clustering. Distance based clustering of XML documents (Francesca *et al*., 2003) it focus on the notion of XML cluster representative, it is a prototype XML document subsuming the most relevant features

Corresponding Author: Palanisamy, S., Department of Computer Science and Engineering,
Institute of Road and Transport Technology, Erode, India

of the set of XML documents with in the cluster. Dynamic XML documents clustering (Rusu *et al*., 2008) are established an intelligent and efficient technique to reassess the distance between dynamic XML documents when one or all of the initially clustered documents have changed. It allows the user to reassess the pair-wise XML document distances, not fully comparing each new pair of versions in the clustering solution, but by determining the effect of temporal changes on the previously known distances between them. It is both time and I/O effective, as the number of operations involved in distance reassessing is reduced. A clustering model (Yang *et al*., 2005) is developed for representing XML documents. The term semantics, element similarity, as well as elements relative importance for a given set of documents can all be taken in to account. It is also formulated an iterative estimation procedure for automatically learning an element similarity matrix associated to this model. The structural similarity between a pair of XML documents can thus be computed based on different edit distances (Zhang *et al*., 2003; Nierman and Jagadish, 2002) which differ from each others in terms of the set of allowed edit operators and their support for repetitive and optional XML elements. It has been proved in (Zhang *et al*., 1992) that computing the edit distance for unordered labeled trees is NP-Complete and yet the distance is not optimal in any sense related to the elements semantics. None of the existing algorithms are used to cluster the elements of large XML document based on similarity. Similar elements are moved into a closure places. It is quite nature that similar elements may be accessed for computing summarization or manipulation as a group. It is difficult for the system to identify the similarity between the elements directly. It is not possible to build the system to automatically detect the similarity between elements.

MATERIALS AND METHODS

Similarity based clustering algorithm: The basis for similarity based clustering algorithm is defined the same type of elements present in the XML document. In this method, our human expertise is used to identify and specify the similar elements present in the document. The entire XML data and schema of the data are displayed to the user. Based on his expertise user can specify the element which is used to cluster the entire XML document into clusters. Similar elements present in the XML documents are grouped. The number of clusters is computed based on the different data values of XML elements. Figure 1 shows the basic steps of clustering algorithm.

```
Clustering algorithm using similarity
input(sc,xd)  sc – schema, xd – xml data file
output(cd) clustered xmldata
procedure similar_cluster(sc,xd)
begin
if schema exists scan(sc)
/* user views the entire schema */
scan(xd)
/* user views the entire xml data */
input(param)
/* user specify the parameter based on which
clustering will be done */
count=0
for each element in (xd)
read param value
if param value is distinct
count=count+1
/* compute the number of clusters to be
created*/
create cluster(cd,count)
for each element in (xd)
move(element,cd)
/* move the element to the corresponding
clusters*/
return(cd) /* return the clustered XML data */
end
```

Fig. 1: Pseudo code for clustering algorithm

Experimental setup: Similarity based clustering algorithm is implemented in Java and all experiments were run on a PC with a 2.66 GHz Intel core 2 Duo processor, 250 GB SATA HDD and 2GB DDR RAM with windows XP operating system environment. The SQLserver2005 and the open source software tool BDBXML are used as a database engine to store and manipulate XML data.

XML indexes: One major advantage of modern native XML databases is their ability to index the XML documents they contain. Proper use of indices can significantly reduce the time required to execute a particular Xquery expression. If no index exists, SQL Server 2005 evaluating each and every element in the table against the query. If index, exits SQLServer2005 can find a subset of matching documents with a single or significantly reduced set of lookups. By carefully applying XML indexing strategies, the retrieval performance can be improved considerably. Indices are specified in four parts: path type, node type, key type and uniqueness. The Fig. 2 shows the time required to access the elements of XML document stored in a table using index and without index. It shows that indexing reduces the time required to retrieve elements of XML document from the table.

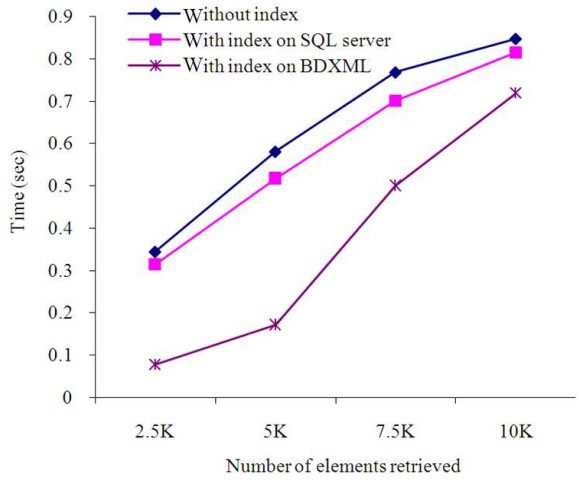

Fig. 2: Effect of indexing

```
create table empc([Sno] [int] not null,[Empinfo] [xml]
not null)

create unique index ui_emp on empc(Sno);

create fulltext catalog ft as default;

create fulltext index on empc(Empinfo) key index
ui_emp;

Insert into empc (Sno, Empinfo) values (1,XMLdata)
```

Fig. 3: Data generation SQL query

SQL server 2005: The Microsoft SQL Server 2005 database engine is the core service for storing, processing and securing data. The database engine provides controlled access and rapid transaction processing to meet the requirements of the most demanding data consuming applications within the enterprise. The database engine also provides rich support for sustaining high availability. In SQL Server 2005 the xml data type allows to store XML documents and fragments in SQL Server database. An XML fragment is an XML instance that is missing a single top-level element. In a table XML data type can be used to create columns and variables that store XML instances in them. Note that the stored representation of XML data type instances cannot exceed 2GB. XML instances are stored in the XML type columns as Binary Large Objects (BLOBs). These XML instances can be large and the stored binary representation. Without an index, these binary large objects are shredded at run time to evaluate a query that can be time-consuming. Indexes can reduce the amount of data that must be read to return the query result set. The XML indexes fall into two categories: Primary XML index, Secondary XML index. The first index on the XML type column must be the primary XML index. Using the primary XML index, three types of secondary indexes are supported. These include PATH, VALUE and PROPERTY. Depending on the type of queries, these secondary indexes may help improve query performance. Figure 3 show the query for the generation of sample XML document in SQL Server 2005.

Berkeley database XML: BDBXML an embedded XML database engine that provides support for Xquery access. In BDBXML all XML data are stored within files called containers. The BDBXML shell provides a simple and convenient way to work these containers and exposes most of the BDBXML functionality in a friendly, interactive environment. Containers also store XML documents as either whole documents or as nodes. When containers stores whole documents, the XML documents are stored as all one unit in the containers exactly as it was presented to the system. When documents are stored as nodes, the XML document is deconstructed into smaller pieces-nodes and those small chunks are stored in the container. Node storage offer better performance than does document storage and for this reason node storage is the default container type.

RESULTS

Performance analysis SQL server 2005: The similarity based clustering algorithm, is tested using several thousand XML elements stored in a document. Initially database named empfull.mdb is created. In this database, table named empc is created with the following fields Sno (int) and Empinfo (xml). The XML column Empinfo contains name, deaprtment, designation and address as elements. Ten thousand well formed XML elements are inserted into the table. From the table queries can be used to retrieve the data. For example without indexing and clustering the following queries used to retrieve data. Timestamp before and after the query command present the time required to retrieve the data from the table. The following Xquery would read as “from the table named empc select all staff elements that contain designation as ‘System Analyst’”:

```
select current_timestamp
select Empinfo.query(‘/staff/[desig=”System Analyst”]’)
from empc
select current_timestamp
```

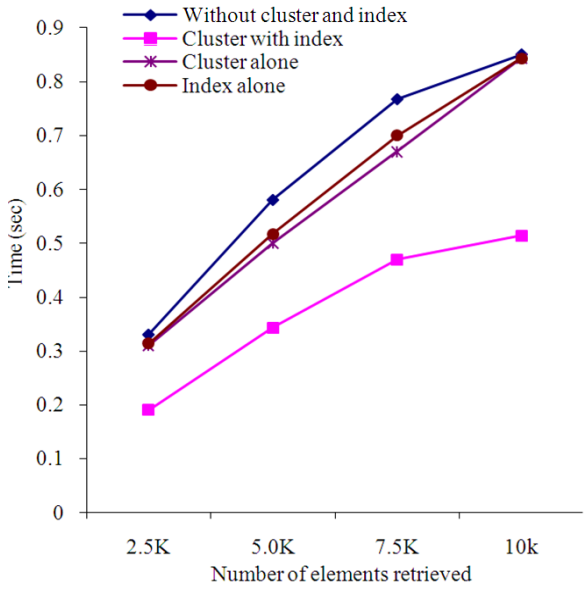

Fig. 4: Performance comparisons SQL server2005

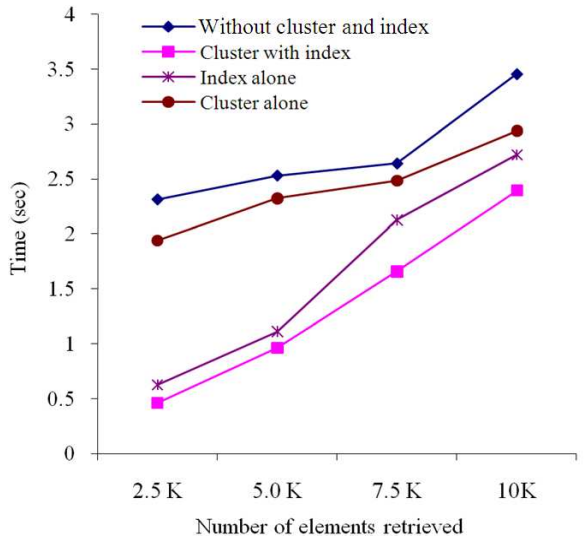

Fig. 5:Performance comparisons Berkely DBXML

The query "from the table named empc select all staff elements that contain designation as 'System Analyst' or 'Project Manager'"

```
select current_timestamp
select empinfo.query('/staff/[desig="System Analyst" or
desig="Project Manager"]' from empc
select current_timestamp
```

The time required for executing the above query without indexing and clustering is 0.58 sec. One of the major advantages of modern native XML databases is their ability to index the XML elements they contain. Proper use of indices can significantly reduce the time required to execute a particular Xquery expression. The column Empinfo elements are retrieved and clustered using the field designation. The elements with designation value "System Analyst" are grouped and placed in the beginning of the document and similarly other groups are placed. After clustering the clustered data are inserted into the table empc and is indexed with column empinfo. Now the queries are executed to retrieve the elements of particular category. Figure 4 shows that time required to retrieve XML data from the table under different strategies. Y-axis represents the time required to retrieve XML data from the table in seconds. X-axis represents the number of elements retrieved. The time required to retrieve 2500 elements of particular type from the table without cluster and indexing is 0.33 sec.

After indexing time required retrieving the data is 0.314 sec. The time required to retrieve the same data with cluster and indexing is 0.19 sec.

Performance analysis BDBXML: In BDBXML the XML data are maintained in the containers. Initially container named staff.dbxml is created. In this container ten thousand elements are added with the following field. Serial number as attribute, name, department, designation, address as elements. The command create container is used for creating the container and elements are inserted to the container by putDocument. From the container staff.dbxml queries can be used to retrieve the data. For example without indexing and clustering the query used to retrieve data. The query could read as "from the container named staff.dbxml select all staff elements that contain designation as "System Analyst".

Time in seconds for command query is 2.313.

The above time is required for retrieving the data from the container without indexing and clustering. With indices Berkely DBXML can find the subset of matching elements with a single or significantly reduced set of lookups. By carefully applying indexing strategies the retrieval performance can be improved considerably. The container elements are first clustered using field designation. After clustering the clustered data are inserted into the container staff.dbxml and is indexed with filed designation. Now the queries executed to retrieve the elements of particular category. Fig. 5 shows that time required to retrieve XML data from the container under different strategies. Y-axis represents the time required to retrieve XML data from the whole document container in seconds.

Table 1: Performance comparison SQL server 2005

	Number of elements retrieved			
Strategy	2.5.k	5.0 k	7.5k	10k
Without cluster and index	0.330	0.58	0.767	0.85
Cluster alone	0.310	0.50	0.670	0.84
Index alone	0.314	0.53	0.700	0.84
Clusterwith index	0.190	0.34	0.470	0.51

Table 2: Performance comparisons Berkeley DBXML

	Number of elements retrieved			
Strategy	2.5.k	5.0 k	7.5k	10k
Without cluster and index	2.310	2.530	2.640	3.450
Cluster alone	1.940	2.325	2.485	2.937
Index alone	0.625	1.110	2.125	2.718
Clusterwithindex	0.462	0.964	1.658	2.390

X-axis represents the number of elements retrieved. The time required to retrieve 2500 elements of particular type from the container without cluster and indexing is 2.313 sec. After indexing time required retrieving the data is 0.625 sec. The time required to retrieve the same data with cluster and indexing is 0.462 sec. This graph proves that, clustering with indexing procedure yields the better performance.

DISCUSSION

Table 1 shows the time required for the data retrieval from the table in SQLserver2005. Table 2 contains the time required to retrieval of data from the container of Berkely database. Clustering increases performance for data retrieval certain extent. Indexing supports fast retrieval of data for manipulation. Figure 4 and 5 show that the proposed similarity based clustering method with indexing provides better performance in data retrieval.

CONCLUSION

Performance of similarity based clustering method with indexing is analyzed for XML document in SQLServer2005 and Berkeley DBXML. From the results it is concluded that the indexing alone does not yield the expected performance improvement. If clustering is combined with indexing it offers better performance than expected level of indexing or clustering alone. In future it is planned to modify the existing indexing techniques of XML to improve the efficiency of data retrieval. Further it is proposed to test clustering method using oracle database management systems.

REFERENCES

Dalamagas, T., T. Cheng, K.J. Winkel and T. Sellis, 2006. A methodology for clustering XML documents by structure. Inform. Syst., 31: 187-228. DOI: 10.1016/j.is.2004.11.009

Francesca, F.D., G. Gordano, R. Ortale and A. Tagarelli, 2003. Distance-based clustering of XML documents. Proceedings of the 1st International Workshop on Mining GraphsTrees and Sequences (MGTS' 03), pp: 75-78.

Hwang, J.H. and K.H. Ryu, 2010. A weighted common structure based clustering technique for XML documents. J. Syst. Software, 83: 1267-1274. DOI: 10.1016/j.jss.2010.02.004

Kim, T.S., J.H. Lee and J.W. Song, 2008. Semantic structural similarity for clustering XML documents. Proceedings of the International Conference on Convergence and Hybrid Information Technology, Aug. 28-30, IEEE Xplore Press, Daejeon, pp: 552-557. DOI: 10.1109/ICHIT.2008.183

Lee, J.W., K. Lee and W. Kim, 2001. Preparations for semantics-based XML mining. Proceedings of the IEEE International Conference on Data Mining, Nov. 29-Dec. 02, IEEE Xplore Press, San Jose, USA, pp: 345-352. DOI: 10.1109/ICDM.2001.989538

Leung, H.P., F.L. Chung and S.C.F. Chan, 2005b. On the use of hierarchical information in sequential mining-based XML document similarity computation. Knowl. Inform. Syst., 7: 476-498 DOI: 10.1007/s10115-004-0156-7

Leung, H.P., F.L. Chung, S.C.F. Chan and R. Luk, 2005a. XML document clustering using common Xpath. Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration, Apr. 08-09, IEEE Xplore Press, pp: 91-96. DOI: 10.1109/WIRI.2005.39

Lian, W., D.W.I. Cheung, N. Mamoulis and S.M. Yiu, 2004. An efficient and scalable algorithm for clustering XML documents by structure. IEEE Trans. Knowl. Data Eng., 16: 82-96. DOI: 10.1109/TKDE.2004.1264824

Nayak, R. and W. Iryadi, 2007. XML Schema clustering with semantic and hierarchical similarity measures. Knowl.-Based Syst., 20: 336-349. DOI: 10.1016/j.knosys.2006.08.006

Nierman, A. and H.V. Jagadish, 2002. Evaluating structural similarity in XML documents. University of Michigan.

Rusu, L.I., W. Rahayu and D. Taniar, 2008. Intelligent dynamic XML documents clustering. Proceedings of the 22nd International Conference on Advanced Information Networking and Applications, Mar. 25-28, IEEE Xplore Press, Okinawa, pp: 449-456. DOI: 10.1109/AINA.2008.122

Tagarelli, A. and S. Greco, 2004. Clustering transactional XML data with semantically-enriched content and structural features. Web Inform. Syst., 3306: 266-278. DOI: 10.1007/978-3-540-30480-7_28

Yang, J., W.K. Chenung and X. Chen, 2005. Integrating element and term semantics for similarity-based XML document clustering. Proceedings of the IEEE/WICA/ACM International Conference on Web Intelligence, Sept. 19-22, IEEE Xplore Press, pp: 222-228. DOI: 10.1109/WI.2005.80

Zhang, K., R. Statman and D. Shasha, 1992. On the editing distance between unordered labeled trees. Inform. Process. Lett., 42: 133-139. DOI: 10.1016/0020-0190(92)90136-J

Zhang, Z., R. Li, S. Cao and Y. Zhu, 2003. Similarity metric for XML documents. The Pennsylvania State University.