*IOSR Journal of Engineering (IOSRJEN)*
*ISSN: 2250-3021 Volume 2, Issue 7(July 2012), PP 78-81*
*www.iosrjen.org*

# OBJECT BASED DATA PARTITION ALGORITHM FOR XML DATABASE

Suja Jayachandran[1], Sachin Bojewar[2]
*Department Of Computer Engineering Vidyalankar Institute Of Technology Wadala(E), Mumbai-400037,India*

***ABSTRACT : -*** *The growing use of XML data format in global information needs an effective XML data management system. With the rapid growth of XML data in internet, we are confronted with big data issues; it's becoming a new research direction for managing massive XML data now. Conventional centralized data management technologies are limited in the aspects of efficiency, throughout and maintenance cost. This ability coupled with the increase use of XML data in different areas have triggered the need for a better method to structure a large data in order to improve query performance. Issues concerning the ways to efficiently partition large XML documents into a more manageable form are yet to be addressed. At the same time, it is essential to ensure that the partitioning method maintains the preservation of XML data hierarchical structure. Effective data management system for storing and querying large document repositories is required. Managing large XML repositories are storing and querying XML data sets within either an Enabled XML database or a Native XML database. This limitation related to xml database is resolved in this project using partition algorithm-Object Based Data Partition Algorithm. It structures large XML data logically by partitioning them into object based XML components.*

***Keywords:*** *XML Database, Object Based Partition, OXDP algorithm, Query performance, Search optimization*

## I. INTRODUCTION

XML is rapidly emerging as t he new standard for data representation and exchange on the Internet. As large corporations and organizations increasingly exploit the Internet as a means of improving business-transaction efficiency and productivity, it is increasingly common to find operational data and other business information in XML format. In light of the sensitive nature of such business information, this also raises the important issue of securing XML content and ensuring the selective exposure of information to different classes of users based on their access privileges. XML processing is gaining momentum as it affects the reliability and performance of many computationally intensive applications, ranging from the recent Web-based and Grid-based infrastructures to the more traditional integrated and cooperative information systems. In all such scenarios, efficiently querying native XML databases is a critical issue, due to the evidence that non-native databases may be too inefficient.

In section 2, we will site some advantage of using xml database, then in section 3 we will see xml limitation then we proceed to section 4 in which we will see the proposed algorithm for data partitioning to improve the query performance. In section 5 we will compare the result obtained from both partitioned and un-partitioned database.

## II. ADVANTAGES OF USING XML DATABASE

XML has wide-ranging features to support global data representation and exchange over the web. Consequently, XML data has grown to expand significantly. It is mainly used for storing and transferring of data. It removes two constraints which were holding back Web developments:

1. dependence on a single, inflexible document type (HTML) which was being much abused for tasks it was never designed for;
2. The complexity of full SGML, whose syntax allows many powerful but hard-to-program options.

XML allows the flexible development of user defined document types. It provides a robust, non-proprietary, persistent, and verifiable file format for the storage and transmission of text and data both on and off the Web; and it removes the more complex options of SGML, making it easier to program for. Here data can be reused and we can easily import data from anywhere since it is platform independent.HTML tells how data looks where as XML tells what data means. XML applies context to the data and separates content from the data presentation. XML provides a hierarchical structure

## III. XML DATABASE LIMITATION IN TERMS OF QUERY PERFORMANCE

Along with query performance other limitations are issues of maintaining security in a dynamic XML environment have attracted interest from researchers in the field. [3]The current security mechanism preferred is the use of an encryption method for client-side access control. However, this method has setbacks, including the problem of encryption key management or limitations of fine-grained access control. In XML, the issue of quality data for mining purposes and also using data mining techniques for quality measures is becoming more necessary as a massive amount of data is being stored and represented over the web. [4]Two important interrelated issues: how quality XML data is useful for data mining in XML and how data mining in XML is used to measure the quality data for XML. Most existing research efforts focus on efficient filtering algorithms for achieving a high system performance or supporting more complex XPath syntax. Each proposed scheme has its advantages and limitations. Not much research, however, has considered using caching in the context of XML filtering. For this we propose [2] two caching schemes to be used in conjunction with an XML filtering engine. A complete message caching algorithm that is a strict caching policy to reduce the computation cost that accrues from multiple filtering of the same messages, by reusing results of previously processed messages. Second, we investigate a structure-based caching method that is an approximate caching policy for messages sharing the same structure.

Effective data management system for storing and querying large document repositories is required. Managing large XML repositories are storing and querying XML data sets within either an Enabled XML database or a Native XML database. Another previous approach is mapping or shredding XML data into Object-Relational Database Systems (ORDB) or Relational Database Systems (RDB).
This limitation related to xml database is resolved in this paper using improved partition algorithm-OXDP [1] i.e. Object Based Data Partition Algorithm. It structures large XML data logically by partitioning them into object based XML components. An evaluation is shown to demonstrate the effectiveness of ODP in XML partitioning which subsequently has the potential of improving query performance in Enabled XML DB environments.

In the area of data storage, data in various applications are built in RDB and many users want to take advantages of XML technology beside RDB features. Shredding XML data affects the hierarchical structure of XML documents. Storing entire XML document into column or tables can be very costly especially during manipulation of large data. As a result, there is a need to find a method to partition XML data to improve the query performance. It is necessary to be aware of the number of obtained partitions and the balance of their size since XML data would not gain any advantage from the partitioning process [5] if the number of partitions is very large or the partitions have large dissimilarity between them. This issue needs to be put into consideration to prevent the degradation of query performance. XML documents are partitioned into partitions with correct data structure that correspond to its schema.

It structures XML components logically by partitioning XML data into objects. Algorithm takes XML Entry that is an XML document and its corresponding XML schema (XSD) as input. XSD is used since it represents XML objects and their semantic structure. XML Entry will run through two layers:

i. The semantics layer that is concerned with the semantics and structure of XML data and consists of XTree and Object Based Partitioning algorithms (OBP) and
ii. The optimization layer that will be used to optimize the generated partitions

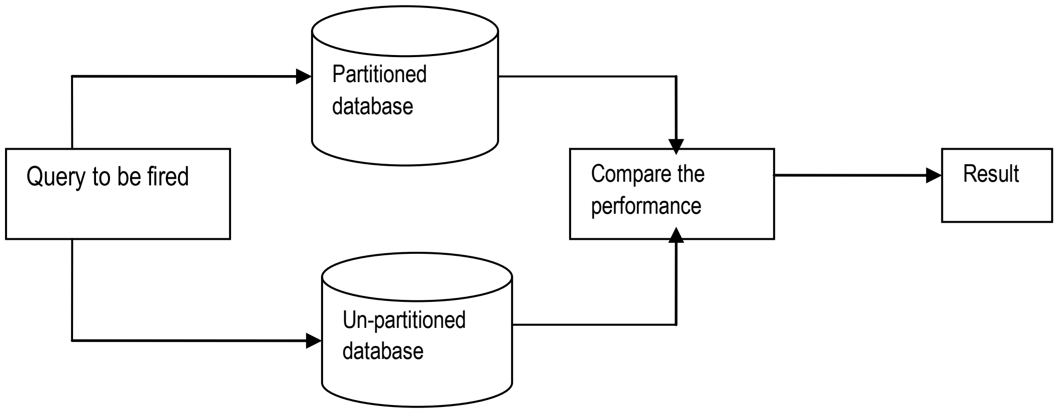


Fig1: Basic flow diagram of the proposed algorithm

**Module 1:** XML Conversion: Data entered by the user from the front end user will be converted into xml format.

**Module 2:** Database partitioning: Algorithm will be used to partition the database to improve the query performance. The xml format will give xtree, by parsing xml data, extracting object oriented features for xtree. Parent root node, basic node (leaf node of the tree) and complex node (non leaf node) are identified. If basic node is absent, then one of the complex nodes will be the parent node of the new partitioned tree. Relationships like generalization, association, aggregation are identified.

**Optimization Layer, in this:**

1. When object (non leaf node with many siblings) size is optimal call partition object (): Process xml data and write all data contained in target object into a new xml file. Then we partition the XSD file by extracting all the elements and attributes of complex node into a new xml schema file.
2. When object size exceeds the maximum limit then call partition_object _horizontally (): Xml data is partitioned into disjointed but similar structured documents that can be stored and accessed separately into its physical storage.
3. When object size is small call merge object (): Take the object name and its parent node and merge them in one partition.

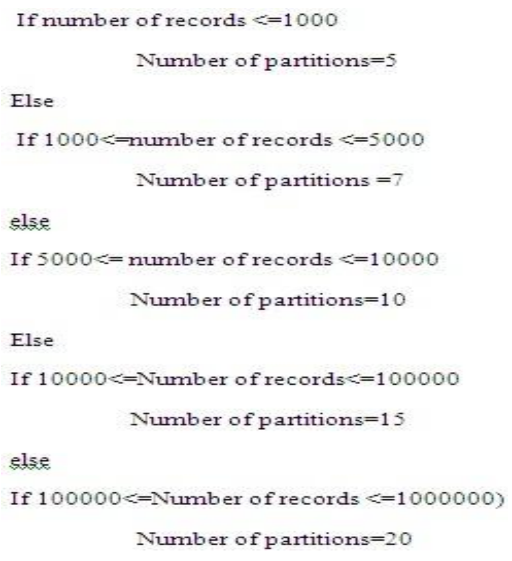
Module 3: Validation: Query performance is checked for partitioned database by comparing it with un-partitioned database for same query on the basis of data retrieval. And we conclude that after partitioning the xml database, its limitation of query performance is improved.

## IV. PROPOSED ALGORITHM FOR DATA PARTITIONING - OBJECT BASED DATA PARTITION ALGORITHM

Object Based XML Data Partition (OXDP) algorithm [1] is modified in this proposed algorithm. The main focus of this paper is to optimize the query performance using the partition algorithm. In this we will run the same query on both partitioned and un partitioned database and check the time of execution and we must get the same query executed in lesser time. But for getting the difference in time the database should be large in size. And here in the proposed partition algorithm we are partitioning the database on the basis of its size. The proposed system can be divided into three modules:

In the module 1 we will accept the data from the user and this data is then converted into xml file after parsing the data entered. In this module we are converting the data which was entered from the User Interface into XML with user-defined tags.As in this project the main focus is on execution time of the same query being fired to partitioned and un-partitioned database, the database must be large enough to give a considerable time difference. So here in this project we will take employee database where each employee will be given unique employee_code which will be auto incremented and then this employee details will be converted into XML format for further processing.

In module 2 based on the amount of data, partitions will be made. According to our algorithm, the data can be randomly entered i.e there is no necessity that all entered employee record (for e.g) are in series, so we have to partition them such that the order of records doesn't matters. Now we will partition the data such that:



```
If number of records <=1000
        Number of partitions=5
Else
If 1000<=number of records <=5000
        Number of partitions =7
else
If 5000<= number of records <=10000
        Number of partitions=10
Else
If 10000<=Number of records<=100000
        Number of partitions=15
else
If 100000<=Number of records <=1000000)
        Number of partitions=20
```

Using this there will be a format in forming the partition. By this method the un-partitioned database will have all the five thousand rows (say for an example) one after other and hence if we search five hundred row's record we have to travel all four hundred and ninety nine rows where as partitioned database will have seven hundred fourteen rows with seven records in one row as we have already declared the total partition to be made is seven for five thousand data records. In this way if the user wants to search five hundredth records then it will go directly to the seventy-two row, so automatically the data fetching time will be less as compared to un-partitioned database.

In module 3 we will run same query in both the database and see that time of data retrieval is less in partitioned database in compare to un partitioned database. Thus query optimization has taken place by three times.

## V. TESTING AND RESULT

Test Plan Objectives will be to verify the result obtained for query performance from the partitioned and un-partitioned database. Scope of testing will be to compare the performance of partitioned and un-partitioned database on the basis of same query entered by the user, expected numbers of record to be tested is more than one lakh records. The test strategy consists of a Stress Testing and Performance Testing which will fully exercise the system. The primary purpose of these tests is to uncover the systems limitations and measure its full capabilities. Result can be compared graphically i.e. if we consider x-axis as number of records in the respective database and y-axis as execution time of the query in milli second, then graph shows that after using the Object Based Data Partition Algorithm the query execution time has been improved.

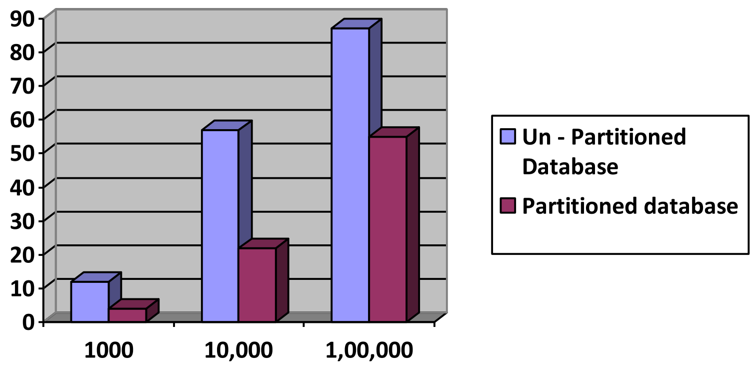


Fig 2: Result Comparison

## VI. CONCLUSION

XML has wide-ranging features to support global data representation and exchange over the web. Consequently, XML data has grown to expand significantly. Hence an effective data management system for storing and querying large document repositories is required. In this paper we have studied various limitations of xml database and out of this for improving query performance we have used enhanced Object Based Data Partition algorithm. Storing entire XML document into column or tables can be very costly especially during manipulation of large data. As a result, there is a method to partition XML data to improve the query performance. It is necessary to be aware of the number of obtained partitions and the balance of their size since XML data would not gain any advantage from the partitioning process if the number of partitions is very large or the partitions have large dissimilarity between them. This issue is put into consideration to prevent the degradation of query performance by using partition algorithm. We can implement this Object Data Partition algorithm with many more features like for solving other limitation in terms of security as future work.

## REFERENCES

[1] Norah AlGhamdi, Wenny Rahayu and Eric Pardede "Object-Based Methodology for XML Data Partitioning (OXDP)" *2011 International Conference on Advanced Information Networking and Applications*

[2] Yang Cao, Shikharesh Majumdar Chung-Horng Lung "Caching Techniques for XML Message Filtering" *2009*

[3] Tomáš Knap, Irena Mlýnková. "Towards More Secure Web Services Exploiting and Analysing XML Signature Security Issues"*2009*

[4] Md. Sumon Shahriar "Quality Data for Data Mining and Data Mining for Quality Data: A Constraint Based Approach in XML" *2008 Second International Conference on Future Generation Communication and Networking Symposia*

[5] Zografoula Vagena,Mirella M. Moro and Vassilis J. Esotras "Efficient Processing of XML Containment Queries using Partition-Based Schemes" *2004*