Query Processing of XML Data Warehouse Using XML Pattern Matching Techniques

[1.]Ms. P. Kavitha, [2.]Mrs. S. Vydehi
[1.] M.Phil Scholar, Department of Computer Science,Dr.SNS Rajalakshmi College of Arts and Science, Coimbatore-49.
[2.]Professor & Head of the Department,Department of Computer Science,Dr.SNS Rajalakshmi College of Arts and Science, Coimbatore-49.

***Abstract*— Data warehousing systems are used by managers and analysts to acquire, integrate and analyze information flexibly from different sources. XML has become a standard for data exchange over the Internet. Mostly XML data's are used in B2B and B2C communication. So there is a need of integrating XML data into warehousing systems. An effective well-formed XML document structure helps convert data into useful information that can be processed quickly and efficiently. The XML query languages like XQL (XML Query Language), XML-QL(a query language for XML), XPath (Extensible path language), XQuery (Extensible Query language) represent queries on XML data as twigs(small tree patterns). This paper describe XML query processing which is the procedure to find all the occurrences of twig patterns efficiently on XML data warehouse. This paper also presents the review of different XML pattern matching techniques in data warehousing.**

Keywords*— XML, XML data warehouse, Pattern Matching Algorithms.*

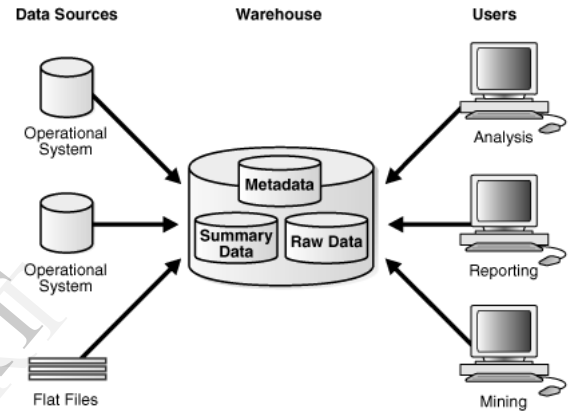
I. INTRODUCTION

A. XML

XML is the Extensible Markup Language.XML improves the functionality of the Web by identifying information in a more accurate, flexible, and adaptable way. It is not a fixed format like HTML, where as HTML is a single, predefined Markup Language. XML is used for describing other languages which lets user designs their own markup languages for limitless different types of documents. XML can do this because it's written in SGML, the international standard meta language for text document markupXML structures can nest, so they can be used to store and identify any kind of hierarchical information especially long, deep, or complex document sets or data sources, which makes it ideal for an information-management back-end to serving the Web. XML is also used for enclosing or encapsulating information in order to pass it between different computing systems.

B. Data Warehouse

A data warehouse is a relational database designed for analysis and query rather than for transaction processing. A data warehouse is specifically structured for querying and reporting. Data warehouse contains historical data which is derived from transaction data. It can also include data from different sources. It enables an organization to consolidate the data from many sources.



A data warehouse environment which includes the applications that manage the process of gathering data and delivering it to business users. The applications like extraction, transportation, transformation, and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools. The data warehouse focuses on data storage but managing the data dictionary is also an essential component of a data warehousing system.

- Types of Data Stored in a Data Warehouse

Historical Data

A data warehouse contains many years of historical data. The amount of data which is decided to make available is based on the available disk space and the analysis type supported by the user. These data's can come from transactional database or other sources.

Derived Data

Derived data is generated from existing data or a data transformation. The existing data uses mathematical operation. The derived data is created when database maintenance operation is performed or generated at run-time in response to a query.

Metadata

Metadata is defined as data about data. The data which is used to represent other data is called metadata. Metadata is the summarized data which leads us to the detailed data. In terms of data warehouse the metadata is defined as follows.

- Metadata is a road map to data warehouse.
- Metadata in data warehouse defines the warehouse objects.
- The metadata act as a directory. This directory helps the decision support system to locate the contents of data warehouse.

C. XML Data Warehouse

Data warehousing system is a set of technologies and tools that enable users and analysts to integrate and analyze information flexibly coming from different data sources. The database which is specialized for complex analysis of historical data, called a data warehouse. In Recent days many industries prefer XML data exchange format. These results the organisations want to find ways to manipulate and manage XML efficiently within their data warehouses. Many leading industries like finance, healthcare and other already having the production environments that leverage both relational and XML database technologies. Increasing use of XML, lot of valuable external data sources will be available in XML format on the Internet. The possibility of integrating available XML data into data warehouses will play an important role in providing enterprise managers with up-to-date and comprehensive information about their business domain.

D. XML Pattern Matching

The extensible markup language XML has a new standard for information exchange and representation on the internet. With the increasing use of XML for data representation, there is a lot of interest in query processing over data. The data objects are written in a variety of languages (XPath, XQuery) are typically trees [1]. For querying the data from xml data sources the pattern matching is used. Twig pattern matching is referred as finding all the instances of the query tree embedded in the XML data tree. So there is need for efficient pattern matching algorithms on large volume of XML data for evaluating tree patterns (twigs).

```
1.<?xml version="1.0" encoding="UTF−8"?>
2.<Book>
3.<author>jane</author>
4.<title> XML </title>
5.</Book>
6.</xml>
```

Figure 1.4.1: Example XML document

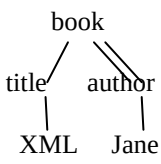


Figure 1.4.2 Example XML tree model

An XML query pattern can be represented as a rooted, labelled tree (Twig), for example Fig 1.4.2 shows an example XPath query:

Book [title = ‟XML‟] // author [. =’’Jane‟]

A query tree pattern can be decomposed into a set of basic P-C and A-D relationship between pairs and nodes [4]. The above example query are the ancestor-descendent relationship (book, author) and the parent-child (book, title) and (title, XML) and (author, Jane).The xml query languages such as XPath and XQuery which is used to represent the queries as ordered labelled small trees(twigs). Finding all distinct matching’s of a twig pattern is a core operation in an XML query evaluation.

E. Existing Techniques of XML Pattern Matching

In the early days, many research efforts made on storing and querying XML data using RDBMS. In relational approaches, they shred XML data into relational tables, and XML queries are converted into SQL statements to query the database. The node-based approach [5], the edge-based approach and the path-based approach shred XML documents based on tree components. But, they all suffer from efficiency problems when dealing with structural search. The schema-aware de-composition methods are efficient than schema less method, but they are still not efficient for structural search. Later, many native approaches are proposed to process twig pattern queries like the navigational approach, subsequence matching approach, structural join based approach.

Later an improved stack-based structural join algorithm is proposed by Al-Khalifa et al.. This algorithm decomposes a twig pattern into a set of binary relationships, i.e., parent-child and ancestor-descendant relationships. Twig pattern matching is done by matching every binary relationship and combining these basic binary matches. The main problem of those approaches is that the intermediate result size may be very large, even when the input and final result sizes are more manageable. To overcome this limitation problem, Bruno et al. [5] proposed a holistic twig join algorithm called TwigStack, which avoids an unnecessarily large intermediate result produced. However, TwigStack algorithm is only optimal for twig pattern queries with ancestor descendent relationships.

II. VARIOUS XML PATTERN MATCHING TECHNIQUES

A. TwigStack

Bruno et al. proposed a “holistic” XML twig pattern matching method called TwigStack. If all edges in query pattern are ancestor – descendant (A-D) relationships, Twigstack algorithm ensures that each root – to – leaf intermediate solution is merge – joinable. The TwigStack algorithm is used to evaluate twig patterns that operate in two phases. In first phase, some solutions to individual query root-to-leaf paths are computed. In second phase, the computed solutions are merge-joined to compute the answers to the twig query.In TwigStack each root-to-leaf solution is merge-joinable with at least one solution to each of the other root-to-leaf query paths. The algorithm might produce some extraneous intermediate solutions where the twig pattern contains a parent-child edge between two nodes.

Drawbacks

- TwigStack is optimal, in terms of the size of the intermediate solutions.
- TwigStack has been proved to be I/O optimal in terms of output sizes for queries with only A-D edges.
- The algorithm cannot control the size of intermediate results for queries with parent-child (P-C) edges.

- TwigStack outputs many intermediate paths that are not merge-joinable .
- The main reason for bad performance is that in the TwigStack, it assumes that all edges in queries are A-D relationships and therefore output many useless intermediate results when queries contain P-C relationships.

B. OrderedTJ

In OrderedTJ, if the order of the children accords with the order of corresponding query nodes an element contributes to final results. OrderedTJ is I/O optimal among all sequential algorithms that read the whole input. The optimality of OrderedTJ [3] allows the existence of parent-child edges in first branching edge and the non-branching edges.

Consider an ordered twig pattern Y and an XML database D, a match of Y in D is identified by a mapping from the nodes in Y to the elements in D, such that:

(i)The corresponding database elements must satisfy the query node name predicates; and, (ii) the corresponding database elements must satisfy the parent-child and ancestor-descendant relationships between query nodes; and (iii) the corresponding database elements must satisfy the order of query sibling nodes.

Based on the containment labelling scheme, given any query node p and its right-sibling r, their corresponding elements, say ep and er, must satisfy that ep.end<er.start. Here, we do not allow ep to be an ancestor of er.

Drawbacks

- OrderedTJ output much less intermediate results.
- OrderedTJ scales linearly with the size of the database.
- OrderedTJ is not optimal and outputting less useless intermediate results.

C. TJFast

I have presented two holistic algorithms for answering XML twig queries in the above sections**.** The same containment labelling scheme is used by those two algorithms [4]. While the containment scheme preserves the positional information within the hierarchy of an XML document, we observe that this is not the only labelling scheme that can be used for XML twig query processing. Indeed, there are at least two limitations in the containment scheme.

1. The information contained by a single containment label is too limited. For example, from any single containment label we cannot get the path information.
2. When element names are unknown or do not matter wildcard steps in XPath are commonly used.

If the branching nodes consist of wildcards then the containment labelling scheme is difficult to answer queries. For example, consider an XPath: "//p/*/[q]/r". where "*" denotes a wildcard symbol which can match any single element. The containment labels of p, q and r do not provide enough information to determine whether they match the query or not. This is because even if q and r are descendants of p and their level difference with p is 2, q and r may not be query answers, as they do not have the common parent.

TJFast operates in two phases. In first phase, solutions to individual root-leaf path patterns are computed. In second phase, the computed solutions are merge-joined to compute the answers to the query twig pattern.

Drawbacks

- TJFast outputs one useless intermediate path and it is outputs the path solution for all nodes in query.
- It does not produce the individual solution for each node when there are multiple return nodes in a query.
- TJFast cannot work with ordered restriction and negation function.

D. TreeMatch

The above algorithms TwigStack, OrderedTJ, and TJFast requires bounded main memory for small class of queries with Parent-Child, Ancestor-Descendant relationships.The functions such as negation, wildcard, order-based functions and relationships are defined by the XML query languages like XPath , XQuery. The TreeMatch algorithm defines an extended XML tree pattern [1] means P-C, A-D, negation, wildcard and/or order restriction.

The TreeMatch algorithm is proposed to for larger optimal query classes. It uses encoding technique to match the results and reduces the useless intermediate results. The TreeMatch algorithm does not need to de-compose the tree pattern into linear patterns and do not produce any intermediate results that are not part of the final results. The TreeMatch algorithm is applicable when the non-leaf pattern nodes do not have occurrences with self-containment. The self-containment is seldom found in real XML documents and such a property can easily be identified. Therefore, the TreeMatch algorithm is more efficient than the existing methods under most cases.

Advantages

- The TreeMatch algorithm does not need to de-compose the tree pattern into linear patterns.
- The TreeMatch algorithm does not produce any intermediate results that are not part of the final results,
- The TreeMatch algorithm is applicable when the non-leaf pattern nodes do not have occurrences with self-containment.
- The TreeMatch algorithm is more efficient.

III. RESULT

From the above study, when comparing the algorithms TwigStack, OrederedTJ, TJFast and TreeMatch the first three techniques of XML pattern matching has some drawbacks on query processing when comparing to TreeMatch .The TreeMatch algorithm has more advantages than existing techniques so here i conclude that when we use TreeMatch algorithm for query processing of XML data's it gives good result and performance even for complex queries.

IV. CONCLUSION

This paper deals with study of different kind of XML pattern matching algorithms. It first defines the XML, it was designed to transport and store data, with focus on what data is. Then it defines the Data warehousing process and the Types of Data Stored in a Data Warehouse. After that a detailed study of XML pattern matching and its algorithms and their

comparison in different perceptions are examined. The existing methods for tree pattern matching in XML are typically a decomposition-matching-merging process.

The drawback of the decomposition-matching-merging method is that the size of intermediate results may be much larger than the final answers. The decomposition-matching-merging problem will be overcome by using TreeMatch algorithm. The previous twigpattern matching algorithms (TwigStack, OrderedTJ and TJFast) requires more features than TreeMatch algorithm. From these points we can say that TreeMatch twig pattern matching algorithm can answer complicated queries and has good performance. This paper highlights the concerned issues and challenges which may be helpful for the upcoming researchers to carry on their work.

REFERENCES

[1] D.Bujji Babu, Dr. R.Siva Rama Prasad, M.Santhosh, Twig Pattern Matching Algorithms for XML, International Journal of Advanced Research in Computer Science Engineering , Volume 2, Issue 5, May 2012

[2] J. Lu, T. W. Ling, Z. Bao, and C. Wang. Extended xml tree pattern matching: theories and algorithms. IEEE transactions onknowledge and data engineering, vol.23, no. 3, march 2011

[3] Iyad Batal, Alexandros Labrinidis QuickStack: A Fast Algorithm for XML Query Matching, Department of Computer Science University of Pittsburgh, June 6, 2008

[4] J. Lu, T. Chen, and T. W. Ling, TJFast: Efficient processing of XML twig pattern matching. Technical report, National university of Singapore, 2010.

[5] Huayu Wu, Tok Wang Ling, Bo Chen, and Liang Xu, TwigTable: using semantics in XML twig pattern query processing, School of Computing, National University of Singapore, 2009.

[6] M.Muthukumaran, R.Sudha. Efficiency of TreeMatch Algorithm in XML Tree Pattern Matching, IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661 Volume 4, Issue 5 (Sep-Oct. 2012), PP 19-26

[7] Boris Vrdoljak,Marko Banek,Zoran Skočir , A Methodology for Integrating XML Data into Data Warehouses, University of Zagreb Faculty of Electrical Engineering and Computing .

[8] Marko Banek, Zoran Skočir and Boris VrdoljakFER, Logical Design of Data Warehouses from XML, University of Zagreb, Zagreb, Croatia