Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

Selective Sparse Mapping of XML to a Widely Populated Table

Supritha D Shettigar[1], Mrs. Sadhana[2]

[1]*IV Semester, M.Tech, Computer Science and Engineering, Sahyadri College of Engineering and Management*

[2] *Asst. Prof., Department of Computer Science and Engineering, Sahyadri College of Engineering and Management*

Abstract- XML is usually supported by SQL database System. Existing mapping cannot support the use cases that provide normalization. Here, the novel mapping of XML data into sparse column of populated table is proposed. This helps the enterprise application by providing the better performance for document types and for the queries that are not yet supported by existing system. The purpose of the project is to reduce the document size and also to avoid the storage overhead. Based on new mapping, document size will be reduced and improves the execution time.

Keywords- XML:Extensible Mark-up Language, DG:Data Guide, SQL: Structured Query Language, indexes, mapping, XML documents.

I. INTRODUCTION

XML is an Extensible Markup language, which defines a set of rules for encoding documents in a format that is either in human readable or machine readable. The simplest method to store XML data in a relational database System is by using a long-character-string data type, such as CLOB in SQL, to store XML documents or fragments as text in columns of tables [11].This approach provides textual fidelity so that it can preserve the originality of the XML in the form of character-string level. The disadvantage of the character-string is to take the advantage of the structural information so that it can be available to the XML markup. A generic string data type does not provide any specialized support for searching based on semantic content, or for retrieving XML data at a fine level of granularity. Another method in the relational database for XML documents to be stored is nothing but by the shredding process, which allows the distribution of XML information around the columns of one or more tables, which protect the originality of both the data values and structural relationships. This technique is most frequently used for the XML documents so that its structure can be described by an XML schema.

A mapping which is nothing but sparse mapping, maps the entire document collection from the database, but the applications are interested only in the small portion of the document. This method cannot solve the use cases which are found in the XML spectrum such as many small documents and many optional elements. This incurs in storage overhead and also increases the execution time. So the new technique is introduced to overcome the problem of the sparse mapping. The mapping from XML to wide sparse table is done by introducing DataGuides and a node encoding scheme. Data Guide summarizes all paths in XML documents into a labeled tree. All nodes having same root to node path will be mapped to single node in the DG so that it enables distinctly labeled path appears exactly once [1].

II. RELATED WORK

A. Barta, M. P. Consens, and A. O. Mendelzon proposed two-level optimization strategy for XML query optimization. In this the higher level consists of join order selection along with the cost-based selection of access method. The lower level consists of XPath expression in the form of cost based selection for evaluation plans.

In holistic path pruning, any path summary performing certain path pruning will also be the part of the path expression evaluation. The difference between holistic pruning and traditional pruning is that the traditional path pruning will be intrinsic to particular path summary evaluation method, while holistic path pruning will be generic, so that any path evaluation method can be used afterwards. Since it is generic, it is possible to reduce the search space. The access order selection works in conjunction with the holistic path summary pruning and also computed using either bottom-up or top-down evaluation for the path in the XML documents [2].

Thus the holistic path pruning reduces query plan search space by identifying the query answer that is present in the document. Whereas Access-order selection works in conjunction with holistic path pruning and also reduces the query search space based on cost-based heuristics.

A. Barta, M. P. Consens, A. O. Mendelzon proposed a two level-based cost optimization for XML path indexes. A path index called as structural index or as a structural summary, that represents the summarization of path actually occur in an XML document. The high level cost-based selection is based on two access methods specialized for query processing. The first method supports single pass pattern matching in XML documents whereas second method takes path index structure for match the preprocessed pattern of XML documents. The second access method determines the path in which it has to be traversed and also encapsulate the pruning of the patterns against path index [3].

S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava mainly focued on the constraint dependent minimization (CDM) and Algorithm constraint independent minimization (ACIM) in the presence or absence of Integrity Constraints (ICs). ACIM also a polynomial take the worst case of time $O(n^6)$ in the query size, where as CDM produces a locally minimal equivalent query for a given query in the time $O(n^2)$ in the query size. ACIM time depends on number of redundant nodes in a query pattern, the degree of redundancy, the query size and finally, the constraints that generate redundancy in the query pattern. CDM time varies with varying query sizes, shapes. Thus polynomial time can be reduced in the absence of ICs for tree pattern queries [4].

A. Arion, A. Bonifati, G. Costa, S. D'Aguanno, I. Manolescu, and A. Pugliese mainly focused on the compression principles known as Order-agnostic compression and Order-preserving compression. In Order-agnostic algorithm, classical Huffman also known as simple algorithm is chosen so that the best possible redundancy among the resulting code is achieved and the process of encoding and decoding is also faster than universal compression technique. Finally, the strings compressed with Huffman along with the set of fixed codeword's can be compared within the equality predicates in the compressed domain and the inequality predicates need to be decompressed.

In Order preserving algorithm is not immediate. ALM uses relational databases for blank-padding and for indexes compression, because of its dictionary based nature decompression is faster than Huffman, also outputs bigger portion of a string at a time when decompressing. The ALM algorithm allows inequality and equality predicates in the compressed domain but not wildcards, whereas Huffman supports prefix-wildcards and equality but not inequality. Thus, the choice of the algorithm can be aided by a proper query workload [5].

A. Arion, A. Bonifati, G. Costa, S. D'Aguanno, I. Manolescu, Murray mainly focused on the ALM method, where the encoding and decoding of the same parser will be done. Since the parser have the prefix property, the dictionary entries for that parser is not required. Flexible compression of the long strings will be permitted by ALM. Ordering of the encoded form will be preserved by keeping it in the same order as that of the dictionary entries. Encodings are reduced so that preserved order will be compared with the real competitor that is ZIL method [5].

Antoshenkov, G., Lomet, D., Murray proposed compression technique which considers both sorting and searching. In tag sorting method, sort key for each record will be extracted and it will be stored in the record pointer so that the shifting the entire records will be prohibited. Order preserving key compression technique will reduce the tag size, comparison and moves will speed up. In Index tree compressed key will be stored instead of uncompressed variants. Both the compression technique need to be static as well as order preserving, since static technique will preserve the order all time [6].

Jayavei Shanmugasundaram, Rajasekar Krishnamurthy, Igor Tatarinov XML documents are created over a relational database system so that schema generation technique is automatically creating relational tables for storing XML documents. These documents are stored as rows in these tables. Virtual reconstruction of the stored XML document will be done from the shredded rows. XML documents will be treated as XML view of relational data [7].

K. S. Beyer proposed the representative object concept which provides a clear representation of the schema of the hierarchy of the data source. Here the usefulness of the objects is highlighted along with their primary uses. The implementation of the FRO's in OEM gave the advantage of storing and querying of the data part as an object in OEM. A construction method minimal FRO's was presented which benefited in allowing efficient querying of schemas of the represented data. Many alternative approaches for constructing k-RO were described in order to overcome the very high complexity of minimal FROs construction [8].

R. Goldman and J. Widom mainly focused on Data Guide, which provides a concise, accurate, and convenient summary of the structure of a database and it, is referred to as source database. Source database is identified by its root object. Conciseness will be achieved by describing the DataGuides unique label path of a source exactly once, regardless of the number of times appeared in that source. Accuracy will be ensured by specifying the DataGuides with no label path that does not appear in the source. Finally to obtain convenience, DataGuides need to represent itself as OEM object [9].

A. Marian and J. Siméon mainly focused on projecting the XML document by having the set of paths in the document tree, by specifying the nodes to keep in a projected document. The technical challenge is to identify the compile time for the path which need to evaluate a given query. For this static analysis of the paths is used within a given XQuery expression and is difficult because XQuery is the aspects of both syntactic and semantics. Path analysis technique are used to evaluate queries up to two GB on files even the machines have a limited memory [10].

III. PROPOSED SYSTEM

In proposed approach the technique called Selective Sparse Mapping is used. Selective sparse mapping is an index mechanism that maps only the records that are of the user interest to the sparse column. It's a pure path expression so that it uses only / and // axis and the symbol '*'. The elements of XML which satisfies are stored in the sparse column. Elements of separate columns are gathered in one row as specified by mapping rules [1]. The selected paths are combined into tree called as super DG, which is represented as logical subtables, called as Meta data.

Mapping is done based on the search field where it may be any of the attributes from the current database. Records related to that search field will be displayed and record of the user interest is mapped. ID's of mapped records will be stored in the XML document named by that search field whose size will be of Kb's. This XML documents are represented as logical sub-tables. If the end user needs to view the record he can view it from the XML document if that record is created or directly from the database.

A. System Architecture

A System Architecture is a composition of the components that define the structure, behavior of the system. User will login and inserts the record to the database. Since it is large in number it will be difficult for the end user to search a record from the database. User will perform mapping operation on the database to retrieve records. For mapping operation indexing mechanism is used so that the records only of the user interest will be retrieved. The Id's of retrieved record will be getting saved in the XML documents.

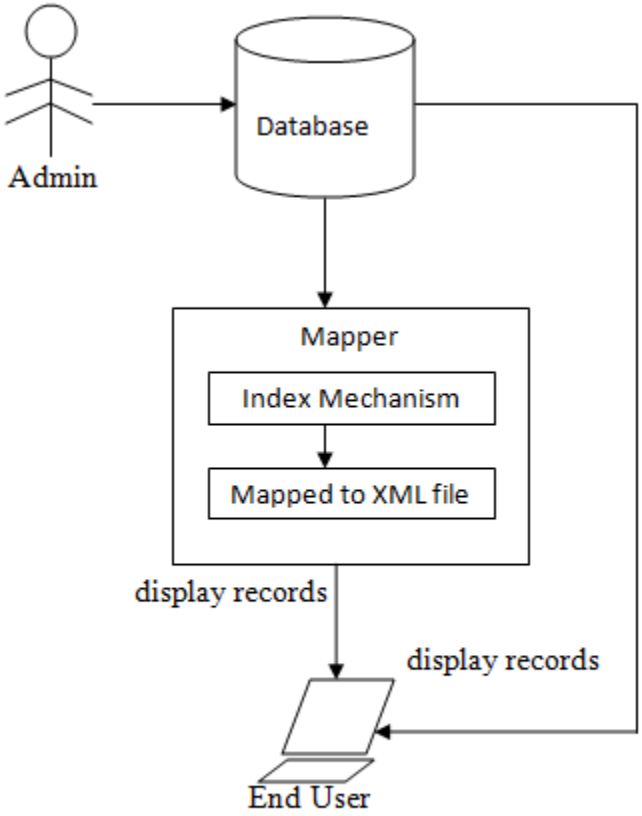

Figure 1. System Architecture

The role of the end user is to just view the records from the database if its XML document is not created or it can directly view it from the XML file. To make comparisons first the end user will view the record from the database and will note down the time which will be larger. To get the better result mapping operations will be performed by the admin for the records and that mapped records will get stored in the XML document. For the next time after performing the mapping operation on the same record end user will view the record from the XML document rather than from the database and once again will note the time. It is found that the records viewed from the XML document take less time than that from the database. This time difference will be shown in the form of graph for the better appearance of the result.

IV. RESULT

The performance of the proposed system is good in terms of the execution time. Its execution time is less when compared to that of the existing system i.e., sparse mapping. Rather than mapping the entire document, only the small part of the document is mapped which result in the creation of small document which was not achieved by the existing system.

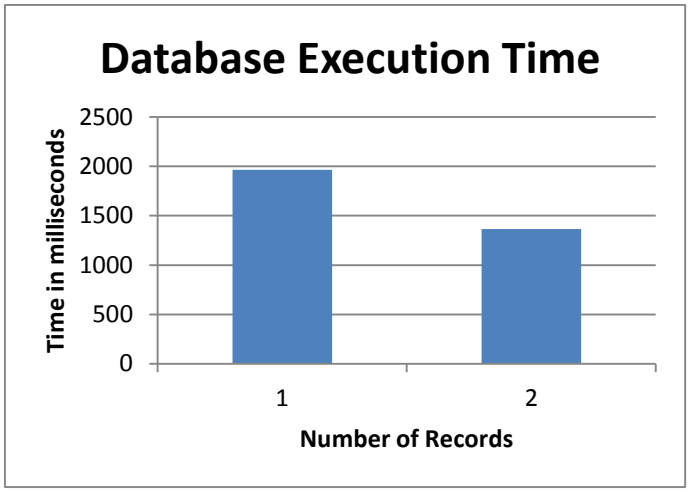

Figure 2. Graph representation for Database Execution Time

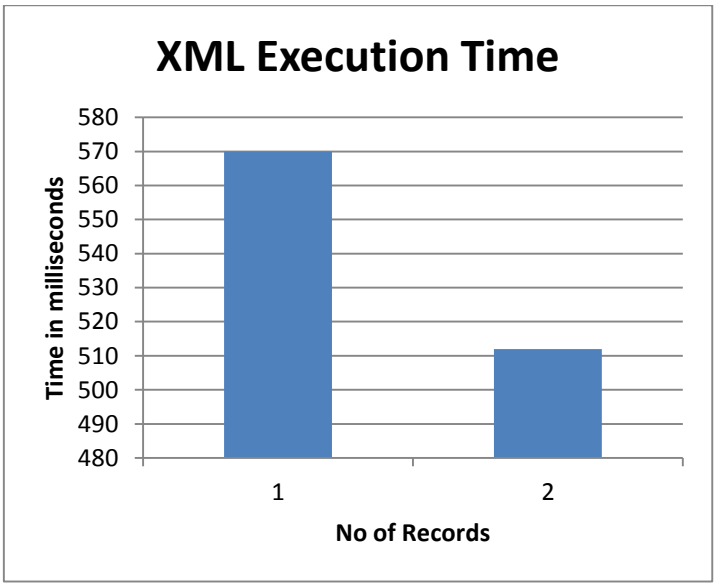

Figure 3. Graph representation for XML Execution Time

Graph representation in the figure 2 gives the execution time for database before mapping which is in thousands of milliseconds whereas the figure 3 gives the execution time for XML after performing the mapping operations which is in hundreds of milliseconds. From the figure it is clear that the execution time for the XML is much less than that of the Database execution time.

V. CONCLUSION

Selective Sparse Mapping reduces disk I/O, since it knows the query workload in advance and only the elements asked by the queries are indexed. Hence the size of XML documents is reduced and also

leads to the creation many XML documents which is the use case of XML spectrum achieved. This technique improved the execution time and also avoided the storage overhead.

References

[1] L. J. Chen, P. A Bernstein, P. Carlin, M. Rys,Mapping XML to Wide Sparse Table ",in Proc. IEEE28thICDE,vol.26,NO.6,June2014.
[2] A. Barta, M. P. Consens, and A. O. Mendelzon, "Benefits of path summaries in an XML query Optimizer supporting multiple access methods," in *Proc. 31st Int. Conf. VLDB*, 2005.
[3] A. Barta, M. P. Consens, A. O. Mendelzon, "XML Query Optimization Using Path Indexes", Proc.XIME–P2004.
[4] S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava, "Minimization of tree pattern queries," in *Proc. SIGMOD*, Santa Barbara, CA, USA, 2001.
[5] A. Arion, A. Bonifati, G. Costa, S. D'Aguanno, I. Manolescu, and A. Pugliese, "Efficient query evaluation over compressed XML data," in *Proc. EDBT*, Heraklion, Greece, 2004.
[6] Antoshenkov, G., Lomet, D., Murray, Order preserving string compression. In: Proc. Of the ICDEConf.(1996)655–663.
[7] Jayavei Shanmugasundaram, Rajasekar Krishnamurthy, Igor Tatarinov, "A General Technique for Querying XML Documents using a Relational Database System " , SIGMOD Record, Vol. 30,No.3,September2001.
[8] K. S. Beyer *et al*. "System RX: One part relational, one part XML," in *Proc. SIGMOD*, Baltimore,MD,USA,2005.
[9] R. Goldman and J. Widom, "DataGuides: Enabling query formulation and optimization in semistructured databases," in *Proc. 23rd Int. Conf. VLDB*, San Francisco, CA, USA, 1997.
[10] A. Marian and J. Siméon, "Projecting XML documents," in *Proc. 29th Int. Conf. VLDB*, Berlin, Germany, 2003.
[11] M. Rys, D. D. Chamberlin, and D. Florescu, "XML and relational database management systems: The inside story", in *Proc. SIGMOD*, Baltimore, MD, USA, 2005.

