A Novel Method for XML/SQL Query Translation Using Twice Indexing With Caching Approach

Ms. P.P.Talan Dr.V.M.Thakare Dr. P.P.Karde Dr. S.S.Sherekar

Abstract **:- SQL/XML has gained increasing interest whenever relational data has to be transformed to XML data, which is transferred, stored, or further processed in internet-based systems. The performance of query processing in a relational XML storage system is closely related to the query translation technique employed in the query mapping stage. For efficiently querying XML documents, many indexing methods have been proposed. In this paper ,study and analysis of four approaches regarding XML/SQL query translation are focused like query translation using intelligent path deviation ,query translation using interval encoding ,query evaluation using double indexes and generalizing and improving SQL/XML query evaluation. Each method have its own advantage and drawbacks. This paper survey these methods and propose an efficient twice indexing with caching which provides further modification to these methods and provides efficient query processing technique**

Keywords **: XML, SQL, Query Translation, indexing methods**

I. INTRODUCTION

The amount of data stored in XML has increased enormously as XML is adopted by a variety of fields as a standard of data exchange over the last decade. As XML has been, and continues to be adopted by a variety of fields, the amount of data stored in XML has increased enormously. A number of researchers have proposed to use the mature relational database technology to store and query XML data. This approach requires algorithms to map XML schemas, documents and queries, into their relational equivalents[1] . XML-to-Relational mappings are mainly split into schema-based and schema-less mapping schemes. Schema-less mapping schemes consider that XML schema is missing and generate a fixed schema in the target relational database. On the other hand, in schema based mapping schemes, XML schema is provided and used to generate a good relational schema which can vary depending on the input XML schema [2]. An issue that is worth discussing is the reuse of previously evaluated query.

There are hundreds or thousands of query patterns can be composed from an XML document. However, only few of them are meaningful and invoked frequently in practice. Each time when a query is initiated, the evaluation process is executed one more time. As a result, the system wastes lots of time doing the same work. Especially, when a query pattern is very complex, the total amount of time wasted could be enormous [3].

II. BACKGROUND

The algorithms proposed in the literature for translating XML queries into SQL queries are primarily intended to work on operational databases where the data is updated often. The problem of mapping recursive XML queries in the presence of recursive schemas has been reported in schema-less query mapping space. These query mapping algorithms are not applicable to the schema-based query mapping space as they use a fixed relational schema. Although analytical or static databases are not as common as operational databases, there are quite a few examples of them in real life such as dictionaries, legal texts, aircraft maintenance manuals and inventory catalogs which are used primarily for data retrieval rather than data update [1]. Three major techniques are identified to translate XML queries into SQL queries which are ID-based, path based and interval-based query translations. In ID-based approach, each XML element is associated with a unique ID in the relational database. In path-based approach, each XML element is associated with a path ID in the relational database representing the root-to-leaf path for that element. In interval-based approach, each XML element is associated with a pair of IDs in the relational database to denote the interval of node IDs in the subtreeunder that element. Developing a query translation algorithm that uses both the schema information and the intervalinformation is non-trivial [2]. To retrieve XML data efficiently, many indexing methods were proposed. There are hundreds or thousands of query patterns can be composed from an XML document. However, only few of them are meaningful and invoked frequently in practice. Each time when a query is initiated, the evaluation process is executed one more time. As a result, the system wastes lots of time doing the same work. Especially, when a query pattern is very complex, the total amount of time wasted could be enormous[3]. The task of XML data exchange is to restructure a document conforming to a source schema under a target schema according to certain mapping rules. The rules are typically expressed as source-to-target dependencies using various kinds of patterns, involving horizontal and vertical navigation, as well as data comparisons. The target schema imposes complex conditions on the structure of solutions, possibly inconsistent with the mapping rules. In consequence, for some source documents there may be no solutions [5]. XML-to-relational mapping storage is a principal settlement of XML data management and sharing. The approach maps XML Schema to relational schema is the core of XML schema mapping storage. In order to address the strong dynamics of XML, incremental mapping approach is urgently needed. An incremental approach of X-R mapping named "T-Schema"is proposed. It extends P-Schema approach andintroduces tag system and integration mechanism of relational schema increment. T-Schema realized incremental X-R mapping and integrate new relational schema and old one. This ensures the continued use of historical data which are important for many applications [6]. To evaluate clustering performance, two new performance evaluation methodologies, namely R_ClusterRatio and R_DocuRatio are introduced. It is motivated by the observations of relevant documents

distribution and the fact that the experiment data collection, IEEE CS corpus,do not provide classification information[7].XML is commonly supported by SQL database systems. However, existing mappings of XML to tables can only deliver satisfactory query performance for limited use cases. In this paper, we propose a novel mapping of XML data into one wide table whose columns are sparsely populated. This mapping provides good performance for document types and queries that are observed in enterprise applications but are not supported efficiently by existing work.

Based on the characteristics of the new mapping, we present rewriting optimizations that dramatically reduce the number of joins[8].Data exchange is the problem of taking data structured under a source schema and creatingan instance of a target schema, by following a mapping between the two schemas. There is a rich literature on problems related to data exchange, e.g., the design of a schema mappinglanguage, the consistency of schema mappings, operations on mappings, and query answering over mappings [9].Integration of multiple data sources is becoming increasingly important for enterprises that cooperate closely with their partners for e-commerce. OLAP enables analysts and decision makers fast access to various materialized views from data warehouses. However, many corporations have internal business applications deployed on different platforms. It introduces a model for heterogeneous data exchange based on XML. The system can exchange and share the data among the different sources[10].

III. PREVIOUS WORK DONE

Various methodes had been studied ,but none of those recursive query mapping algorithms could take advantage of a stored XML document to optimize query translation unlike the *XMLToSQL*algorithm proposed. Moreover, *XMLToSQL*algorithm translates the recursive XML queries to SQL queries in the presence of recursive schemas efficiently using intelligent path derivation. The *XMLToSQL*algorithm requires only the standard relational operators in an output SQL query which enables it to work with all database systems[1]. Another method is schema-Based XML-to-SQL Query Translation Using Interval Encoding, in schema-based XML storage space, there was no published interval-based XML-to-SQL translation algorithm in the literature. Developing a query translation algorithm that uses both the schema
information and the interval information was identified as an interesting open problem in "XML-to-SQL query
translation literature: The state of the art and open problems". The proposed query mapping algorithm Interval- XMLToSQL addresses this issue and provides an elegant and simple solution for the challenging recursive query translation problem as well[2]. A double-index system is proposed to index both XML documents and processed queries. The contributions of our work are as follows. Based on the NCIM (Node Clustering Indexing Method), which was developed by our research team, the proposed system first constructs the XML index and uses a convenient way to store/retrieve index into/from disk [3]. Based on a given SQL/XML query, the approach extracts the generated XML structure and derives a plain SQL query for collecting the relevant data. Using the result of the plain SQL query and the extracted XML structure, SAX events can be created, allowing to use any relational database as an input source for the generation of XML data. A compressed XML format can be generated directly, called succinct, whichprovides a fastergeneration of compressed XML data [4].

IV. EXISTING METHODOLOGIES

Some known methodologies are discussed here ,first method is based on intelligent path deviation provides following algorithm shown in figure 1 and uses PPT(path prefix tree) [1].

```
00 Algorithm XMLToSQL
01 Input: Path Expression P, APPT T_a
02 Output: SQL query sql
03 Begin
04  Let p_i, i=1,2, ...,n, be the set of all matching annotated paths of P in T_a
05  sql=∅
06  sql = ⋃_{i=1}^{n} ASPEToSQL(p_i)
07  Return sql
08 End

00 Procedure ASPEToSQL(Annotated Path Expression p)
01 Begin
02  Let p be /(e1,t1)/(e2,t2)/.../(ek,tk)
03  FromClause = "From "
04  WhereClause = "Where "
05  previous = null
06  For i=1 to k do /* Construct From Clause and Where Clause */
07    If t_i ≠ previous then
08      FromClause += "$t_i"
09      If previous ≠ null then
10        WhereClause += "$previous.(e_{i-1}.ID) = t_i.(e_i.parentID)"
11      End If
12    Else
13      If e_i.ID not used then WhereClause += "$t_i.(e_i.ID) is not null"
14    End If
15    previous = t_i
16  End For
17  sql="Select $t_k.(e_k.ID)" + FromClause + WhereClause
18  Return sql
19 End
```

Figure 1: Algorithm XMLTOSQL

Second method uses interval encoding scheme and provides two algorithm. The first interval-based XML-to-SQL query translation algorithm in the schema-based XML storage space. The proposed algorithm can be given as follows [2] :

```
00 Algorithm Interval-XMLToSQL
01 Input: Path Expression P, σ-mapping σ
02 Output: SQL-Query sql
03 Begin
04  Let P = a1/a2/a3/...an
05  FromClause="From"
06  WhereClause="Where"
07  For i=1 to n do /* Construct From Clause */
08    FromClause += "Tσ (ai) as Ti"
09  End For
10  For i=2 to n do /* Construct Where Clause */
11    If (ai = '//') then
12      WhereClause += "T_{i-1}.(a_{i-1}.ID) ≤ Ti.(ai.ID)
        AND T_{i-1}.(a_{i-1}.endID) ≥ Ti.(ai.endID)"
13    Else /* ai is '/' */
14      WhereClause += "T_{i-1}.(a_{i-1}.ID) = Ti.(ai.parentID)"
15    End If
16  End For
17  sql="Select T_n.(a_n.ID)" + FromClause + WhereClause
18  Return sql
19 End
```

Figure 2:Interval XMLTOSQLAlgorithm

The outline of Interval-XMLToSQL-C is shown in Figure 3 ,which remove unnecessary self join [2]. The another studied approach generates either compressed XML or SAX events as a result to given SQL/XML queries [4].Theproposed approach in [4] does not need an SQL/XML query engine, it provides a

useful technique to answer SQL/XML queries on SQL database systems that do not support the SQL/XML standard.

```
00 Algorithm Interval-XMLToSQL-C
01 Input: Path Expression P, σ-mapping σ
02 Output: SQL-Query sql
03 Begin
04   Let P = a1[n1]a2[n2]...an[nn]
05   Use σ to cluster P as c1c2...ck
06   FromClause="From"
07   WhereClauses="Where"
08   For i=1 to m do /* Construct From Clause */
09     If ni is the first element of a cluster then
10       FromClause += "So (ni) as Ti"
11     End If
12   End For
13   For i=2 to m do /* Construct Where Clause */
14     If ni is the first element of a cluster then
15       If (ai = '//') then
16         WhereClause += "Ti-1.(ni-1.ID) ≤Ti.(ni.ID)
           AND Ti-1.(ni-1.endID) ≥ Ti.(ni.endID)"
17       Else /* ai is '/' */
18         WhereClause += "Ti-1.(ni-1.ID) = Ti.(ni.parentID)"
19       End If
20     End If
21     If ni is neither first nor last element of a cluster then
22       WhereClause += "Sσ(ni).(ni.ID) not null"
23     End If
24   End For
25   sql="Select Tm.(nm.ID)" + FromClause + WhereClause
26   Return sql
27 End
```

Figure 3: Interval XMLToSQL C Algorithm

V. Analysis and Discussions

XMLToSQL algorithm is compared with SQLGen algorithm based on the number of derived paths for a set of givenrecursive queries. The number of derived paths is a crucial factor for the recursive query computation time. The input recursive queries are given against the XML document shown in Figure 4.

	Number of Derived Paths	
Path Expression	XMLToSQL	SQLGen[9]
/A//D3	2	4
/A/B2//D3	0	1
/A/C//D3	1	3
/A/B1//D3	1	1
//D2	1	3
/A/B3//D2	0	1
/A/B1//D2	1	1
/A//C	1	3
Total Number of Derived Paths	7	17
Average Number of Derived Paths	0.88	2.13

Figure 4: Number of derived paths

While the average number of derived paths is 0.88 with XMLToSQL algorithm, it is 2.13 with SQLGen algorithmwhich is considerably higher than that of XMLToSQL algorithm. In generation of uncompressed xml by just adding thedata query time and the processing time, the time needed to generate an uncompressed XML document can be obtained with proposed approach. A comparison with the naive approach shows that in nearly all cases of the evaluation, our approach of generating compressed XML is faster than generating uncompressed XML by using the SQL/XML query. This is the case for both, the Oracle implementation of SQL/XML compared to proposed approach using Oracle SQL queries .And the DB2 implementation of SQL/XML compared to proposed approach using DB2 SQL queries. The generation of compressed XML does not require to generate XML output from SAX events. when the target dataformat is compressed XML, the naive approach of compressing the SQL/XML query result requires additional time for a subsequent compression step. In this case, the performance improvement of proposed approach is even better than shown in figures shown above, as our approach already includes the creation of compressed XML in comparison to the naive approach which requires compressing the SQL/XML query result. If either of both data formats, compressed XML or uncompressed XML, is accepted for further processing, data archiving or data transfer, the comparison shows that in most cases, optimized approach generates compressed XML faster than the naive approach generates uncompressed XML.

VI. Proposed Methodology

How to improve the performance of the double-index method in case of a query miss in the query index is an importantissue to be focused. One way to get around this problem is to employ the concepts of cache design. Here we have suggested twice indexing with caching approach to address above problem ,the architecture can be shown in figure 5.
Here ,in twice indexing with caching approach ,the cache memory is provided to manage query miss in query index.

VII. Possible Outcome and Results:

The twice indexing with caching approach provides performance benefits and improve the result of system.

VIII. Conclusion

The twice indexing with caching approach is helpful in order to improve the overall performance of the system which in turn provide reliable ,efficient and optimum query translation and processing. And also provides the seamless working of ML/SQL query processing and storage.

IX. Future Scope

The caching methods mention in the twice indexing with caching approach should be modified for even better performance and results.

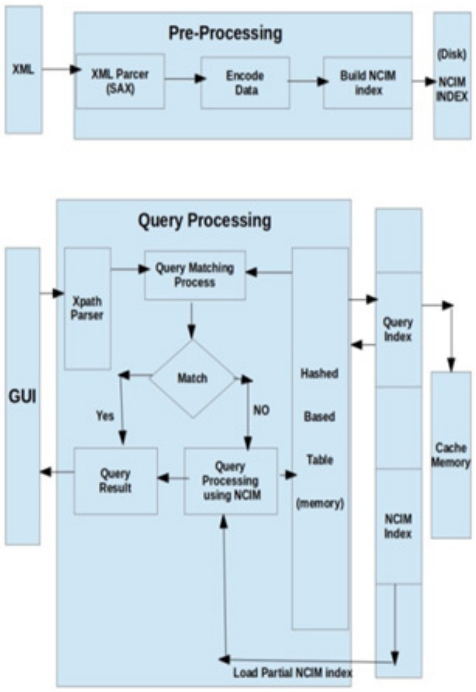

Figure 5: Proposed Twice indexing with caching approach

REFERENCES

[1] Mustafa Atay Winston-Salem State University, Paul S. Fisher Winston-Salem State University, "Optimizing XML to- SQL Query Translation for Analytical Databases Using Intelligent Path Derivation",ACMSE , Oxford, Mississippi USA,ISBN: 978-1-4503-0064-3, pp. 15-17 ,2010

[2] Mustafa Atay Winston-Salem State University , ArtemChebotko University of Texas- Pan American ,"Schema-Based XML-to-SQL Query Translation Using Interval Encoding", IEEE 2011 Eighth International Conference on Information Technology: New Generations , ISBN: 978-1-61284-427-5pp. 84-89, April 2011

[3] Wen-Chiao Hsu, I-En Liao and Pei-Hua Lu, "Supporting Efficient XML Query Evaluation Using
Double Indexes", IEEE 26th International Conference on Advanced Information Networking and Applications Workshops, ISBN:978-1-4673-0867-0pp. 197-202 ,March2012

[4] Stefan Bottcher, Dennis Bokermann, Rita Hartel , "Generalizing and Improving SQL/XML Query Evaluation", IEEE Eighth International Conference on Signal Image Technology and Internet Based Systems ,ISBN:978-1-4673-5152-2 ,pp. 441-449 ,Nov.2012

[5] MikołajBojanczyk ,Leszek A. Kołodziejczyk ,FilipMurlak, "Solutions in XML Data Exchange" ,Proceedings of the 14th International Conference on Database Theory ,ISBN: 978-1-4503-0529-7,pp. 102-113, 2011.

[6] YufengXu, ShilongMa,ShengweiYi,Yizhou Yan , "From XML Schema to Relations: A Incremental Approach to XML Storage",Computational Intelligence and Software Engineering (CiSE), IEEE International Conference, ISBN:978-1-4244-5391-7, pp.1-4, Dec. 2010.

[7] Zhong Min-Juan, Wan Chang-Xuan, Liu De-Xi Jiao Xian-Pei, "Research on XML Element Search Results Clustering", Management of e-Commerce and e-Government (ICMeCG), IEEE International Conference ,ISBN:978-1-4673-2943-9 ,pp. 285-290 ,Oct. 2012.

[8] Liang Jeff Chen, Philip A. Bernstein, Peter Carlin, DimitrijeFilipovic, Michael Rys, Nikita Shamgunov,
James F. Terwilliger, Milos Todic, SasaTomasevic, DraganTomic , "Mapping XML to a Wide Sparse Table",Knowledge and Data Engineering, IEEE Transactions ,ISBN: 978-1-4673-0042-1, pp. 1-16, April 2012.

[9] ZijingTan ,Liyong Zhang, Wei Wang, Baile Shi, "XML data exchange with target constraints" ,ELSEVIER Information Processing and Management ,PP. 465-483, Nov. 2012.

[10] HuanqinLia, JinfengLiub , "Research on Heterogeneous Data Exchange based on XML" ,SciverseScienceDirect International Conference on Solid State Devices and Materials Science ,volume:25 , pp. 1382-1387 ,April 2012.