International Journal of Engineering, Economics and Management
ISSN: 2319-7927, Volume 3, Issue 1

Review on Querying XML Schema with EX-up Technique for Accuracy

Ms. Trupti N. Mahale Prof. Santosh Kumar

Abstract — **Schemas are generally used to get the contents and structure of XML documents. As schemas are big in size and complex as well so they are difficult to handle manually. The ability to convert a single schema to query or a bunch of schemas is useful in many ways to get structural schema components. Hence proposing a new query language , named XSPath. This language is mainly for XML schema which basically works on graphical based representation of schemas. So that its capable for navigation and it easily allows the proper selection of nodes. Also proposing XPath / XQuery – based conversion that can be useful for the evaluation of XSPath queries. In this way the usability and efficiency of proposed technique is been presented within EX-up system.**

Key Words — **XML schema, XPath, schema querying, EXup, XSPath.**

I. Introduction

There are different schema languages which have been proposed (DTD, XML Schema etc.). After comparing their expressiveness it is been concluded that the most commonly used language is W3C recommendation XML Schema which totally depends on XML based representation[8]. XML became the easier way to represent data with loose structure. Furthermore it became universal data representation format with its flexibility. As well it was treated as medium for exchanging data between different applications. XML Schemas are employed for describing the type and structure of information contained in valid XML documents. However, XML Schemas are XML documents. Hence, the structure of a schema can be followed and its components (namely, element declarations, complex and simple type declarations, and complex types structures) can be optimized through a path language. A schema basically consists a set of elements and type declarations, where each element is associated with a locally defined or a global type, constraining its structure. Cardinality constraints can be specified both for operators and sub elements. Because of the hierarchical nature of XML Schema, navigational expressions on the schema structure are a natural means to retrieve its main components (namely, element declarations, complex and simple type definitions), to navigate in complex type structures, and to filter schema elements according to their values for properties like minimal and maximal occurrences of an element. In XQuery, different types of input and functions are defined in terms of regular expression types, but it is very easy to write queries that generate non-regular languages. Any type system for XQuery need to be dependent on a type inference process which gives the (possibly non-regular) output type of a query. This process, while compulsory and not avoidable, may significantly decrease the precision of the inferred types. This system, which relies on some over-approximating rules for expressions generally used in practice, like for iteration, for instance. Alternative way of undesired over-approximation is given by rules to type horizontal and upward XPath axes, for which the type is always defined. Another and more precise way for typing XQuery has been proposed and used as a basis for other proposals. This type of system has more pinpoint type inference, at the cost an improved computational complexity. Though these methods are well-known by the database and programming language communities, thorough, and complete analysis shows in which the two proposals differ in terms of optimization and complexity for type inference, is still not getting.

Schemas can be small and regular but the data are big and complex in some of the large domains as in aviation or weather information. As the verbose textual representation of XML schema is quiet complex so it is very critical to study it manually. To overcome this problem, logical representation of XML schemas came into picture. Some of the tools are also available to explore schemas into their graphical notation form. But it doesn't satisfy all needs of schema retrieval. The proposed technique focuses on a query language, named XSPath[1]. This language satisfies expressing retrieval needs on schemas without considering verbose XML schema syntax. As well it simplify retrieval tasks and offers flexibility to a query language. Schema contained in XML Schema is itself an XML document. To query different schemas, generally simple approaches could be used like XPath or XQuery. But these simple solutions does not reflect the complex expressions as per user's intention. Suppose, we want to denote the global employees element in the schema as shown in Fig 1.

The XPath expression */schema/ element [@name="employees"]* could be analyzed to retrieve the

International Journal of Engineering, Economics and Management
ISSN: 2319-7927, Volume 3, Issue 1

employees element. This expression is known as verbose and

```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="Employees" xmlns=""
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-
msdata">
  <xs:element name="Employees"
     msdata:IsDataSet="true"
       msdata:UseCurrentLocale="true">
          <xs:complexType>
           <xs:choice minOccurs="0"
            maxOccurs="unbounded">
            <xs:element name="emp">
             <xs:complexType>
              <xs:sequence>
               <xs:element name="ID"
               type="xs:int"
               minOccurs="0" />
               <xs:element name="Name"
               type="xs:string" minOccurs="0" />
               <xs:element name="City"
               type="xs:string" minOccurs="0" />
               <xs:element name="Dept"
               type="xs:string" minOccurs="0" />
               <xs:element name="Salary"
               type="xs:float" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
 </xs:schema>
```

it prefers simpler expression. We can extend this query as : " find the *city* element declaration within *employees*" which makes the XPath expression much more complicated but the extended form gives much more simplified version. To specify the occurrence of referral element within global element requires expressions over internal links. Navigation of such links is difficult in XPath.

Fig.1. Employees.xsd schema

Key feature of this new language is that the schema expressions are specified in two level graphical notation. It makes all expressions easier to navigate. This proposed architecture gives translation of XSPath expression into XQuery expression. Likewise implementing XSPath with translation is just to use XPath/XQuery engine rather than developing new featured engine. An improvement of the usability and efficiency of the givenapproach is covered up within the EX-up system. The approach of EX-up system provides a navigational XML query language with efficiency, easy way of working and simplified expression. Thus proposed a new language called XSPath which is specially designed to retrieve components of XML schema. This language simplifies the retrieval task as well it is offering a flexibility and power of query language over different tools. Main feature of this language is we can specify all the expressions in the form of two-level graphical notation, which can be termed as abstraction of the schemas. This abstract representation makes the expression specification more easier and as well it solves the gap between graph based and textual representation of schemas. The language in this way navigate throughout the nesting structure of element declarations. Transformation of XSPath expressions into XPath expressions is also key key feature of proposed language. In this way a static analysis is defined to reduce the size of the resulting expression with the help of transformation of each step. An efficient navigation approach to querying on XML schemas using the Ex-up technique which improves the usability of XML schema through XSPath language that gives accuracy with the help of translation and optimized algorithms.

This work is explained in different chapters. Most closely related work is surveyed in Chapter 2. Chapter 3 defines problem statement. Whereas chapter 4 gives brief idea about Architecture. Finally will Conclude the overall concept.

II. Related Work

Lots of contributions focus on the satisfactory problem of XPath queries. So the practical algorithm was presented for testing the satisfiability[3]. Next to that the problem of satisfaction of XPath expressions was investigated. To overcome this the new algorithm came into picture to test satisfiability of XPath queries but it was only for non-recursive DTD's and not for XPath axes[5][16].Data mining extracts useful information from the XML document. The association rule mining gives a tree format representation for the XML document. In existing system the TAR (Tree Based Association Rule) gives approximate results to the query, hence it decreases the answering ratio. Performance degradation is avoided by indexing technique to increases the efficiency of the result[2]. Next to that a novel and alternate approach was identified to optimize XPath queries by reducing wildcard steps[4]. More precise and efficient algorithms for checking of strings with and without modifications according to deterministic finite state automata. These algorithms are essential for efficient validation of the content models of elements[15]. The problem of checking whether the result of XML query satisfy the given type was focused[6]. A main feature of XQuery is its type system. The types of functions and input data in the form of regular expressions are defined in XQuery[17]. The XQuery has to depend on type inference process which approximates the type of output of query. This process is compulsory and unavoidable but it decreases the uniqueness of inferred types. To avoid this approximation critical issues regarding core of XQuery and its type language miniXQuery were studied[6].

Concerning XPath query, its first evaluation algorithm was presented which runs in polynomial time considering size of both data and query[7]. After considering the evaluation algorithm a schema dependent approach was identified to check XPath query in order to find semantic errors[8]. No results of queries shows that there is a semantic error in queries. As manual schema formation leads to errors, a user must needs to be familiar with the schema of the queried data and with implementation information of the query engine[18]. Thus, automatic optimization techniques have been developed and came into picture for decades in database management systems for the correctness of schemas operation. Hence focused on the problem of satisfiability for XML queries formulated in XPath language[9].

The different facilities provided by main DBMSs to work with XML Schema and their level of support to schema evolution and versioning was came into picture. The brief study regarding schema evolution and versioning was introduced[10]. Considering schema evolution the problem regarding it was investigated. Firstly different kind of changes were discussed that might be required on an XML Schema. Proceeding to that minimizing document revalidation was introduced to detect some parts of the document to invalidate by schema changes[11]. The main contribution in operating schema was revealed by introducing extension of SQL called SchemaSQL which gives the ability of uniform manipulation of data as well as meta-data in multi-database system. To achieve these features a special syntax and semantics of SchemaSQL was developed in such a way that it would extend the previous SQL syntax which gives (1)It supports both data as well as meta-data for querying. (2) It can represent data in database format in which data and meta-data can be interchanged. (3) It creates views whose schema is dynamically dependent on input. (4) SchemaSQL allows horizontal aggregation. (5) SchemaSQL supports for data and meta-data management in multi-database by interoperability[12]. The W3C introduced a query language SPAQL 1.1 for RDF data. It allowed queries to evaluate regular expressions over graph data. These expressions are different from regular expressions likewise they have numerical occurrence as well their semantics over graph are defined in non-standard manner[13]. XML graphs is simple and effective way of representing XML documents in program analysis. This graph notation technique have evolved through different range of applications. Hence presented an outline for XML graphs including its key properties against different kind of schema languages. To study the graph notation a survey was conducted for different applications like XML in JSP, Java and XSLT[14]. XML schemas evolved as different applications get integrated , but it is more important to check the validity of the document to relate one schema with other schema. After modification to the XML document, it is necessary to validate schema. Hence It was focused that how to conform an XML Schema or DTD to find out conformance with other schema[15].

III. Problem Statement

An efficient navigation approach to querying on XML schemas using the Ex-up technique which improves the usability of XML schema through XSPath language that gives accuracy with the help of translation and optimized algorithms. XML schema gives a set of different facilities to give the structure and contents of XML documents. Path expression is the main part of XSPath. The notation of XPath and XSPath expression is same which consists of sequence of all steps.

IV. Overview

Since a schema is an XML document, one could directly use the XPath language for the specification of navigational expressions on schemas. However, this would result in the specification of complex expressions that do not return what users expect when formulating them. Moreover, path expressions over internal links are not supported and navigating through links. XML schema explanation and querying is to provide a logical model of XML schema proposed different approaches. XPath introduced the concept of retrieval of elements on the basis of type of elements and attributes.[13] A different approach to navigate the structure of XML schemas within Java Programs based on APIs[2]. Considering all previous specifications proposed a query language, named XSPath, specifically for XML schema that gives logical graph-based representations of schemas, which enables the navigation, and allows the selection of nodes. Also proposed system specifies XPath / XQuery-based transformation for the evaluation of XSPath queries. Ex-up technique is used for translation and evolution of XSPath expression. The overall architecture is shown in the following Fig. 2

As shown in above diagram the prerequisites of the system are; there should present both XML and the related Schema files. The schema file must be of the same type of the XML file. The system will verify both files that both are related to each other because the XML file and its Schema file are those contents which needs to work properly. After verification those files gets loaded with the help of parser. After completion of all the above mentioned steps proposed XSPath system loads XSPath syntax and its co-related functions. Now the system is ready to use. User can give a query dependent on the XML file. Query will retrieve the required information from XML file. But the syntax must be correct to operate on file. The fired query needs to travel through whole data structure to get output. The given system will check the syntax of query as it is in the proper format or not. If the query is in proper format then it proceeds to the system for giving output. In this way the working of the whole system is been defined with the help of this architecture.

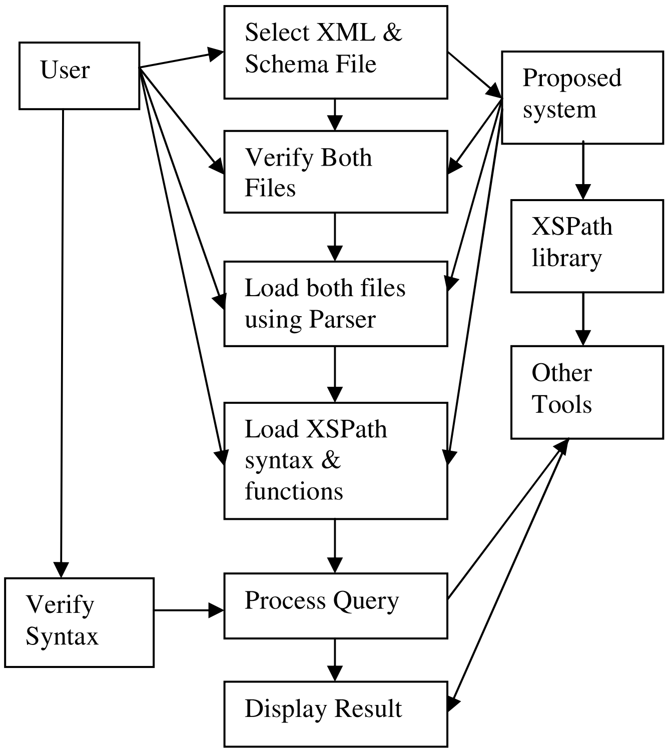

Fig. 2. Architectural view of XSPath system

As shown in above diagram the prerequisites of the system are; there should present both XML and the related Schema files. The schema file must be of the same type of the XML file. The system will verify both files that both are related to each other because the XML file and its Schema file are those contents which needs to work properly. After verification those files gets loaded with the help of parser. After completion of all the above mentioned steps proposed XSPath system loads XSPath syntax and its co-related functions. Now the system is ready to use. User can give a query dependent on the XML file. Query will retrieve the required information from XML file. But the syntax must be correct to operate on file. The fired query needs to travel through whole data structure to get output. The given system will check the syntax of query as it is in the proper format or not. If the query is in proper format then it proceeds to the system for giving output. In this way the working of the whole system is been defined with the help of this architecture.

CONCLUSION

Defined a type analysis for the given language that optimizes schema conversion with translation algorithm independently. As well given a specification of the conversion of XSPath to XPath and its schema-independent transformation to XQuery. XPath evoluation algorithm is introduced to derive correctness and complexity of result conversion. In this way Ex-up is an optimized technique for translation and evolution of XSPath expression.

ACKNOWLEDGEMENT

The authors would like to acknowledge Computer Engineering department, SITRC and all the people who provided with the facilities being required and conductive conditions for completion of the review paper.

REFERENCES

[1] F.Cavalieri, G. Guerrini, M. Mesiti "XSPath: Navigation on XML Schemas Made Easy", IEEE Transactions on Knowledge and Data Engineering,vol.26, no. 2,2014

[2] D.Karthiga, S.Gunasekaran," Optimization of Query Processing in XML Document Using Association and Path Based Indexing", International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 2, April 2013

[3] Z. Bellahsene, A. Bonifati, E. Rahm, eds., Schema Matching and Mapping. Springer, 2011.

[4] C.Y. Chan, W. Fan, and Y. Zeng, "Taming XPath Queries by Minimizing Wildcard Steps," Proc. 30th Int'l Conf. Very Large Data Bases, pp. 156-167, 2004.

[5] D. Colazzo, G. Ghelli, and C. Sartiani, “Schemas for Safe and Efficient XML Processing,” Proc. IEEE 27th Int’l Conf. Data Eng., pp. 1378-1379, 2011.
[6] D. Colazzo and C. Sartiani, “Precision and Complexity of XQuery Type Inference,” Proc. 13th Int’l Symp. Principles and Practices Declarative Programming Languages, pp. 89-100, 2011.
[7] G. Gottlob, C. Koch, and R. Pichler, “Efficient Algorithms for Processing XPath Queries,” Proc. Int’l Conf. Very Large Data Bases, pp. 95-106, 2002.
[8] J. Groppe and S. Groppe, “Filtering Unsatisfiable XPath Queries,” J. Data Knowledge Eng., vol. 64, no. 1, pp. 134-169, 2008.
[9] J. Groppe and S. Groppe, “Satisfiability-Test, Rewriting and Refinement of Users’ XPath Queries According to XML Schema Definitions,” Proc. 10th East European Conf. Advances in Databases and Information Systems, pp. 22-38, 2006.
[10] G. Guerrini and M. Mesiti, “XML Schema Evolution and Versioning: Current Approaches and Future Trends,” Open and Novel Issues in XML Database Applications, E. Pardede, eds., Idea Publishing, 2009.
[11] G. Guerrini, M. Mesiti, and D. Rossi, “Impact of XML Schema Evolution on Valid Documents,” Proc. Seventh Ann. ACM Int’l Workshop Web Information and Data Management, 2005.
[12] L. Lakshmanan, F. Sadri, and S. Subramanian, “SchemaSQL – An Extension to SQL for Multidatabase Interoperability,” ACM Trans. Database Systems, vol. 26, no. 4, pp. 476-519, 2001.
[13] K. Losemann and W. Martens, “The Complexity of Evaluating Path Expressions in SPARQL,” Proc. 31st Symp. Principles Database Systems, pp. 101-112, 2012.
[14] A. Møller and M.I. Schwartzbach, “XML Graphs in Program Analysis,” Science Computer Programming, vol. 76, no. 6, pp. 492515, 2011.
[15] M. Raghavachari and O. Shmueli, “Efficient Schema-Based Revalidation of XML,” Proc. Ninth Int’l Conf. Extending Database Technology, pp. 639-657, 2004.
[16] S. Groppe and S. Bottcher, “Schema-based Query Optimization for XQuery Queries”,pp.80-95,2005
[17] M. Montazerian, Peter T. Wood, Seyed R. Mousavi, “XPath Query Satisfiability is in PTIME for Real-World DTDs”, Volume 4704, pp 17-30, 2007
[18] M. Raghavachari, O. Shmueli,” Efficient Schema-Based Revalidation of XML”, EDBT, Volume 2992, pp 639-657, 2004

AUTHOR’S PROFILE

	Trupti N. Mahale B.E. Computer (Savitribai Phule Pune University) PG Student, Savitribai Phule Pune University, SITRC College, Nashik-422213 Trupti.mahale31@gmail.com
	Prof. Santosh Kumar Assistant Professor, Savitribai Phule Pune University, ME Coordinator, Computer Department, SITRC, Nashik. Santosh.kumar@sitrc.org ISTE Member