ISSN: 2321-7782 (Online)
Volume 3, Issue 6, June 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

XML Schema Routing Translation: XPlus Path Discovery

Pallavi S. Samdolikar[1]
Department of Computer Engineering
B.S.I.O.T.R. Wagholi
Pune - India

Prof. Sanchika A. Bajpai[2]
Department of Computer Engineering
B.S.I.O.T.R. Wagholi
Pune - India

Abstract: ***A Markup language has been mainly chosen for data representation, storage, and exchange in many different arenas to optimize the specific work. A query language need to progress both schema and path specification. Generating X+ for extracting the xml attributes and elements from XML schema. A non-constraint development on XML path schema transformation provides a content retrieving process like tool area. This language development tool will increase the accuracy on repossessing work. It works on tree representations of schema, on which it enables the navigation, and allows the selection of nodes and querying on schema. Also XQuery based translation proposed that can be exploited for the evaluation of queries.***

Keywords: XML Schema, XSD, XML Validation, XPath, XQuery, XSPath query language.

I. INTRODUCTION

XML [Extensible Markup Language] is designed to describe data on web. It defines a set of rules for encoding document in format that is both human readable and machine readable.

A. XML Processing: XML Schema is an abstract collection of metadata, consisting of a set of schema components: chiefly element and attribute declarations and complex and simple type definitions. These components are usually created by processing a collection of schema documents, **XML Schema Definition [XSD]** which enables you to define structure and data types for xml document [4].

***B. Validation*:** When an instance document is validated against a schema document (a process known as assessment), the schema document to be used for validation can either be supplied as a parameter to the validation engine, or it can be referenced directly from the instance document using following components/attributes.

***a. Element declaration*:** It is an association of a name with a type definition. It will be simple or complex, an (optional) default value and a (possibly empty) set of identity-constraint definitions. Element declarations define the properties of element and consist of element name and target namespace.

b. Attribute declaration: It defines the properties of attributes. Again the properties contain the attribute name and target namespace.

c. Model group and attribute group definitions: These are essentially macros: named groups of elements and attributes which can be reused in different type definitions.

d. An attribute use: Represents the association of a complex type and an attribute declaration. It indicates whether the attribute is mandatory or optional when it is used in that type.

e. An element particle: Again it represents the relationship of a complex type and an element declaration, and indicates the minimum and maximum number of times the element may appear in the content. Element declaration, attribute declaration, Model group and attribute group, attribute use and element particle.

C. Data model representation: The After validation of schema documents, xml document structure can be expressed in terms of data model which contain following components:

a. Vocabulary: element and attribute names

b. Content model: relationships and structure

c. The data types.

This set of information is called the Post-Schema-Validation Info set (PSVI) which gives output a valid XML document and its "type" and facilitates treating the document as an object, using object-oriented programming (OOP) paradigms. The schema can be used to generate code, referred to as XML Data binding as given in [4]. It generates human readable documentation of xml structure. Different query languages are used to query the schemas like XPath[9], XQuery[10], XSPath[1].

XS-Path [1] uses tree representation instead of xml schema definition because in case of large XSD, manual inspection is quite complex. XSPath uses location path to navigate around the schema. A location path consists of a sequence of location steps. Each location step has axis, node test and predicates. An XS-Path expression is evaluated with respect to a context node. But it is failed to extract complex schema reference files easily.

Proposing verbose have advanced query pre-processing module for extracting all elements from the xml schema. Discovering the way of tags commencement will obtain the path of root element to child element even the complex type of parsing will be made.

II. LITERATURE REVIEW

A. Updating xml schema and Associated Documents through EXup

Web data are mostly specified in XML format and the need often arises to update their structure, commonly described by an XML Schema. After data modification different issues are occurred with the documents that need to be faced. XS Update [13] is a language that easily identifies parts of an XML Schema, applies a modification primitive on them and finally defines an adaptation for associated documents. EX up is the corresponding engine which processes schema modification and document adaptation statements. This language is easy to use for updating schemas. Its semantics are well specified and it copes with all the effects of schema updates on related files. It provides user interface to get details on web use and local use. One important advantage of it is XQU expressions are also used to validate XML documents or parts of them. But it has some drawbacks. Its functionality failed to be propagated to a third party repository holding their master copy. So it led to be data lost. Old documents may need to be (incrementally) re-validated against the new schema, and, if they are not valid anymore, processes have to adapt a new schema.

B. Scalable, Memory-Efficient Data Extraction from Web Applications

To address key requirements of web extractions like,

I. Interact with sophisticated web application interfaces,

II. Precisely capture the relevant data for most web extraction tasks,

III. Scale with the number of visited pages,

IV. Readily embed into existing web technologies

A language OXPath, an extension of XPath is introduced in [12]. OX-Path allows the simulation of user actions to interact with the scripted multipage interfaces of web applications. OX-Path's page-at-a-time evaluation guarantees memory use independent of the number of visited pages, yet remains polynomial in time as defined in [12]. It identifies data for extraction

and assembles it in to hierarchical records, regardless of its original structure. It guarantees memory use independent of the number of visited pages. This is the most important advantage of OXPath. But it has disadvantage that required data failed to extract from existing, human-oriented user interfaces so automation could not able to get process.

C. Visualizing and navigating ontology

An ontology summarization method has been entirely used to support displaying key concepts and using these as the basis for further exploration can be seen as assisting information foraging from ontology [11]. In particular, the flexible set of options provided by KC-Viz for manipulating the visualization enables the user to construct a view on the ontology that allows them to compare the information scent of different paths through the ontology and control how they pursue these paths. Key concepts use a number of factors to estimate the importance of a particular class and therefore provide means for estimating the potential profitability of an information scent.

D. Retrieving ontology fragments

In Onto Path [8], User defines concept definition and properties between it. Onto Path language have very simple syntax and aims for non-expert user as well XSPath [1] is also aimed to design simple enough so that non-expert user also can able to understand it. Onto Path is used in applications that are not XML based. To implement this graphical database is used. Ontology fragments are accessed and retrieved using Prot´eg´e OWL plug-in. But it has some disadvantage like it will not be able to handle medium sized Ontologies. Current one does not follow any formalism. For concrete application some modules are not useful.

III. EXISTING SYSTEM

XML markup language used to represent data on web. Schema documents (XSD) defines the rules to constrain the type and structure of the xml. XSD can be quite big and complex and thus, difficult to be accessed manually. The ability to query a single schema, a collection of schemas or to retrieve schema components that meet certain structural constraints is provided in XSPath query language. It is specifically tailored for XML schema that works on tree representations of schemas. The most important kind of expression in XSPath is a location path. A location path consists of a sequence of location steps. Each location step has axis, node test and predicates. An XS-Path expression is evaluated with respect to a context node. An Axis specifies the direction to navigate from the context node. The node test and the predicate are used to filter the nodes specified by the axis. It also provides XPath/XQuery-based translations that can be used for the evaluation of XSPath queries. An extensive evaluation of the usability and efficiency of the XS-Path is finally presented within the EXup system [13].

A. Limitations:

a. Dependency on path expression needs specification lead to expect flexible query language.

b. It failed to query on composition of schemas.

c. Complex schema reference files are not easily recovered.

d. An instance of application context in graphical work needs more schema advancement.

IV. SYSTEM ARCHITECTURE AND DESCRIPTION

In this paper an additional XPath discovery from the xml schema definition is provided to give the complex schema reference files easily recoverable and using that content for the future application work. XML tags can be easily retrieved by avoiding dependency constraints on development. Initially xml file has been given as text file format as input. After that parsing of xml tags are done. In this process all attributes and contents have been extracting. Depending on file size different task are performed by a process like taking input file, parsing will be directing for getting the contents sequentially. Simultaneously getting and putting progress will be appearing and updating it on database also. Iteration flowing on XML tree i.e. full markup

schema information for getting required content inside the model. Based on uncertain redundancy value will be converted into structured elements. Translation work is used in the case of modification such as adding information or deleting something from that. At this time translating text file into xml file sorting can be progress. At the end validation process will occur. Documents are only considered valid if they satisfy the requirements of the schema with which they have been associated. A validated xml file and its equivalent file contents with the correct attributes and elements can be gathered into the database.

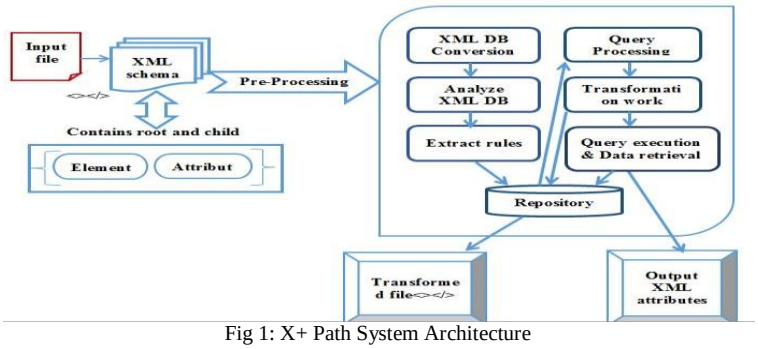

Fig 1: X+ Path System Architecture

A. *Modules And Description*

XML Pre-Processing

a. XML data can be given as input and it contains markup content, attributes, elements, declaration with comments in text format inside the tags.

b. The user defined tags included on any xml file whether it is simple or complex schema definition.

c. By tokenizing all tags in the name of parsing each and every content will be gathered and put it on other file format for the further progression.

XML root-child Extraction

a. From the parsing work inside the xml tags operation will be continue.

b. Starts from the parent to child element with its attribute can be retrieving step by step precisely.

c. By navigating all elements from the root to end tag can be shown to be update sequentially.

d. Initially updating the attributes with the relevant relational model in the database getting and showing on to the iterator.

Schema Transformation Framework

a. A framework which will progress the complex files into user recommended file format.

b. This will provide capabilities for instance: declaring variables and functions, iterating over sequences even though the program syntax are quite different.

c. The model is logic driven, for different cases in contrast, XSLT is data-driven that is push processing model where certain conditions of the input document trigger the execution of templates rather than the code executing in the order in which it is written.

Query translation

a. By viewing the content on the framework, information can be add or delete with the input file form.

b. The approach to translating XML documents to data has been to extract the translatable text and attributes into an external, typically proprietary format where translation memories matches are performed on the data.

c. On completion of the translation process the newly translated sentences are written to traditional non-standard translation memory repositories.

d. XML namespace is used to map a text memory view onto a document. This process is called segmentation.

e. The text memory view works at the sentence level -- the text unit. Each individual text unit is allocated a unique identifier.

f. This unique identifier is immutable for the life of the document. As a document goes through its life cycle the unique identifiers are maintained and new ones are allocated as required.

Data Retrieval

a. In this module user can view all the extracted content of xml schema.

b. User can see navigated path of searched element.

c. Also can see transformed file that can be viewed as xml DOM [5]. The entire XML transformation and query modification framework will induce the usability on schema definition.

d. Comparison can be made with the existing path language navigator with its properties.

e. Performance has been improved by increasing the retrieving data easily for the application.

f. Consuming time for transforming and query processing framework has been lower than others.

B. Mathematical Model

Input:

Consider {<root><child></root>} includes complex XML file content

This implies

1. *Root value = R*
2. *Child value = C*

⇨ *m attributes, n elements*

Process:

F (Extraction) *– All elements and attributes are extracted from xml schema.*

$\{<><></></>\} \rightarrow \{R \rightarrow C_{\rightarrow \{m\ attributes\}\{n\ elements\}}\}$

F (Navigation) *–*

$R > C >_{c>c>c>c}$ *creating way to destined tag required*

F (Modification) *—*

a. *R to* R_{end} ➔ $\pm$ *{insert, delete}*

b. $\{R^{+\ to}\ R^{-}\}$ ➔ *{+n, +m, -n, -m}*

c. $\sum$ *{± values} U {outdated (R, C)}* ➔ *designated into modified elements =* ➔ *Tags updated*

d. *Extracted Tags values can be change if user can add or delete any information content*

F (Transform) *—*

{R value, C value, Element value, Attribute value} => {<Root><Child></Root>}

Output:

F (view) *Document retrieval – View {xml} and {modified text file content}*

V. PERFORMANCE EVALUATION

Efficiency of the system have been extensively evaluated to assess the effectiveness of X+ Path. The results of this experimental evaluation are given in the following sections.

A. Results

X+ Path is providing an additional of XPath discovery in order to give the complex schema reference files has been easily recovering and using that content for the future application work. Results of X+ Path a query processing tool are shown in following snapshots according to module execution sequence.

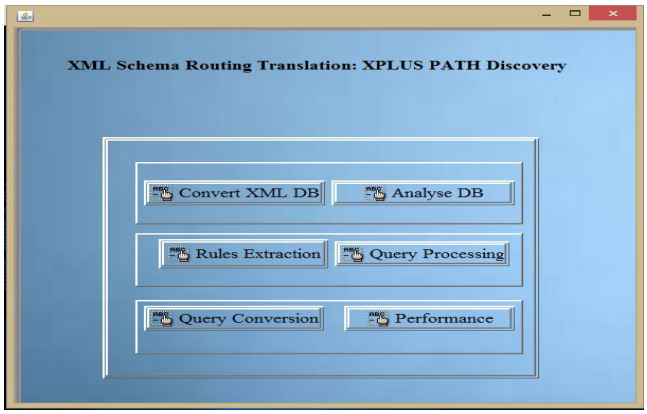

Fig. 2 Home Page of XPlus Path Discovery

At the end of project, performance of the X+ Path system is compared with existing system XSPath.

Following parameters are used to measure the performance.

1. Translation time : Time required to translate query in XQuery.
2. Response time: Time required executing XPlus Quey.

Proposed system compared with existing system using translation time parameter as shown in graph.

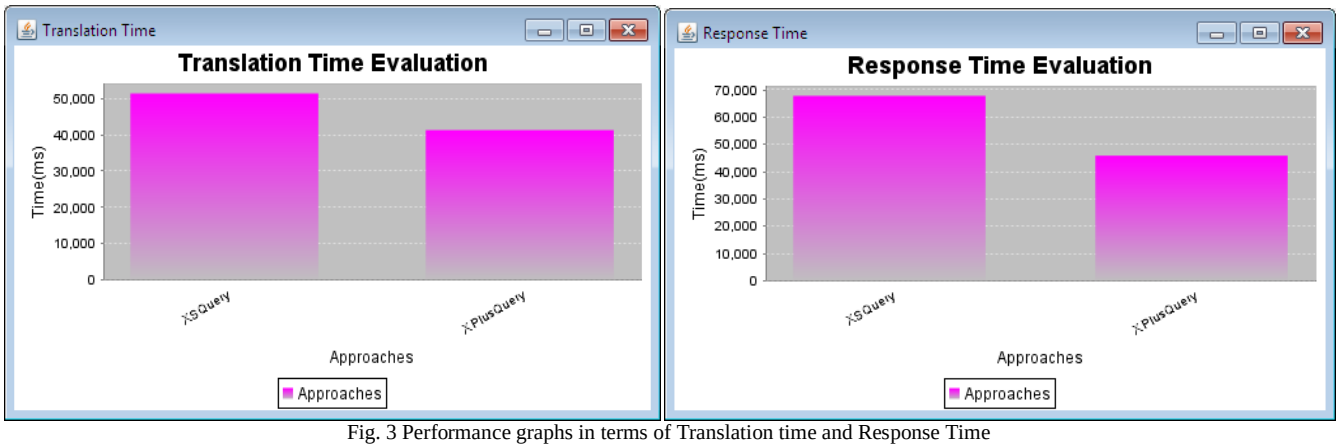

Fig. 3 Performance graphs in terms of Translation time and Response Time

VI. CONCLUSION

On conclude the markup language proposal can be basically performed on simple and complex XML file. An exact schema comparing would be stimulating for application scenarios invent for the x plus discover language. X+ Path easily recovers complex schema reference files. Thus query with transformed retrieval of XML documents has been extensively produced. This framework will support full-text predicates, full-fledged attributes and Schema document for the xml declaration and query process for the entire development. On evaluating this path discover work for the xml document with old version of XS-Path has been compared and contrast for the variances.

ACKNOWLEDGEMENT

The author would like to thank Prof. S. A. Bajpai (PG Guide) as well as Prof. G. M. Bhandari (HOD), Prof. A. C. Lomte (PG Coordinator) those gives us a good guideline for Research Work throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in Completing this Work.

References

1. Federico Cavalieri, Giovanna Guerrini and Marco Mesiti, "XSPath:Navigation on XML Schemas Made Easy" IEEE Transactions on knowledge and data engineering, vol. 26, no. 2, February 2014
2. J. Clark and M. Murata, "RELAX NG Specification," http://relaxng.org/spec-20011203.html, 2001.
3. 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.
4. "XML Schema," W3C, second ed., 2004.
5. "Document Object Model (DOM)," W3C, 2004.
6. M. Raghavachari and O. Shmueli, "Efficient Schema-Based Revalidation of XML," Proc. Ninth Int'l Conf. Extending Database Technology, pp. 639-657, 2004.
7. M. Murata, D. Lee, M. Mani, and K. Kawaguchi, "Taxonomy of XML Schema Languages Using Formal Language Theory," ACM Trans. Internet Technology, vol. 5, no. 4, pp. 660-704, 2005.
8. E. Jimnez-Ruiz, R. Berlanga, V. Nebot, and I. Sanz, "OntoPath: A Language for Retrieving Ontology Fragments," Proc. OTMConfederated Int'l Conf. Move to Meaningful Internet Systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I, pp. 897-914, 2007.
9. "XML Path Language (XPath) 2.0," W3C, 2007.
10. "XQuery 1.0: An XML Query Language (Second Ed.)," W3C, 2010.
11. E. Motta, P. Mulholland, S. Peroni, M. d'Aquin, and J. Gmez-Prez,V. Mendez, and F. Zablith, "A Novel Approach to Visualizing and Navigating Ontologies," Proc. 10th Int'l Conf. Semantic Web, vol. 1, pp. 470-486, 2011.
12. T. Furche, G. Gottlob, G. Grasso, C. Schallhart, and A.-J. Sellers,"OXPath: A Language for Scalable, Memory-Efficient Data Extraction from Web Applications," Proc. VLDB Endowment,vol. 4, no. 11, pp. 1016-1027, 2011.
13. F. Cavalieri, G. Guerrini, and M. Mesiti, "Updating XML Schemas and Associated Documents through EXup," Proc. IEEE 27th Int'l Conf. Data Eng., pp. 1320-1323, 2011.

AUTHOR(S) PROFILE

Ms. Pallavi S. Samdolikar, received the BE degree in Information Technology and M.E. degrees in Computer Engineering from B.S.I.O.T.R. Wagholi, Pune University, Pune, India. in 2013 and 2015, respectively. She had been an Assistant Professor with the Information Technology Department, AITRC, Vita, Sangli, Shivaji University, Kolhapur for 02 years. She is the author of 01 international journal. Her research interests include web technology and data mining.

Prof. Sanchika A. Bajpai, received the M.Tech. degree in Computer in 2011. She is currently working as an Assistant Professor with the Computer Engineering Department in B.S.I.O.T.R Wagholi, Pune, Savitribai Phule University, Pune since 04 years. She is the author of many international journals. Her research interests include data mining.

