Volume 4, Issue 11, November 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Optimized Xml Routing Translation: X plus Path Discovery

Pallavi Samdolikar
Department Of Computer Engineering,
B.S.I.O.T.R Wagholi Pune, India

Sanchika Bajpai
Department Of Computer Engineering,
B.S.I.O.T.R. Wagholi Pune, India

Abstract - In this paper, the concept of XML schema and path specification is progressed. Key technology for extracting the xml attributes and elements dynamically and expressing xml schema to reduce user anonymity is researched. Different ways to support rational illustration of schemas, with coherent and distinct details about the required formulation are surveyed.

Keywords: XML Schema, XPath, XSPath query language.

I. INTRODUCTION

In the web world data representation and data transformation are important areas. For the purpose of data representation HTML (Hypertext Markup Language) is used. For Transforming data XML (Extensible Markup Language) is used. Structure of XML is defined by XSD document. XSD contains information that defines how business or application information xml document should carry or represent. A schema document contains rules and regulation for xml document. A schema is an abstract collection of metadata, consisting of a set of schema components. XML document are validated against XSD to check whether particular XML follows rules of that XSD or not. XSD Contains following main content

A. Element:

This component defines tag in xml. It will contain properties of that element and constraint on that element. For example 1, In XSD, we define book-name element then that element may contain constraint like book-name length should not be more than 50character. For example 2, Suppose Fruit element has been defined in XSD then it may contain attribute like color of fruit. It will also contain definition of another sub element. This tag contains properties of element like name, type.

B. Attribute:

This component is used to give extra information related to element. This component is optional. In XSD, while defining attributes various properties are used like name, type, use.

C. Type:

XSD contains built in types like string, int etc. XSD also contain information for new type using existing type. This newly defined type is used in element and attributes declaration. This type definition may be simple or complex.

D. Model group and attribute group definitions.

These are essentially macros: named groups of elements and attributes that can be reused in many different type definitions.

An **element particle** similarly 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. As well as element particles, content models can include **model group** particles, which act like non-terminals in a grammar: they define the choice and repetition units within the sequence of permitted elements. In addition, **wildcard** particles are allowed, which permit a set of different elements.

XSD contains reference for another XSD component through namespaces. XSD component can be declared in different file and later on those file will be imported in main file.

Before XSD, DTD was used for XML element definition.

1. DTD can not be parsed using XML parser. As XSD is an xml document it can be parsed using XML parser
2. In XSD, two elements may have same name in different context. But DTD can not contain two element with same name in two different context as all element in DTD are global
3. In DTD, there is no way to define data type of element but in XSD can specify what type of data element will contain.

Schemas can be small and simple in application contexts where data are quite regular. But for schemas which contains more than 2000 lines, their manual inspection is quite complex, also because of the verbose textual representation of XML schema.

For better utilization of XSD, there should be proper language for reading component of XSD. X-Path [1] or X-query [2] are used to read xml document. Same can be used to query schemas. But it will results in specification of complex expressions that do not reflect user intuitions in formulation. So new language is defined XSPath [8] specifically for retrieval of XML schema components.

The most important kind of expression in XS-Path 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 [8] 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.

II. RELATED WORK

Our work takes advantage of concepts developed in the context of schema querying, extraction of elements and attributes and exploration for various kinds of data and schema modification based XML processing. In what follows, we survey the most closely related work.

A. Updating XML Schemas and Associated Documents:

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 [3] 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) revalidated against the new schema, and, if they are not valid anymore, processes have to adapt a new schema.

B. Memory-efficient Data Extraction from Web Applications:

To address key requirements of web extraction like, (1) Interact with sophisticated web application interfaces, (2) Precisely capture the relevant data for most web extraction tasks, (3) Scale with the number of visited pages, and (4) Readily embed into existing web technologies A language OXPath, an extension of XPath is introduced in [4]. OX-Path allows the simulation of user actions to interact with the scripted multi-page 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 in [4].

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 Ontologies:

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[7]. 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. In addition, sub tree summaries provide a topological clue as to the potential profitability of a class.

All users were involved in the context of community-wide ontology development activities due to its operable work. It has often performed better in evaluation scenarios than the graphical alternatives. It has drawbacks also. Tree maps are not necessarily effective in supporting an understanding of topological structures. The lack of effective mechanisms in visualizing and navigating ontologies needs to support selective parts. Affecting performance on subjective view in the usability of ontology engineering tools.[7]

D. Retrieving Ontology Fragments:

In Onto Path[5], concept definition and properties between different concepts are allowed to define by user. Syntax and aims of Onto Path language are very enough for non-expert user as well XSPath [8] is also aimed to design simple enough so that non-expert user also can able to understand it. Onto Path is used in non-XML application. Graphical database is used to implement Onto Path. Ontology fragments are accessed and retrieved using Protégé 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.

This extract consistent, closed, and useful ontology Fragments, suitable for knowledge exploration purposes. Proposed good formalisms to automatically define ontology modules and their connections. Some modules may not be useful for concrete applications. Currently the maintained reference does not follow any formalism.[5]

E. Complexity calculation for Path expression in SPARQL:

Linked data on web is represented by RDF. RDF (Resource Description Framework) is proposed by W3C [6]. It has improved way data is read by computer from web. It has introduced new way of data querying on web. Edge labeled graph is used to show linked data in RDF. For property path W3C semantics are formalized. Problems of query evaluation are investigated here. But it has issues like property path and memory management is not handled correctly. Users always are not interested in labels and occurrences on edge.

III. 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. Thus query with transformed retrieval of XML documents has been extensively produced. Our framework will support full-text predicates, full-fledged attributes and Schema document for the xml declaration and query process for the entire development.

REFERENCES

[1] "XML Path Language (XPath) 2.0," W3C, 2007.

[2] "XQuery 1.0: An XML Query Language (Second Ed.)," W3C, 2010.

[3] F. Cavalieri, G. Guerrini, and M. Mesiti,(2011) "Updating XML Schemas and Associated Documents through EXup," Proc. IEEE 27th Int'l Conf. Data Eng., pp. 1320-1323,

[4] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, and A.-J. Sellers, ,(2011) "OXPath: A Language for Scalable, Memory-Efficient Data Extraction from Web Applications," Proc. VLDB Endowment, vol. 4, no. 11, pp. 1016-1027.

[5] E. Jimnez-Ruiz, R. Berlanga, V. Nebot, and I. Sanz, (2007)"OntoPath: A Language for Retrieving Ontology Fragments," Proc. OTM Confederated Int'l Conf. Move to Meaningful Internet Systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I, pp. 897-914.

[6] K. Losemann and W. Martens, (2012)"The Complexity of Evaluating Path Expressions in SPARQL," Proc. 31st Symp. Principles Database Systems, pp. 101-112,

[7] E. Motta, P. Mulholland, S. Peroni, M. d'Aquin, and J. Gmez-Prez, V. Mendez, and F. Zablith,(2011) "A Novel Approach to Visualizing and Navigating Ontologies," Proc. 10th Int'l Conf. Semantic Web, vol. 1, pp. 470-486.

[8] Federico Cavalieri, Giovanna Guerrini, Macro Mesiti, (2014)" XSPath: Navigation on XML Schemas Made Easy"

