IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 05, Issue 04 (April. 2015), ||V1|| PP 23-27

Querying the schema's using xspath in xml language

T. Vamsi Vardhan Reddy[1], D.V. Subbaiah. M.Tech, (Ph.D)[2]
1. M.Tech Student 2. Associate Professor

Abstract: - Schemas are often used to constrain the content and structure of XML documents. That is by using previous one we can't navigate an XML document in an easy way and quickly. We propose a query language, named XSPath, specifically tailored for XML schema that works on logical graph-based representations of schemas, on which it enables the navigation, and allows the selection of nodes. However, XML Schemas are themselves XML documents. Thus, the structure of a schema can be navigated and its components can be retrieved through a path language. XSPath is a language tailored for specifying path expressions on XML Schemas.

***Index Terms**: - XML schema, XPath, schema querying.*

I. INTRODUCTION

An XML document contains a prolog and a body. The prolog consists of an XML declaration, possibly followed by a document type declaration. The body is made up of a single root element, possibly with some comments and/or processing instructions. For instance, an XML document may be dynamically compiled from information contained in a database.

The first few characters of an XML document must make up an XML declaration. The declaration is used by the processing software to work out how to deal with the subsequent XML content. A typical XML declaration is shown below. The encoding of a document is particularly important, as XML processors will default to UTF-8 when reading an 8-bit-per-character document. A document author can use an optional document type declaration after the XML declaration to indicate what the root element of the XML document will be and possibly to point to a document type definition.

An **XML schema** is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.

There are languages developed specifically to express XML schemas. The Document Type Definition (DTD) language, which is native to the XML specification, is a schema language that is of relatively limited capability, but that also has other uses in XML aside from the expression of schemas. XSD (XML Schema Definition), a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. Like all XML schema languages, XSD can be used to express a set of rules to which an XML document must conform in order to be considered "valid" according to that schema. This topic contains the World Wide Web Consortium (W3C) purchase order examples. The first example is the schema for the purchase order. The second example is the instance document that is validated by this schema example.

The following example shows a schema, po.xsd, that defines a purchase order. This example shows the use of element**,** and attribute declarations. This example also shows simpleType and complexType definitions.

XPath, the **XML Path Language**, is a query language for selecting nodes from an XML document. In addition, XPath may be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath was defined by the World Wide Web Consortium (W3C). The XPath language is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting nodes by a variety of criteria. XPath has been adopted by a number of XML processing libraries and tools, many of which also offer CSS Selectors, another W3C standard, as a simpler alternative to XPath. The simplest XPath takes a form such as

- /A/B/C

That selects C elements that are children of B elements that are children of the A element that forms the outermost element of the XML document. The XPath syntax is designed to mimic URI (Uniform Resource Identifier) and Unix-style file path syntax.

More complex expressions can be constructed by specifying an axis other than the default 'child' axis, a node test other than a simple name, or predicates, which can be written in square brackets after any step. For example, the expression

- A//B/*[1]

XML document:

```
<?xml version="1.0" encoding="utf-8"?>
<wikimedia>
 <projects>
  <project name="Wikipedia" launch="2001-01-05">
   <editions>
    <edition language="English">en.wikipedia.org</edition>
    <edition language="German">de.wikipedia.org</edition>
    <edition language="French">fr.wikipedia.org</edition>
    <edition language="Polish">pl.wikipedia.org</edition>
   </editions>
  </project>
  <project name="Wiktionary" launch="2002-12-12">
   <editions>
    <edition language="English">en.wiktionary.org</edition>
    <edition language="French">fr.wiktionary.org</edition>
    <edition language="Vietnamese">vi.wiktionary.org</edition>
    <edition language="Trukish">tr.wiktionary.org</edition>
   </editions>
  </project>
 </projects>
</wikimedia>
```

XPath expression applied to an XML file

II. RELATED WORK

XML schema exploration and querying. Many approaches have been proposed by the academic and industrial communities. Few approaches have been proposed to evaluate XPath expressions on XML schemas. XML is a meta-markup language developed by the World Wide Web Consortium (W3C) to deal with a number of the shortcomings of HTML. As more and more functionality was added to HTML to account for the diverse needs of users of the Web, the language began to grow increasingly complex and unwieldy. The need for a way to create domain-specific markup languages that did not contain all the cruft of HTML became increasingly necessary and XML was born.

The main difference between HTML and XML is that whereas in HTML the semantics and syntax of tags is fixed, in XML the author of the document is free to create tags whose syntax and semantics are specific to the target application. Also the semantics of a tag is not tied down but is instead dependent on the context of the application that processes the document. The other significant differences between HTML and XML is that the XML document must be well-formed.

Although the original purpose of XML was as a way to mark up content, it became clear that XML also provided a way to describe structured data thus making it important as a data storage and interchange format. XML provides many advantages as a data format over others, including:

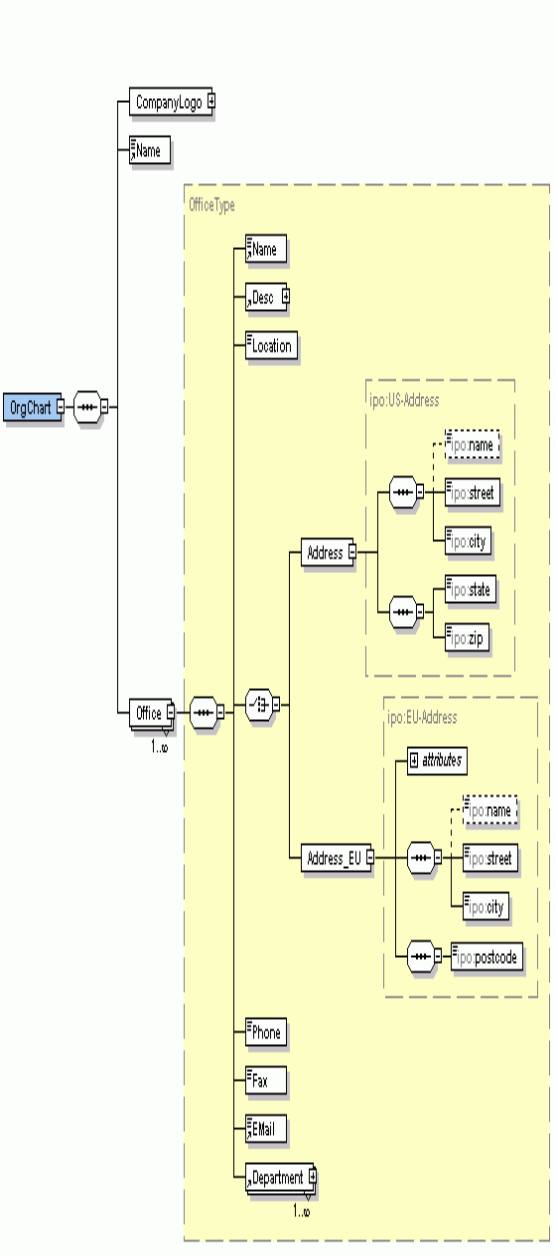
1. Built in support for internationalization due to the fact that it utilizes Unicode.
2. Platform independence (for instance, no need to worry about endianess).
3. Human readable format makes it easier for developers to locate and fix errors than with previous data storage formats.
4. Extensibility in a manner that allows developers to add extra information to a format without breaking applications that where based on older versions of the format.
5. Large number of off-the-shelf tools for processing XML documents already exist.

The world of traditional data storage and XML have never been closer together. To better understand how data storage and retrievel works in an XML world, this paper will first discuss the past, present, and future of structuring XML documents. Then we will delve into the languages that add the ability to query an XML document similar to a traditional data store. This will be followed by an exploration of how the most popular

RDBMSs have recognized the importance of this new data storage format and have integrated XML into their latest releases.

III. SCHEMA REPRESENTATION

According to the World Wide Web Consortium (W3C), which approved XML Schema as an official recommendation in 2001, "XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content, and semantics of XML documents."
XML Schema was born out of a need to provide a more powerful and flexible alternative to the standard DTD (Document Type Definition), a language for expressing SGML and XML content models. XML Schema offers a lengthy list of advantages for defining XML documents.
The pages on this site provide a large amount of resources, including links to W3C specifications, an industry standards schema library, XML Schema applications and uses, and descriptions of some powerful tools for working with XML Schema.
The diagram below is a graphical representation of an XML Schema.



IV. ABBREVIATED SYNTAX AND EXAMPLES OF XSPATH

XPath is a language for addressing parts of an XML document. Basic understanding of XPath is needed for XSLT and XQuery programming. In this piece, we shall show how to create XSLT style sheets that use some moderately complex XPath expressions.

Learning Objectives

- Understand how to use XPath expressions, in order use XSLT and XQuery more effectively
- Learn some XSLT programming constructions (conditions and loops)
- Being able to cope with most XML to HTML transformations

Prerequisites

- Editing XML tutorial (being able to use a simple DTD).
- XSLT Tutorial - Basics, introductory XSLT (xsl: template, xsl: apply-templates and xsl: value-of).

XPath Expression

The primary syntactic construct in XPath is the expression. An expression matches the production Expr. An expression is evaluated to yield an object, which has one of the following four basic types:

- node-set (an unordered collection of nodes without duplicates)
- Boolean (true or false)
- number (a floating-point number)
- string (a sequence of UCS characters)

Evaluating Expressions with Respect to a Context

Expression evaluation occurs with respect to a context. XSLT and XPointer specify how the context is determined for XPath expressions used in XSLT and XPointer respectively. The context consists of the following:

- Node, the context node
- Pair of nonzero positive integers, context position and context size. Context position is always less than or equal to the context size.
- Set of variable bindings. These consist of a mapping from variable names to variable values. The value of a variable is an object, which can be of any of the types possible for the value of an expression, can also be of additional types not specified here.
- Function library. This consists of a mapping from function names to functions. Each function takes zero or more arguments and returns a single result. See the XPath Recommendation for the core function library definition, that all XPath implementations must support. For a function in the core function library, arguments and result are of the four basic types:
  - Node Set functions
  - String Functions
  - Boolean functions
  - Number functions

Both XSLT and XPointer extend XPath by defining additional functions; some of these functions operate on the four basic types; others operate on additional data types defined by XSLT and XPointer.

- Set of namespace declarations in scope for the expression. These consist of a mapping from prefixes to namespace URIs.

XPath type system:

XPath is a strongly typed language in which the operands of various expressions, operators, and functions conform to expected types.

The type system for DB2 XPath includes a subset of the built-in types of XML schema and the predefined types of XPath.

The built-in types of XML Schema are in the namespace http://www.w3.org/2001/XMLSchema, which has the predeclared namespace prefix xs. Some examples of built-in schema types include xs: integer and xs: string.

- Overview of the type system
  the type system for DB2 XPath includes simple atomic types and complex types. A *simple atomic type* is a primitive or derived atomic type that does not contain elements or attributes. A *complex type* can contain mixed content or element-only content.
- Constructor functions for built-in data types
  every built-in atomic type that is defined in the XML Schema Definition language has an associated constructor function.
- Generic data types
  Generic data types support data that is not strongly typed.

- Data types for untyped data
  the xs: untyped and xs: untyped Atomic data types support untyped data.
- Xs: string
  The data type xs: string represents character strings in XML. Because xs: string is a simple type, it cannot contain any children.
- Numeric data types
  the xs: decimal, xs: double, and xs: integer data types support numeric data.
- Xs: boolean
  the data type xs: boolean supports the mathematical concept of binary-valued logic: true or false.
- Date and time data types
  the xs: date, xs: time, and xs: date Time data types support date and time data.
- Casts between XML schema data types
  you can use data type constructor functions to cast a value to a specific data type. Specify the value that you want to cast and the type to which you want to cast it.

REFERENCES

[1] S. Amer-Yahia, N. Koudas, and D. Srivastava, "Approximate Matching in XML - Tutorial," Proc. Int'l Conf. Data Eng., p. 803,2003.
[2] Z. Bellahsene, A. Bonifati, E. Rahm, eds., Schema Matching and Mapping. Springer, 2011.
[3] M. Benedikt and J. Cheney, "Semantics, Types and Effects for XML Updates," Proc. 12th Int'l Symp. Database Programming Languages, pp. 1-17, 2009.
[4] M. Benedikt and C. Koch, "XPath Leashed," ACM Computing Surveys, vol. 41, no. 1, article 3, 2008.
[5] M. Benedikt, W. Fan, and F. Geerts, "XPath Satisfiability in the Presence of DTDs," J. ACM, vol. 55, no. 2, article 8, 2008.
[6] A. Calı`, G. Gottlob, G. Orsi, and A. Pieris, "Querying UML Class Diagrams," Proc. Int'l Conf. Foundations Software Science and Computational Structures, pp. 1-25, 2012.
[7] F. Cavalieri, G. Guerrini, and M. Mesiti, "Navigational Path Expressions on XML Schemas," Proc. 19th Int'l Conf. Database and Expert Systems Applications, pp. 718-726, 2008.
[8] 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.
[9] 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.
[10] V.K. Chaudhri and P.D. Karp, "Querying Schema Information," Proc. CEUR Workshop Intelligent Access to Heterogeneous Information, vol. 8, pp. 4.1-4.6, 1997.