XML Schema-driven Generation of Architecture Components

Ali El bekai[1], Nick Rossiter[1]

[1]School of Informatics, Northumbria University,
Newcastle upon Tyne, UK

Abstract. It is possible to code by hand an XSL stylesheet that validates an XML document against some or all constraints of an XML schema. But the main goal of this paper introduces general techniques as a technology solution for different problems such as (a) generation of SQL schema from XMLSchema, (b) generating XSL stylesheet from XMLSchema, and (c) XQuery interpreter generating. Each of the techniques proposed in this paper works by XMLSchema-driven generation of architecture components with XSL stylesheet. As can be seen the input is XMLSchema and XSL stylesheet and the output is generic stylesheets. These stylesheets can be used as interpreter for generating other types of data such as SQL queries from XQueries, SQL data, SQL schema and HTML format. Using XSL stylesheets we present algorithms showing how we can generate these components automatically.

1 Introduction

XML is fast emerging as the dominant standard for *representing* and *exchanging* information over the Internet [4,2]. If data is stored and represented as XML documents, then it should be possible to query the contents of these documents in order to extract, synthesize and analyze their contents. Also, it is possible to transform theses data to another format and to generate a component from the XML data. Originally, XML was created to meet the challenges of *data exchange* in Web applications or between applications and users, not for data presentation purposes. To deal with presentation issues, XML needs to be used in conjunction with stylesheets to be easily viewed on the web. For this reason, eXtensible Stylesheet Language was created. XSL (Extensible Stylesheet Language) is being developed as part of the W3C stylesheets activity [13,14]. It has evolved from the CSS language to provide even richer stylistic control, and to ensure consistency of implementations. It also has document manipulation capabilities beyond styling. Of course, designing "traditional" software transformation tools for that purpose can achieve such a task. However, the power of having a cross-platform and XML independent language would be lost. Precisely, isolating content from formatting needs to be considered, especially when dealing with Web based documents. Therefore, any method of transforming XML documents into different formats such as XML, HTML, SQL, flat files or WML needs to be tailored so that it can be used with different platforms/languages.

This paper introduces the technological solutions for different problems such as (a) SQL schema generation, (b) XSL stylesheet generation, and (c) XQuery interpreter generation automatically by transforming an XMLSchema through an XSLT.2.

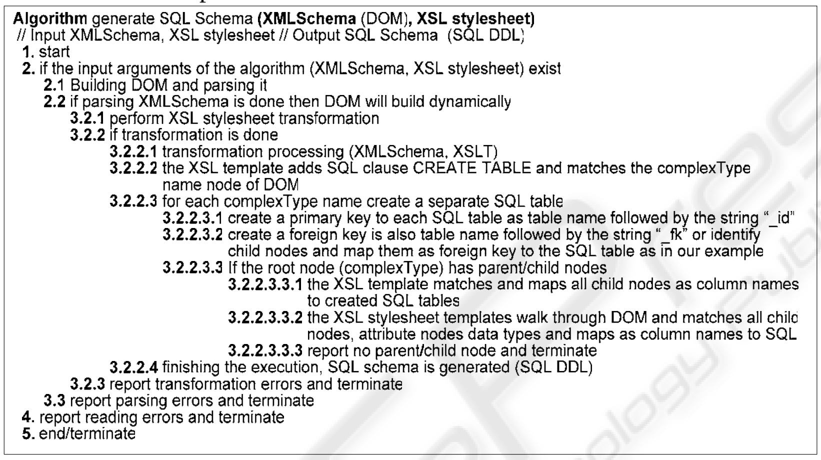
2 Related Work

Bourret [2] noted that XML and its surrounding technologies have many facilities in common with real databases such as storage (XML documents), schemas (DTDs, XML schema languages), query languages (XQuery, XPath, XQL, XML-QL, QUILT, etc.), and programming interfaces (SAX, DOM, JDOM). On the other hand, XML lacks efficient storage, indexes, security, transactions and data integrity, multi-user access, triggers, queries across multiple documents, and so on. Aboulnaga *et al* [1] started a discussion in the XML community about characterizing and generating XML data. Provost [10] considered the most common patterns for document content constraints, and finds that XML Schema validation is only the first of several necessary layers. Bourret [3] summarized two different mappings. The first part of the process, generally known as *XML data binding*, maps the W3C's XML Schemas to object schemas. The second known as *object-relational mapping*, maps object schemas to relational database schemas. In [6,11] techniques are presented for querying XML documents using a relational database system, which enables the same query processor to be used with most relational schema generation techniques. Norton [8] presents an XSL as validation to validate XML document. W3C and Peterson describe a query processor that works for different schema generation techniques [12, 9]. Their work is done in the context of data integration, and the tables generated by each relational schema generation technique are specified as materialized views over a virtual global schema. In [5] a translation is presented of XQuery expression drawn from a comprehensive subset of XQuery to a single equivalent SQL query using a novel dynamic interval encoding of a collection of XML documents. In [7] an algorithm is presented that translates simple path expression queries to SQL in the presences of recursion in the schema in the context of schema-based XML storage shredding of XML relations.

As a result none of the approaches described above introduce an algorithm to generate a generic XSL stylesheet for transforming XML to SQL statements or an XSL stylesheet for transforming XQueries to SQL queries or for generating SQL schema by using XSL. We will next introduce general algorithms to generate these components automatically.

3 SQL Schema Generations

Basically, the DOM [12,16] is a specification that comprises a set of interfaces that allow XML documents to be parsed and manipulated in memory. The main interfaces for an application with DOM are: Node: the base type, representing a node in the DOM tree; Document: representing the entire XML document as a tree of Nodes (the DOM parser will return Document as a result of parsing the XML); Element: to represent the elements of the XML document; Attribute, representing an attribute of some XML element; Interface enabling setting/getting the value of that attribute; and Text: representing the text content of an element (i.e. the text between tags that is not part of any child element). The DOM tree is composed of nodes, each of which represents a parsed document. Based on these interfaces, we will use the XMLSchema parse file (DOM) as input in our algorithm to generate the components automatically. Now we will introduce an algorithm that can be used automatically to generate the SQL schema as output of an arbitrary XMLSchema. In particular, we present a translation algorithm that takes as input an XMLSchema and XSL stylesheet and produces a SQL schema as the output.



```
Algorithm generate SQL Schema (XMLSchema (DOM), XSL stylesheet)
 // Input XMLSchema, XSL stylesheet // Output SQL Schema  (SQL DDL)
  1. start
  2. if the input arguments of the algorithm (XMLSchema, XSL stylesheet) exist
      2.1 Building DOM and parsing it
      2.2 if parsing XMLSchema is done then DOM will build dynamically
          3.2.1 perform XSL stylesheet transformation
          3.2.2 if transformation is done
               3.2.2.1 transformation processing (XMLSchema, XSLT)
               3.2.2.2 the XSL template adds SQL clause CREATE TABLE and matches the complexType
                       name node of DOM
               3.2.2.3 for each complexType name create a separate SQL table
                       3.2.2.3.1 create a primary key to each SQL table as table name followed by the string "_id"
                       3.2.2.3.2 create a foreign key is also table name followed by the string "_fk" or identify
                                 child nodes and map them as foreign key to the SQL table as in our example
                       3.2.2.3.3 If the root node (complexType) has parent/child nodes
                                 3.2.2.3.3.1 the XSL template matches and maps all child nodes as column names
                                             to created SQL tables
                                 3.2.2.3.3.2 the XSL stylesheet templates walk through DOM and matches all child
                                             nodes, attribute nodes data types and maps as column names to SQL
                                 3.2.2.3.3.3 report no parent/child node and terminate
               3.2.2.4 finishing the execution, SQL schema is generated (SQL DDL)
          3.2.3 report transformation errors and terminate
      3.3 report parsing errors and terminate
  4. report reading errors and terminate
  5. end/terminate
```

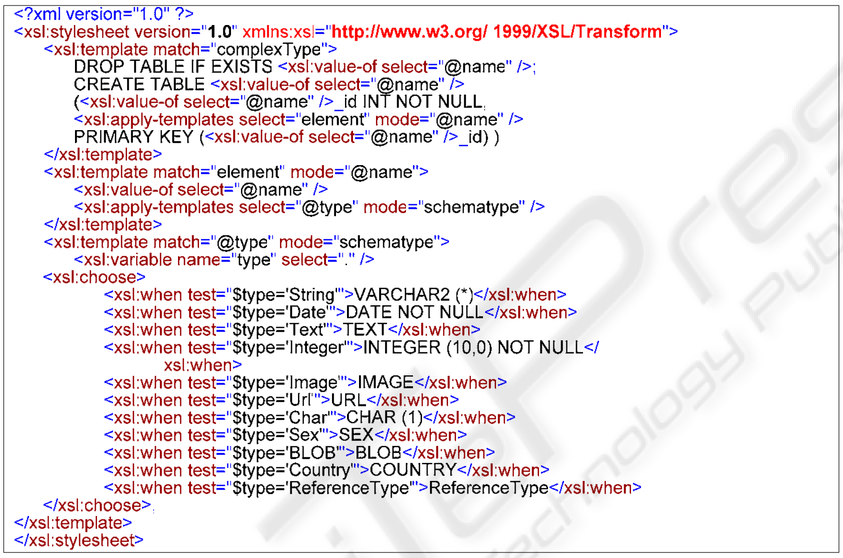
Fig.1. An algorithm for generating SQL schema from XMLSchema.

4 XSL Generations

Basically, it is possible to code by hand an XSL stylesheet that validates an XML document against some or all constraints of an XML schema. But in this section we introduce an algorithm as shown in Fig. 3 for generating an XSL stylesheet from an XMLSchema parse file (DOM) and as mentioned before the DOM tree is composed of nodes, each of which represents a parsed document. In other words this algorithm is the technology solution to the problem of generating an XSL automatically by transforming an XMLSchema through an XSLT. The result is a generic XSL stylesheet providing the mechanism to transform and manipulate XML data. Also the generated XSL stylesheet can be used to transform an XML document into another format such as XML to SQL statements and XML to HTML document.

5 The XQuery Interpreter Generation

Basically, the XQuery [15] is a language containing one or more query expressions. XQuery supports conditional expressions, element constructors, FOR, LET, WHERE, RETURN (FLWR) expressions, expressions involving operators and function calls and quantifiers, type checking and path expressions. Some XQuery expressions evaluate to simple node values such as elements and attributes or atomic values such as strings and numbers. The syntax for retrieving, filtering, and transforming records uses FOR, LET, WHERE and RETURN clauses. A FLWR expression creates some bindings, applies a predicate and produces a result set. XQuery does not conform to the same conventions as SQL. XQuery and SQL share some similar concepts. Both languages provide keywords for projection and transformation operations (SQL SELECT or XQuery RETURN). SQL supports joins between tables and XQuery supports joins between multiple documents.



```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/ 1999/XSL/Transform">
    <xsl:template match="complexType">
        DROP TABLE IF EXISTS <xsl:value-of select="@name" />;
        CREATE TABLE <xsl:value-of select="@name" />
        (<xsl:value-of select="@name" />_id INT NOT NULL,
        <xsl:apply-templates select="element" mode="@name" />
        PRIMARY KEY (<xsl:value-of select="@name" />_id) )
    </xsl:template>
    <xsl:template match="element" mode="@name">
        <xsl:value-of select="@name" />
        <xsl:apply-templates select="@type" mode="schematype" />
    </xsl:template>
    <xsl:template match="@type" mode="schematype">
        <xsl:variable name="type" select="." />
    <xsl:choose>
            <xsl:when test="$type='String'">VARCHAR2 (*)</xsl:when>
            <xsl:when test="$type='Date'">DATE NOT NULL</xsl:when>
            <xsl:when test="$type='Text'">TEXT</xsl:when>
            <xsl:when test="$type='Integer'">INTEGER (10,0) NOT NULL</
                xsl:when>
            <xsl:when test="$type='Image'">IMAGE</xsl:when>
            <xsl:when test="$type='Url'">URL</xsl:when>
            <xsl:when test="$type='Char'">CHAR (1)</xsl:when>
            <xsl:when test="$type='Sex'">SEX</xsl:when>
            <xsl:when test="$type='BLOB'">BLOB</xsl:when>
            <xsl:when test="$type='Country'">COUNTRY</xsl:when>
            <xsl:when test="$type='ReferenceType'">ReferenceType</xsl:when>
    </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```

Fig. 2. The generated XSL stylesheet for generating SQL schema from XMLSchema

Here are two simple examples: one with XPath type of a query and the other with FLWR expression. The single forward slash (/) signifies the parent-child relationship between elements. In tracing a path through a tree the expression starts at the root node and follows parent node and so on.

```
1) X / <collection>/<object>/<objectInfor>
2) For obj in <collection> do
      Where obj = <object>
      Return <object>
```

Finally in this section we introduce an algorithm for generating an XSL stylesheet from the XMLSchema to interpret the XQuery. In other words, this is the technology solution to the problem of generating automatically the XQuery interpreter by transforming an XMLSchema through an XSLT.

```
// Input XMLSchema, XSL stylesheet // Output XSL stylesheet
1. start
2.  if the input arguments (XMLSchema, XSL stylesheet) exist
    2.1 Build DOM and parse it
    2.2 if parsing XMLSchema is done DOM will build dynamically
        2.2.1 perform XSL stylesheet (each XSL stylesheet contains templates and commands to
              select and manipulate structure of data)
        2.2.2 if transformation is ok
              2.2.2.1 invoke the root node of DOM tree
              2.2.2.2 compare the root node with template rules in the stylesheet, if it matches the
                      first one then map to the root node of an XSL stylesheet output (as new template)
              2.2.2.3 If the root node has parent/child nodes
                      2.2.2.3.1 the XSL walks through DOM tree and pulls nodes from DOM tree and
                                places them with formatting as a new template to output
                      2.2.2.3.2 compare and match complexType nodes of the DOM tree with the and
                                XSL template, and for each a complexType name create a separate table
                      2.2.2.3.3 map the child nodes to the table as a column names, and also the
                                data type of XMLSchema mapped as values to the column names
                      2.2.2.3.4 iterate through the DOM tree nodes and set the keyword VALUES to the
                                output  as new template in the XSL stylesheet
                      2.2.2.3.5 insert the required statement and then return all template (new XSL
                                stylesheet generated)
              2.2.2.4 set null and terminate
        2.2.3 report transformation errors and terminate
    2.3 report parsing errors and terminate
 3. report reading errors and terminate
 4. terminate\end
```

Fig. 3. An algorithm for generating XSL from XMLSchema.

```
// Input XMLSchema, XSL stylesheet, // Output  XSL stylesheet  (Generic XSL stylesheet)
 1. start
 2. if the input arguments (XMLSchema, XSL stylesheet) exist
    2.1 Building DOM and parsing it
    2.2 if parsing XMLSchema is done, then DOM will build dynamically
        2.2.1 perform XSL stylesheet transformation
        2.2.2 if transformation is done
              2.2.2.1 XSL templates start from the root node of DOM tree
              2.2.2.2 If the root node has parent/child node
                      2.2.2.2.1 the XSL stylesheet template matches IN clause from DOM nodes and places it
                                as template name
                      2.2.2.2.2 the XSL template adds FROM clauses to the new template and mapped the
                                root node of DOM to the (new template in new XSL stylesheet)
                      2.2.2.2.3 the XSL stylesheet that pulls the nodes from DOM tree and places it with
                                formatting, into a new XSL stylesheet
                      2.2.2.2.4 the XSL stylesheet template walks through the DOM tree nodes and when the
                                template matches RETURN
                      2.2.2.2.5 the XSL template will add the SELECT clauses to the template
                      2.2.2.2.6 map the parent/child of DOM tree as new template to the new XSL stylesheet
                      2.2.2.2.7 Iterate and walk through the DOM tree nodes and when the XSL template matches
                                nodes has name WHERE or IF or Else clause
                      2.2.2.2.8 set the WHERE clauses as a new template into the new XSL stylesheet
                      2.2.2.2.9 When all the templates have been executed and placed in the output, then the new
                                template will be a generic XSL stylesheet generated
              2.2.2.3 report no parent/child node and terminate
        2.2.3 report transformation errors and terminate
    2.3 report parsing errors and terminate
 3.  report arguments error and terminate
 4.  terminate/end
```

Fig. 4. Algorithm generating XSL from XMLSchema to transform XQueries to SQL queries.

6 Conclusion

The contribution of this work is that it introduces general techniques for generating SQL schema, XSL and XQuery using XMLSchema and XSL stylesheet, which (a) enables the use of these techniques for transforming XML data to SQL data and storing it in a relational database, (b) allows the user to present HTML format, and (c) interprets XQueries (transforms XQueries to SQL queries) so that we can then retrieve and query data from the database. A potential cause for concern is that our general techniques may be less overlapping in implementation thus losing some efficiency.

However based on our prototype implementation in java, we have found that it is very quick to generate XSL stylesheets as an interpreter for different types of transformation such as SQL Schema and XQueries in to SQL queries. As we know it is possible to code by hand an XSL stylesheet that validates an XML document against some or all constraints of an XML schema and to generate an XSL stylesheet. However with our automated technique this task is easy, quick and less overlapping and we will use these generated components to integrate the (offline and online) components to satisfy our requirements. Also, we plan to extend these techniques to work with Multiple XMLSchema, not just single XMLSchema.

References

1. Aboulnaga, A., Jeffrey F. Naughton, Chun Zhang. Generating Synthetic Complex-Structured XML Data. Fourth International Workshop WebDB'2001 (2001).
2. Bourret, R, XML and Database www.rpbourret.com/xml/XMLAndDatabases.htm (2004).
3. Bourret, R, Mapping W3C Schemas to Object Schemas to Relational Schemas., www.rpbourret.com/xml/SchemaMap.htm March (2001).
4. Chawathe S. Describing and Manipulating XML Data, Bulletin IEEE Technical (1999).
5. DeHaan D, D. Toman, M. Consens, and M. T. Özsu, A Comprehensive XQuery to SQL Translation using Dynamic Interval Encoding, Proc ACM Int Conf Management Data (SIGMOD'03), San Diego, 623-634 June (2003).
6. Florescu D, D. Koaamann. Storing and Querying XML Data using an RDBMS, IEEE Data Engineering Bulletin, **22** 27-34 (1999)
7. Krishnamurthy R., Venkatesan T.Chakaravarthy, Raghav Kaushik, Jeffrey F. Naughton, Recursive XML Schemas, Recursive XML Queries, and Relational Storage: XML-to-SQL Query Translation, ICDE (2004)
8. Norton Francis. Generating XSL for Schema validation http://www.redrice.com/ci/generatingXslValid, May 20, (1999)
9. Peterson David, Paul V. Biron, and Ashok Malhotra XML Schema 1.1 Part 2: Datatypes. W3C, Working Draft WD-xmlschema11-2-20040716, July (2004)
10. Provost W, XML Validation Architecture using XML Schema, XPath, and XSLT. (2002)
11. Shanmugasundaram J., E. Shekita, J. Kiernan, R. Krishnamurthy, E. Viglas, J. Naughton and Igor Tatarinov. A General Technique for Querying XML Documents using a Relational Database System. ACM SIGMOD Record, 30(3), (2001)
12. W3C, Document Object Model (DOM) Level 2 HTML Specification Version 1.0 W3C Recommendation 09 January. http://www.w3.org/TR/2003/REC-DOM-Level (2003).
13. W3C. Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation. http://www.w3.org/TR/2001/REC-xsl-20011015/ October 15, (2001)
14. W3C XSL Working Group, W3C Recommendation on XSL Transformations (XSLT) http://www.w3.org/TR/xslt. (1999)
15. W3C.XQuery 1.0: An XML Query Language, W3C Working Draft July 23, (2004).
16. W3C.DOM Working Group, Document Object Model, http://www.w3.org/DOM/. (2004).