XML Schema-driven Generation of Architecture Components

Ali El bekai¹, Nick Rossiter¹

¹School 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 these 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, SQL, 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 presence 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 rep-
resent 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 XML Schema 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 XML Schema. In particular, we present a translation algorithm that takes as input an XML Schema and XSL stylesheet and produces a SQL schema as the output.

Algorithm generate SQLSchema (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
2.2.1 perform XSL stylesheet transformation
2.2.2 // Transformation is done
3.2.2.2 transformation processing (XMLSchema, XSL)
3.2.2.2.1 the XSL template adds SQL clause CREATE TABLE and matches the complexType name node of DOM
3.2.2.2.2 For each complexType name create a separate SQL table
3.2.2.2.3.1 create a primary key to each SQL table as table name followed by the string "_id"
3.2.2.2.3.2 create a foreign key is also table name followed by the string "_f" or identify child nodes and map them as foreign key to the SQL table as in our example
3.2.2.2.3.3 if the root node (complexType) has parent/child nodes
3.2.2.2.3.3.1 the XSL template matches and maps all child nodes as column names to created SQL table
3.2.2.2.3.2 the XSL stylesheet templates walk through DOM and matches all child nodes, attribute nodes data types, and maps as column names to SQL
3.2.2.2.3.3 report no parent/child node and terminate
3.2.2.4 finishing the execution. SQL schema is generated (SQL DDL)
3.3 report parsing errors and terminate
5. end terminate

Fig. 1. An algorithm for generating SQL schema from XML Schema.

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 XML Schema 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 XML Schema 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.

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

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 XML Schema, not just single XML Schema.

References