# XML PUBLISHING

Zachary Ives
Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104-6389
zives@cis.upenn.edu

## SYNONYMS

XML Export

## DEFINITION

*XML Publishing* typically refers to the creation of XML output (either in the form of a character stream or file) from a relational DBMS. XML Publishing typically must handle three issues: converting an XML query or view definition into a corresponding SQL query; encoding hierarchy in the SQL data; and generating tags around the encoded hierarchical data. Since in some cases the relational data may have originated from XML, the topics of *XML Storage* and XML Publishing are closely related and often addressed simultaneously.

## HISTORICAL BACKGROUND

The topic of XML Publishing arose very soon after database researchers suggested a connection between XML and semi-structured data [6], a topic that had previously been studied in the database literature [2, 1, 9]. Initially the assumption was that XML databases would probably need to resemble those for semi-structured data in order to get good performance. Florescu and Kossmann [11] showed that storing XML in a relation database could be more efficient than storing it in a specialized semistructured DBMS. Concurrently, Deutsch et al. were exploring hybrid relational/semistructured storage in STORED [7]. Soon after, the commercial DBMS vendors became interested in adding XML capabilities to their products. The most influential developments in that area came from IBM Almaden's XPERANTO research project [3, 13], which formed the core of Shanmugasundaram's thesis work [12]. We observe today that most commercial DBMSs use a combination of all of the aforementioned techniques: storing XML data in relations, storing hybrid relational/semistructured data, and storing XML in a hierarchical format resembling semistructured data.

## SCIENTIFIC FUNDAMENTALS

As every student of a database class knows, a good relational database schema is in first normal form (1NF): separate concepts and multi-valued attributes are split into separate tables, and taken together the tables can be visualized as a graph-structured Entity-Relationship Diagram. In contrast, XML data is fundamentall *not* in 1NF: an XML document has a single root node and *hierarchically* encodes data.

Thus there are two main challenges in XML Publishing: first, taking queries or templates describing XML output and mapping them into operations over 1NF tables; and second, efficiently computing and adding XML tags to the data being queried. Typically, the latter problem is addressed with two separate modules, and hence the XML Publishing problem consists of three steps:

**1. Converting queries**, which typically are posed in a language other than SQL, into an internal representation from which SQL can be constructed.

**2. Composition and optimization** of the resulting queries, such that redundant work is minimized.

**3. Adding tags**, which is often done in a postprocessing step outside the DBMS.

We address each of these topics in the remainder of this section. We primarily discuss the XPERANTO and SilkRoute systems, which established many of the basic algorithms used in XML Publishing.

## Converting Queries

The first issue in XML Publishing is that of taking the specification of the XML output, and converting it into some form from which SQL can be constructed. A number of different forms have been proposed for specifying the XML output:

**Proprietary XML template languages**, such as IBM DB2's Document Access Definition (DAD) or SQL Server's XML Data Reduced (XDR), which essentially provide a scheme for annotating XML templates with SQL queries that produce content.

**SQL extensions**, such as the SQL/XML standard [14], which extend SQL with new functions to add tags around relational attributes and intermediate table results.

**Relational-XML query languages**, such as the RXL language in early versions of SilkRoute [8], which provide a language that creates hierarchical XML content using relational tables. RXL resembles a combination of Datalog and an XML query language (specifically, XML-QL [5]), and it can be used to define XML views that can be queried using a standard XML query language.

**XML query languages with built-in XML views of relations**, an approach first proposed in XPERANTO [4], where each relational table is mapped into a virtual XML document and a standard XML language (like XPath or XQuery) can be used to query the relations and even to define XML views.

Over time, the research community has come to consensus that the last approach offers the best set of trade-offs, as it offers a single compositional language for all XML queries over the data in the RDBMS. The commercial market has settled on a combination of this same approach, for XML-centric users, plus SQL extensions for relation-centric users. Here we shall assume an XML-centric perspective, with XQuery as our language, since it is representative.

XML Publishing systems generally only attempt to tackle a subset of the full XQuery specification, focusing on the portions that are particularly amenable to execution in a relational DBMS. There are many commonalities between basic XQueries and SQL: selections, projections, and joins can be similarly expressed in both of these languages, views can be defined, and queries can be posed over data in a way that is agnostic as to whether the data comes from a view or a raw source. The two main challenges in conversion lie in the input to the XQuery — where XPaths must be matched against relations or XML views defined over relations — and in creating the hierarchical nested XQuery output.

XPath matching over built-in views of relations is, of course, trivial, as each relation attribute maps to an XML element or attribute in the XPath. Conversion gets significantly more complex when the XPaths are over XML views constructed over the original base relations: now we are faced with the problem of XML query composition, where the DBMS should not have to compute each composed view separately, but rather it should statically *unfold* them.

## Composition and Optimization

As a means of composing queries, SilkRoute takes each XQuery definition and creates an internal *view forest* representation that describes the structure of the XML document; it converts an XQuery into an canonical representation called XQueryCore, and its algorithms compose XQueryCore operations over an input document. XPERANTO, based in large part on IBM's DB2, performs very similar operations, but starts by creating an internal representation of each query block called in a model XQGM, then executes a series of rewrite rules to merge the blocks by merging steps that create hierarchy in a view's output with steps that traverse that hierarchy in a subsequent query's input.

The resulting SQL is often highly complex and repetitive, with many SQL blocks being unioned together: each level of the XML hierarchy may require a separate SQL block, and each view composition step in the optimization process may create multiple SQL query blocks (an XPath over a view may match over multiple relational attributes from the base data). Thus, optimizing the resulting SQL becomes of high importance. XML publishing systems typically convert from XQuery to SQL, not to actual executable query plans, so they can only do a limited amount of optimization on their own. Here they use a significant number of heuristics [8, 13] to choose the best form for the sets of SQL queries, and rely on the RDBMS's cost-based optimizer to more efficiently optimize the queries.

a

| *id* | *val* |
|---|---|
| 1 | 123 |
| 2 | 789 |

b

| *id* | *pid* | *val* |
|---|---|---|
| 2 | 1 | 234 |
| 3 | 1 | 456 |
| 4 | 2 | 832 |

(a) Relations for elements a and b

| *id_a* | *val_a* | *id_b* | *val_b* |
|---|---|---|---|
| 1 | 123 | 2 | 234 |
| 1 | 123 | 3 | 456 |
| 2 | 789 | 4 | 832 |

(b) SilkRoute

| *id_a* | *val_a* | *id_b* | *val_b* |
|---|---|---|---|
| 1 | 123 | 2 | 234 |
| 1 | – | 3 | 456 |
| 2 | 789 | 4 | 832 |

(c) XPERANTO

Figure 1: Tuple representations of XML data

(Recently, commercial vendors have invested significant effort in improving their cost-based query optimizers for executing the types of queries output by XML Publishing systems.)

## Adding Tags

The final step in XML Publishing is that of adding tags to the data from the relational query. There have been two main approaches to this task: *in-engine*, relying on SQL functions to add the tags; and *outside the DBMS*, relying on an external middleware module to add the tags. We briefly describe each.

**In-engine tagging.** In [13], the authors proposed a method for adding XML tags using the SQL functions XMLELEMENT, XMLATTRIBUTE, and XMLAGG: each of these takes a relational attribute or tuple and converts it into a partial XML result, encoded in a CLOB datatype. Naturally, the first two create an element or attribute tag, respectively, around a data value. The third function, XMLAGG, functions as an SQL aggregate function: it takes an SQL row set (possibly including XMLELEMENT or XMLATTRIBUTE values) and outputs a tuple with a CLOB containing the XML serialization of the rowset's content. The drawback to the in-engine tagging approach is that CLOBs are not always handled efficiently, as they are typically stored separately from the tuples with which they are associated. Hence, when this method was incorporated into the SQL/XML standard, a new XML datatype was proposed that could be implemented in a manner more efficient than the CLOB.

**Tagging middleware.** A more common approach is to simply a tuple representation of the XML document tree/forest within the SQL engine, and to convert this tuple stream into XML content externally, using a separate tagging module [13, 10]. There are two common techniques for encoding hierarchy in tuple streams: *outer join* and *outer union*. Suppose we have relational tables in Figure 1(a), encoding an element $a$ and its child element $b$, and we wished to encode their output as the following XML:

```
<a id="1" val="123"><b id="2" val="234"/><b id="3" val="456"/></a>
<a id="2" val="789"><b id="4" val="832"/></a>
```

*Outer join.* In SilkRoute, tuples are generated using outerjoins between parent and child relations. The tuple stream will be sorted in a way that allows the tagger to hold the last tuple, not all data (see Figure 1(b)). The "parent" portion of the tree (the $a$ element) will appear in both tuples, and the external XML tagger will compare consecutive tuples in the tuple stream to determine the leftmost position where a value changed — from which it can determine what portion of the XML hierarchy has changed.

*Outer union.* In contrast, XPERANTO uses a so-called *sorted outer union* representation (Figure 1(c)), which substitutes null values for repeated copies of values (but not keys). Its approach relies on two characteristics of IBM's DB2 engine that are shared by some but not all other systems: (1) null values can be very efficiently encoded and processed, meaning that the query is more efficient; (2) null values *sort high*, i.e., DB2's considers null values to have a sort value greater than any non-null value. The tagger simply needs to look for the first non-null attribute to determine what portion of a the XML to emit.

## KEY APPLICATIONS

XML publishing has become a key capability in the relational database arena, as increasingly data interchange mechanisms and Web services are being built using XML data from an RDBMS. Today the "Big Three" commercial DBMS vendors all use some techniques for XML Publishing, and it is likely that smaller DBMSs, including those in open source, will gradually incorporate them as well.

## CROSS REFERENCE

XML Storage, XML Querying.

## REFERENCES

Jayavel Shanmugasundaram, Jerry Kiernan, Eugene J. Shekita, Catalina Fan, and John Funderburk. *Querying XML Views of Relational Data*, VLDB 2001, pp. 261–270.

Jayavel Shanmugasundaram, Eugene Shekita, Rimon Barr, Michael Carey, Bruce Lindsay, Hamid Pirahesh, Berthold Reinwald. *Efficiently Publishing Relational Data as XML Documents*, VLDB Journal 10(2-3), 2001.

Mary F. Fernandez, Yana Kadiyska, Dan Suciu, Atsuyuki Morishima, and Wang Chiew Tan. *SilkRoute: A framework for publishing relational data in XML*, ACM TODS 27(4), 2002, pp. 438-493.

## RECOMMENDED READING

[1] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Winer. The Lorel query language for semistructured data. In *Proceedings of International Journal on Digital Libraries*, volume 1(1), April 1997.

[2] Peter Buneman, Susan B. Davidson, Mary F. Fernandez, and Dan Suciu. Adding structure to unstructured data. In *ICDT*, volume 1186, 1997.

[3] Michael Carey, Jerry Kiernan, Jayavel Shanmugasundaram, Eugene Shekita, and Subbu Subramanian. XPERANTO: A middleware for publishing object-relational data as xml documents. In *VLDB*, 2000.

[4] Michael J. Carey, Daniela Florescu, Zachary G. Ives, Ying Lu, Jayavel Shanmugasundaram, Eugene Shekita, and Subbu Subramanian. XPERANTO: Publishing object-relational data as XML. In *WebDB '00*, 2000.

[5] Alin Deutsch, Mary F. Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for XML. In *8th World Wide Web Conference*, 1999.

[6] Alin Deutsch, Mary F. Fernández, Daniela Florescu, Alon Y. Levy, and Dan Suciu. XML-QL. In *QL*, 1998.

[7] Alin Deutsch, Mary F. Fernandez, and Dan Suciu. Storing semistructured data with STORED. In *SIGMOD*, 1999.

[8] Mary Fernandez, Weng-Chiew Tan, and Dan Suciu. SilkRoute: Trading between relations and XML. In *9th World Wide Web Conference*, November 1999.

[9] Mary F. Fernandez, Daniela Florescu, Jaewoo Kang, Alon Y. Levy, and Dan Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *SIGMOD*, 1998.

[10] Mary F. Fernandez, Yana Kadiyska, Dan Suciu, Atsuyuki Morishima, and Wang Chiew Tan. Silkroute: A framework for publishing relational data in xml. *ACM Trans. Database Syst.*, 27(4), 2002.

[11] Daniela Florescu and Donald Kossmann. A performance evaluation of alternative mapping schemes for storing XML data in a relational database. Technical Report 3684, INRIA, March 1999.

[12] Jayavel Shanmugasundaram. *Bridging Relational Technology and XML*. PhD thesis, University of Wisconsin-Madison, 2001.

[13] Jayavel Shanmugasundaram, Eugene J. Shekita, Rimon Barr, Michael J. Carey, Bruce G. Lindsay, Hamid Pirahesh, and Berthold Reinwald. Efficiently publishing relational data as xml documents. *VLDB J.*, 10(2-3), 2001.

[14] ISO/IEC 9075-14:2003 information technology – database languages – SQL – part 14: XML-Related Specifications (SQL/XML).