Speak the language

Teradata XML Services enables easy relational database publishing. by Srinivas Pandrangi

The Teradata Database is home to some of the most valuable data in your enterprise. Since data must move among applications for it to be useful, the applications must be able to talk to one another; therefore, the language format must be universal. Extensible markup language (XML) is one of the most commonly used formats. XML is integral to the standard technical stacks in reference data services, the method used to move data through applications across the enterprise, with all messages and data exchanged in XML format.

For the Teradata Database to smoothly participate in an XML-based multi-application environment, Teradata XML Services was developed to provide a convenient, high-performance method that makes it easy to publish data from relational database tables in XML format and to populate the database from values in XML documents.

Teradata XML Services is composed of a set of stored procedures and functions. (See figure.) The focus of the XML Services is on XML publishing (converting data in database tables into XML documents) and shredding (updating database tables from data in XML documents).

Also included are functions that provide common XML processing capabilities, such as XML schema validation, extensible stylesheet language transformations (XSLT), XML parsing (checking if documents are well-formed) and querying using XML path language (XPath). Users who choose to write custom XML processing functions can utilize the Xerces and Xalan libraries available on MP-RAS, Linux and Windows.

While data is formatted into an XML representation most often for application integration, for analytics, the same data is best represented in a relational model. Therefore, it sometimes becomes necessary to translate between the hierarchical model of XML documents and the tabular model of relational data stores. Teradata XML Services eases this translation by providing frameworks for conversions between XML and relational models in either direction.

Data services used in service-oriented architecture (SOA) implementations often have an XML interface. XML Services provides the ability to quickly set up data services based on data in the active data warehouse. The data services expose these valuable assets to enterprise applications while imposing strict data use and governance constraints.

XML publishing
To publish simple XML structures, XML Services provides XML construction functions, which mimic the behavior of SQL/XML pseudo-functions such as...
as XMLELEMENT, XMLATTRIBUTE, XMLFOREST and XMLAGG. These functions can be used in SQL-select statements with appropriate nesting to build simple hierarchies. The following snippet illustrates how to create a simple hierarchy by nesting function invocations:

```sql
select sysxml.xmlelement('customer', null, null,
    sysxml.xmlelement('order', null, sysxml.
        xmlattributes('orderid', orderid),
    sysxml.xmlelement('lineitems', null, null,
        sysxml.xmlelement('lineitem', null, null,
            itemid)
    ...
    ))) from ...
```

However, this process might not be suitable for complex XML structures. The XML Services publishing framework supports greater complexity in structure but requires a more deliberate design process. When utilizing the publishing framework, the user first constructs a mapping definition that describes how the database data is mapped to a hierarchical XML structure.

Visual design tools such as the Teradata Plug-in for Eclipse IDE, as well as third-party tools like Stylus Studio, can be used to define this mapping. (See figure 2 for an example of the mapping between XML and relational models.)

The user then completes the design process by registering the mapping with the Teradata Database. At runtime the user can invoke a publishing-stored procedure by supplying it with a user query and the registered mapping and, in return, will receive an XML representation of the query results.

Documents can be published one at a time or in a group, at the same time or streamed. When using the streaming mode, XML Services generates a query that, when instantiated in a view or macro, serves as a transformation object that returns the XML fragments in a row set. The client can cursor through this set and concatenate it to obtain the XML document on the client side. Users will choose a variant based on application requirements and performance considerations.

**XML shredding**

Moving data from its XML representation into database tables—the opposite direction from publishing—is called shredding. Like publishing, users can choose from a number of varying complexity and performance options for shredding, such as XPath to define simple mappings or XSLT to handle greater complexity.

XPath is a query language in which a path notation (such as Customer/Order/LineItem) identifies nodes of interest in an XML document tree. XPath-based shredding uses path expressions to identify column values in an XML document. This method is ideal when mapping definitions do not involve complex transformations or logic.

XSLT-based shredding allows for greater complexity in conditional logic, format conversions and type mappings. Similar to the publishing framework, users follow a design process in which they define a mapping from an XML schema to a relational schema. This can be accomplished through the use of a visual tool, or it can be coded manually. The mapping is then registered with the database. At runtime, the user performs shredding by

---

**Figure 2** Visual mapping definition for XML

Mapping between XML and relational models can be done visually using a tool such as the mapper available in the Teradata Plug-in for Eclipse IDE.
invoking a shredding-stored procedure with one or more source XML documents and a registered mapping definition.

Options for XSLT-based shredding include shredding a single document at a time, shredding in batch mode (parallel set-based update operations provide better performance) and producing intermediate output formats like comma-separated values (CSVs).

Shredding is applicable when XML-formatted data must be stored and indexed for future analysis. Most enterprise applications use XML as a message format, and participants in these message flows often need to persist the XML message either verbatim as strings or by extracting the data components and storing them in columns of appropriate data types. The Teradata Database facilitates both, but the latter is preferred when the data has uses such as analytics and reporting.

Figure 3 illustrates how XML documents that conform to the insurance industry’s Association for Cooperative Operations Research and Development (ACORD) standard are shredded into database tables using the shredding framework available through XML Services. The example also displays the shredding vocabulary defined by XML Services. Applications send update instructions to the Teradata Database in XML format by marking them with appropriate tags defined by this schema (transaction, insert, update, upsert, etc.).

Figure 3  XML shredding

Mapping definition in the form of an XSLT stylesheet describes how contents of an XML document will be used to update target database tables.

**Necessary tools**

The publishing and shredding capabilities, the XML processing functions, and the extensibility afforded by the XML parsing and transformation libraries, user-defined functions and external-stored procedures give users the tools they need to quickly build high-performance XML applications on their Teradata Database.

As XML becomes the language most applications speak, and concepts such as SOA (which uses XML extensively) become popular, leveraging the database’s XML functionality has become a priority. T

Srinivas Pandrangi is the architect of Teradata XML Services. He has more than 10 years’ experience implementing industry standards related to XML.