

Part 5. Leveraging SQL for XML data with XMLTABLE function

Hello DB2 friends, my name is Guogen Zhang. I'm a Senior Technical Staff Member at IBM Silicon Valley Lab, responsible for the delivery of pureXML in DB2 for z/OS server.

This is part 5 of the pureXML podcast. The title is “Leveraging SQL for XML data with XMLTABLE function.” I'd like to repeat the announcement: two SQL built-in functions XMLTABLE and XMLCAST were delivered post V9 GA, in a three-APAR delivery. They are PK51571, PK51572, and PK51573.

We are getting to some more advanced topics today. In this part, I'm going to talk about how to do processing, reporting and some analytics on the XML data using SQL and the XMLTABLE function.

In a previous part of the podcast, we've mentioned that you can extract pieces of a document using XMLQUERY function. The result from the XMLQUERY function will stick together if there are multiple items, and you cannot process further each of them separately, unless you use the XMLTABLE function.

Before getting into usage patterns, let's learn what XMLTABLE does: The XMLTABLE function is an SQL built-in function. The syntax for XMLTABLE is an extension to XMLEXISTS and XMLQUERY. You use XPath for a row expression (we call row path expression), and use the PASSING keyword to pass an XML value, typically an XML column, and also pass other parameters into XPath (this part is the same as XMLEXISTS and XMLQUERY), then followed by the COLUMNS keyword and column specifications. Each column spec contains the column name, SQL type, the PATH keyword, and the XPath. The SQL type could be INT, VARCHAR, or XML, etc. except for binary type such as VARBINARY, BLOB. And the XPath is for the column (we call column path expression), relative to the row path expression to get the column value. XMLTABLE also does implicit cast from XML value to SQL type if you specify non-XML type.

If you need namespaces in the XPath, you need to declare them in XMLNAMESPACES function as the first optional argument of the XMLTABLE function, to apply to all the arguments. The namespaces declared inside the row path expression does not apply to the column path expressions, that's how the SQL standard defines (a mistake in my opinion, because it hurts the usability).

You use the XMLTABLE function in the SQL FROM clause always. And the row expression identifies the nodes in a document for the rows; typically they are repeating elements, such as items in a purchase order. Then you pick the components from the row nodes as the columns. If you need to get to the values from above the repeating nodes, such as order number from the header of a purchase order, you use the parent axis (..) in the column path expressions.

Typically the XMLTABLE function will return multiple rows, but in special cases, you can use it to return one row with multiple components from the document.

As I mentioned, the XMLQUERY function can only extract a sequence; the special case one piece. You can use XMLCAST to cast a singleton value returned from XMLQUERY into SQL value. If you want to get multiple pieces from an XML document, you could use multiple XMLQUERY functions, but XMLTABLE function is much better as it will not traverse the document multiple times like multiple XMLQUERY does. XMLTABLE function is like combine many XMLQUERY and optional XMLCAST into one single function.

So XMLTABLE function can iterate through items of a sequence and let you perform further processing on each item. You can use XMLQUERY or XML constructors and then XMLAGG to combine them back into a single document.

You can also use SQL date and time types for XML data to make up the missing capability in DB2 9 for z/OS XPath support, because we don't have direct date and time support in XPath yet.

Now, let's talk about three main usage patterns for XMLTABLE:

The first, using XMLTABLE function to achieve XQuery capability.

We got lots of questions: what's the difference of XQuery and SQL/XML with XPath. XQuery has the FLWOR expression to iterate through a sequence and construct new documents. This is the major difference from XPath - XPath can only return a sequence. A FLWOR expression can be achieved using XMLTABLE, XMLEXISTS, XML constructors, and XMLAGG functions. You use XMLTABLE function to iterate through items in a sequence, use XMLEXISTS as conditions to filter, use XMLQUERY to extract pieces further if you need to, and then use XML constructor functions to construct new document fragments, and then use XMLAGG to group them together. There are one-to-one correspondence between FLWOR and these SQL/XML functions. I have converted almost all the W3C XQuery use cases into SQL/XML queries. If you like to have the converted queries, please contact me (available in the wiki now).

Now the second usage pattern is you can use XMLTABLE function to decompose a large document into smaller documents for storage. You can generate some regular columns, and XML columns from the XMLTABLE function. You would need XMLDOCUMENT function, the constructor, to construct a document to insert into the XML column, as DB2 9 restricts the stored XML data to well-formed documents only.

The third usage pattern, you can decompose XML data into relational table view, then you can apply SQL functions, especially aggregate functions and rank functions with GROUP BY to the table view and generate reports. This is very useful in bridging the gap between the native XML and existing software that uses relational interface.

This is also very powerful as it is beyond XQuery 1.0 capabilities as XQuery 1.0 does not have a flexible GROUP BY construct. It is also more powerful than XSLT, as the aggregation is across multiple rows/multiple documents in a table, while XSLT typically only works on a single document at a time.

Now performance considerations: we recommend to use a hybrid table design with relational and XML columns, and when defining XMLTABLE views, always use relational columns and relational predicates if possible, and combine with the XMLTABLE result. Prefer relational indexes if possible. If there are XML numeric indexes, the XMLTABLE function will use XML indexes, with pushdown predicates from SQL predicates.

Use cascading XMLTABLE function for nested repeating groups - first XMLTABLE gets to the high-level repeating group, then for each group, the second XMLTABLE gets to the next repeating level. That will use fewer parent axes, and performs better.

To recap, XMLTABLE function significantly increases the XML capability. Together with XMLEXISTS, XMLQUERY, XML constructors, and XMLAGG, it can achieve some XQuery capabilities and even beyond, for transaction processing, reporting and some analytics.

Please contact me or send emails to db2zxml@us.ibm.com for any questions. Until next time, you have a wonderful day. Bye-bye.