Part 4. SQL/XML Constructor Functions (XML Publishing Functions)

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.

This is part 4 of the pureXML podcast. The title is “SQL/XML Constructor functions.” SQL XML constructor functions, also known as XML publishing functions, are used to generate XML data from relational data or construct new documents from relational data and XML data. They can be used to construct XHTML documents for web pages, generate SOAP messages for web services, or produce industry standard XML data for exchange, or just XML data representing relational data for distribution. As long as you have the data in DB2 and want to output in XML, you can use them.

Let me briefly introduce the XML functions available in DB2 V8 first. In DB2 for z/OS V8, we have a set of XML publishing functions, including constructor functions supported in the new function mode. The functions are:

- **XMLELEMENT**: to construct one element.
- **XMLAttributes**: to construct one or more attributes for an element.
- **XMLNAMESPACES**: to declare in-scope namespaces for an element or elements.
- **XMLFOREST**: to construct a sequence of elements -- trees of nodes from expressions.
- **An additional function to concatenate multiple pieces of XML together**: that’s XMLCONCAT.
- **An XML aggregate function -- XMLAGG**: to concatenate multiple XML pieces into one value from a group of rows, usually used with GROUP BY. It is very useful in nesting rows or denormalizing relational rows into nested structure. If you denormalize relational data into nested structure where multiple sub-elements are contained in an element, you must use XMLAGG with GROUP BY.
- **The last function is XML2CLOB**: it’s a cast function that converts transient XML data type into a CLOB. There is no real XML type available in DB2 V8 and bind-out of XML data has to go through CLOB.

These functions are available in V8 New Function Mode. As SQL built-in functions, they are available without any additional setup. These functions provide enough capability to allow you to generate almost all practical XML documents from relational data. The benefit is simplified application client, and much higher performance. If you compare DOM-based construction using Java with SQL, SQL XML construction is much simpler and much more efficient, because you get XML document in one shot, instead of piece by piece in relational data format, and constructing them into XML format in the client. Let me tell you my own experience in using these functions: It's very natural - almost one to one correspondence from the functions to the document constructs – the same sequence and nesting structure. It’s really straight-forward. The main challenge in using these functions in SQL has been to match the pairs of parentheses - due to nesting nature.

In DB2 9, we extended constructor functions in the following way:
- First, we added binary data support and null handling options for the V8 functions.
Second, we added XMLTEXT, XMLPI, XMLCOMMENT, and XMLDOCUMENT constructors to make a complete set of XML constructors.

Third, they take and generate real XML data type.

Last, a new XMLSERIALIZE function to convert XML data into a LOB (was introduced). This is optional as you can select XML back directly and implicit XMLSERIALIZE will occur.

These are all SQL/XML standard functions.

V8 functions still work as is, no rebind is necessary in V9. One thing is different in the result if you rebind an existing package in V9: it will generate empty element tag for empty elements, instead of V8's way -- we have start tag and end tag pair for an empty element. With V9 features, you can achieve XQuery capability using SQL/XML with XPath.

IBM Data Studio provides assistance in generating SQL queries that construct XML documents from relational data if you have XML schema for the result documents.

Before I explain the functions, let me explain the mappings from relational to XML: there are name mapping and value mapping. The name mapping is to map an SQL identifier (ID) to XML name, while value mapping is to map an SQL value to an XML value according to the data type correspondence. XML values have different representations from SQL, such as date and time. For the names, the general rule is if an expression is a column, the name will be mapped to an XML name, escaping will apply if the SQL name is not a legal XML name. If an expression is not a column, then it does not have a name, a name may be required in the AS name clause when a name is needed, such as in XMLATTRIBUTES and XMLFOREST.

Now, let me explain each function. XMLELEMENT is to construct an XML element, using the arguments: the first argument is the name. You supply an ID for the element name. However, for a regular SQL function, an ID means a column reference. That's why it has a NAME keyword before the ID to signify it's a name for an element. The second argument can be XMLNAMESPACES, which is optional. If it appears, it declares in-scope namespaces for the element. The next argument is the optional XMLATTRIBUTES. All the rest arguments are the content of the element; they can be any SQL type except for binary data in V8. The types will be mapped to XML values during the construction. XMLELEMENT itself can be nested to the level allowed by function nesting. You can have as many content arguments as you need in one XMLELEMENT function. If all the content expressions are null, an empty element is generated.

Next, XMLATTRIBUTES is to construct attributes for an element. It takes expressions with optional AS attribute-name for each attribute. XMLATTRIBUTES can only be used inside XMLELEMENT, as either the second or the third argument, depending on whether you have XMLNAMESPACES. If the value of an argument is NULL, then the corresponding attribute will not be there in the element.
XMLNAMESPACES declares in-scope namespaces for the element. It has namespace URI AS namespace prefix pairs as the arguments. The scope of the namespaces is from the beginning of the XMLELEMENT function to the end of the XMLELEMENT function, including scalar fullselect inside the function if any.

XMLFOREST is a convenience function; it can have a variable number of arguments. It generates one element for each argument, except for the optional XMLNAMESPACES in the first argument. The name of each element is either the name of the expression when it's a column, or the name in the AS name clause. If an argument is null, the corresponding element will not appear in the result. It is very convenient to generate one element for each column in a row.

XMLCONCAT is to put multiple pieces together. The arguments are from the same row of the intermediate result table.

XMLAGG is very powerful, it concatenates multiple XML pieces from each row of a group together. It is commonly used together with GROUP BY. If you need to have some order for the XML pieces in the same group, you can specify ORDER BY inside XMLAGG. The argument is the same as the ORDER BY clause in the SELECT. The ordering keys have to be SQL types.

To nest groups, you can use SELECT with XMLAGG nested inside an upper XMLAGG function. This is very natural. There is an alternative though, it is to use GROUP BY one after another. But that is not as natural as nesting scalar fullselect with XMLAGG.

You can use these functions to generate almost all kinds of documents. If you need to generate recursive documents, you need to use nesting of the functions, as the SQL recursive query feature based on common table expression (CTE) does not fit this recursive needs. Contact me for the details of a solution if you need to generate recursive documents.

With V9 complete constructor functions, you can construct any document you want. In V9
1. We added binary data support: FOR BIT DATA, binary/varbinary, BLOB can be used in XML with BASE64 or HEX encoding. BASE64 is the default. And we also added new null handling options for V8 functions, including NULL ON NULL or EMPTY ON NULL. Specify NULL ON NULL if you don't want to generate an empty element when all the content is NULL for XMLELEMENT function.

2. We Added XMLTEXT, XMLPI, XMLCOMMENT, and XMLDOCUMENT constructors to make a complete set of constructors: if you construct a document and insert into an XML column, the top constructor must be XMLDOCUMENT as it's required to be a document to store in an XML column. If you want XMLAGG to generate mixed content, you would need XMLTEXT to wrap the text as XMLAGG takes XML values only.
3. Furthermore, all these functions take and generate real XML data type. They can take as input XML columns, or XMLQUERY result or XMLTABLE XML column result. When you use XML columns in constructor functions, the document nodes will be removed in construction.

To recap, we’ve talked about a set of SQL standard XML publishing functions in DB2 for z/OS V8 and DB2 9. They can be used to generate XML data, they’re easy to use and are much more efficient.

If you have any questions, please contact us by sending emails to db2zxml@us.ibm.com. Until next time, Good bye.