Eye On XQuery

By Justin Whitney

With the addition of "pureXML" to IBM DB2 9, database developers gain powerful XQuery functionality. For those new to the syntax (and for veterans seeking a faster way to build complex queries), the new DB2 Developer Workbench includes the Visual XQuery Builder, a versatile visual design tool for XML queries.

IBM's new DB2 9 gives developers a faster, more efficient way to manage XML documents within a relational database. However, in order to make the most of this potential, database developers need to add a new syntax to their arsenals: XQuery. While straightforward to learn, XQuery offers enough sophistication to make more advanced queries time-consuming and laborious to piece together.

Enter the DB2 9 Developer Workbench. Packaged as a separate install, the new Workbench takes over from the former DB2 Development Center. Based on the Eclipse engine, it offers advanced new XML functionality in Visual XQuery Builder, including the ability to visually design XQueries based on existing XML fields or separate XML schema documents. The Developer Workbench can also be used with DB2 for z/OS data servers.

Visual XQuery Builder

You can run some simple queries on your SAMPLE database to give the XQuery Builder a try. The version of the SAMPLE database that ships with DB2 9 includes several tables with populated XML fields, such as the CUSTOMER table, which has INFO and HISTORY fields of type XML. You can also create your own sample database with the command:

```
db2sampl -xml
```

To try out XQuery Builder, open up DB2 Workbench and start a new project: Select File → New Data Development Project or select Data Development Project from the New Project icon.

Next, call the project XQueryProject and leave the other defaults as they are. If you're already connected to SAMPLE, then choose it under Use an Existing Connection. If you're not yet connected, take these steps:

1. Select Create a New Connection.
2. Under Connection Parameters, select DB2 UDB v9.1.
3. Specify SAMPLE for the database name and leave the other defaults.
4. Enter your admin login and password (which will be your system login/password, unless you specified otherwise), and click Test Connection.
5. If it's good, click Next to continue.

In the last part of the New Project wizard, keep the defaults for Routine Parameters and click Finish.

After the Workbench creates your new project, you should see it in the Data Project Explorer on the upper left. Below it, in the Database Explorer, you'll see your SAMPLE database. If it's not already connected, then right click and choose Connect or Reconnect and enter your credentials.

Expand the items in both of these windows. Notice that under your new project in the Data Project Explorer, you have several empty folders:

- XML Queries
- SQL Scripts
- Stored Procedures
- XML Schema Documents
- User-Defined Functions.

Part of the power of the Developer Workbench lies in the collection of wizards it provides, most of which can be launched contextually from these folders. Here, you can add XSD files to your project to facilitate XML manipulation. You can build new stored procedures, scripts, or even your own functions. But for this walkthrough, you'll focus on the first folder: XML Queries.

Create an XML Query

You can create your first XML query by launching the New XML Query wizard. First, right click on the XML queries folder in your Data Project Explorer pane and choose New XML Query. Call this SimpleXQuery.

When given the option to Add representative XML documents, click Add. Here you have a couple of options. You can have the Workbench discover and create an XML schema for you based on sample data, or you can use an existing schema:

- Local workspace: If you've already generated a schema for the data set you want to use, or you've saved an existing XSD or DTD document to your workspace, then choose it and locate the schema.
- Database: If you want to create a new schema based on existing XML data, choose the Database option.

For this example, select Database. For the XML column or schema, navigate to CUSTOMER.INFO (see Figure A online if you're not sure how to find it).

On the next screen of the wizard, you can associate the documents with the specific XML document on which you'll be basing the query. In the SAMPLE database, each row can contain an XML document with a different schema. Here, you can navigate through the rows to select the schema you want to use. For this example, select Document 2 (see Figure 1).

FIGURE 1. XML document rows.
In the last stage of the wizard, you're asked to associate the documents with the XML columns. If you had been using an existing XML schema, then you would need to associate it here. But this example is based on the schema of existing data, so the schema document is associated by default.

Now you've created a new XQuery View under the name you provided, SimpleXQuery.xqm. Take a moment to explore the view. You'll see that sample data is listed using the XML schema of the first row of the specified column. You'll also see plenty of syntax you can drag and drop into the design workspace, including Constructors, Expressions, Functions, and Variables, all based on the XQuery spec currently under review by the World Wide Web Consortium (W3C).

Build a Visual XQuery

With your new XQuery grid in place, you'll create a simple query with some conditionals.

Expand the Constructors category and drag a new Element onto the grid. For each item you drag onto a grid, you'll see a Step Into icon to its right. Click the one for new_element.

Change the Name of the element to toronto (see Figure 2). Don't worry about the Attributes. Under Element Value, drag the name node. Notice that the name node now has a Step Into icon as well. Click it.

FIGURE 2. XML element <toronto>.

In the Visual Builder, each time you click Step Into, you bring up a subpage unique to that item. For XML nodes, it brings up the For, Let, Where, Order by, Return diagram, or FLWOR. This diagram, basically the equivalent of the SELECT-FROM-WHERE clause in SQL, allows you to define selection criteria for your query.

The Workbench automatically assigns the binding variable, which is essential to the FLWOR statement, but you can change it. In this example, it's called name0. The Sequence and Return values in this example are both name nodes by default. You only need to define the conditional. To do so, expand the Functions category on the left and, if not already selected, choose String Functions from the drop-down menu. You have a variety of built-in functions to choose from, giving you a lot of latitude in how you want to define your WHERE clause. The goal here is to choose all the customers from Toronto. The most direct method would be to drag the city node under Operand 1, choose = as the Operator, and type Toronto for Operand 2. You could also use the match() function to define a pattern, or mix any number of string functions together to create more explicit criteria. To keep things simple, drag the contains() function and drop it under Operand 1 (see Figure 3).

FIGURE 3. FLWOR with contains() function.
Because the `contains()` function returns True or False, you don't need to assign a second operand. Click the Step Into icon for this function to define its arguments. The `contains()` function takes two arguments: string and substring. Drag the city node onto the string argument. Next to substring, type onto. This will pull any city that contains the letters onto, all lower-case (see Figure 4). This step is similar to `LIKE '% onto%'` in SQL.

**FIGURE 4.** Defining the `contains()` function.

In the upper-right corner of the window, you'll see a Step Out icon. Click it or select OK from the drop-down menu to go up a level. In your For Logic definition, add one more field. Drag the phone element to the Return grid under name. Then click Step Out again.

Click Step Out one more time to return to the root level node. Here, you can continue adding additional nodes. For now, run the query and take a look at the results. From the icon on the upper-right, choose Run... from the drop-down menu or choose Run → Run... from the main menu.

You'll have another chance to associate the XML column with a definition document, but since it's already associated correctly, just click Finish. In the Data Output view, you should see `<toronto> ...` Click the ellipsis to see the full result set.

You can view the results in two formats: XML Tree (see Figure 5) or Source, which is an unformatted text dump. In this simple example, all the results have been collected under a single root node, including any duplicate customers. However, with some minor tweaking, you could change the node hierarchy, add additional fields, remove duplicates, or add aggregating functions.

**FIGURE 5.** Results of sample XQuery.

You can also Save As... an external file. In fact, if you click Save As and save to demo.xml, you can open this file in a Web browser and see a basic (though not particularly well-formed) XML document with the results of your query.

**Last but Far From Least**

Now we've gotten to what I think is the Visual Builder's best feature. Go back to the grid and choose the Source tab on the lower-left. You'll see autogenerated XQuery code that describes your query. The bulk of this code can be used directly in a database query or stored...
procedure, which makes this a fast and efficient method for building XQueries on the fly. The example we've been working on generated the source code show in Listing 1.

Learn by Doing

This example just begins to explore the potential of the Visual XQuery Builder. In addition to helping you create complex nested XQueries, it can also help you learn XQuery syntax by generating source code that you can examine or use elsewhere. You can learn a lot more about the Visual XQuery Builder by looking through its Help files. DB2 9 Developer Workbench ships with extensive contextual documentation to help you get started. Also, be sure to check out the articles and Web sites listed in Resources.

Justin Whitney is a regular contributor to DevX.com and Jupitermedia. He currently lives in San Francisco, where he consults for leading high-tech firms and writes about emerging technologies.

Resources

DB2 9

"An Introduction to XQuery," Howard Katz, IBM developerWorks, January 6, 2006

"Integration of SQL and XQuery in IBM DB2," F. Ozcan, D. Chamberlin, K. Kulkarni, and Jan-Eike Michels, IBM Systems Journal, Volume 45, Number 2, 2006

"Query DB2 XML data with XQuery," Don Chamberlin and Cynthia M. Saracco, IBM developerWorks, April 6, 2006

"Query DB2 XML Data with SQL," Cynthia M. Saracco, IBM developerWorks, March 16, 2006

XML Query Use Cases

Return to Article