Integrate XML into your enterprise using WebSphere Federation Server Version 9.5

Skill Level: Intermediate

Ioana Ursu (ursu@us.ibm.com)
Advisory Software Engineer
IBM

Aakash Bordia (bordia@us.ibm.com)
Advisory Software Engineer
IBM

Eileen Lin (etlin@us.ibm.com)
Senior Technical Staff Member
IBM

17 Apr 2008

XML Federation in WebSphere® Federation Server Version 9.5 allows you to seamlessly integrate with remote XML data from XML documents and DB2® for Linux®, UNIX®, and Windows® (DB2) sources enabled with pureXML technology. Not only can you use the SQL/XML and XQuery languages to manipulate remote XML data, you can also perform other operations such as decomposition, schema validation, and executing XML utilities. In this article, you'll walk through various use cases to help you build your distributed XML applications quickly.

Introduction

IBM WebSphere Federation Server provides real-time, virtualized access to disparate data sources. As a result, you benefit from delivering new projects with a much shorter time to market, extending your warehouse with ad-hoc queries, and producing a unified view for your enterprise data.

With the XML Federation feature, WebSphere Federation Server Version 9.5 provides direct access to remote XML data in DB2 and XML documents. You can
create a relational nickname over a remote table or view that contains the XML data type. You can also use the XML wrapper to create a nonrelational nickname that contains the XML data type over XML documents.

You can use these nicknames in the XQuery and SQL languages. The XQuery language is the primary mechanism for querying XML documents. You can use SQL to perform basic operations, such as selecting XML columns and inserting, updating, or deleting XML data. You can also integrate SQL and XQuery to create queries for both existing relational data and XML data by using SQL/XML functions and predicates, and XQuery functions.

In addition, you can perform a sub-document update, which allows portions of an XML document to be updated. Also, the federated server can validate remote XML data. XML validation is the process of determining whether the structure, content, and data types of an XML document are valid. Finally, with annotated XML schema decomposition, you can decompose documents in columns of one or more nicknames.

This article demonstrates how you can take advantage of this XML Federation feature in WebSphere Federation Server Version 9.5. Using a scenario about books, inventory, and reviews, you will walk through various use cases to demonstrate the use of the SQL/XML and the XQuery languages to manipulate your remote XML data. This article also covers related XML support such as IMPORT, EXPORT, XML schema validation, and decomposition. For those of you who would like to try out the examples, the appendix includes sample data and configuration scripts.

Work with remote XML data

The examples in this article to the scenario described in Figure 1.

Figure 1. The books scenario
The following table, nicknames, and view are defined on the federated server:

- Table BOOKS contains information about the books. Book information is stored in column "book" of type XML.

```
CREATE TABLE books (isbn char(13), book xml);
```

- Nickname BOOKS_DB2 defined over a remote DB2 table called BOOKS. The remote table BOOKS also contains information about books stored as type XML.

```
CREATE NICKNAME books_db2 FOR drda_server.myuser.books;
```

- Nickname INVENTORY_DB2 defined over a remote DB2 table INVENTORY.

```
CREATE NICKNAME inventory_db2 FOR drda_server.myuser.inventory;
```

- Nickname REVIEWS_XML defined over a set of XML documents stored in files of a directory using the XML wrapper.
CREATE NICKNAME reviews_xml (book XML)
    FOR SERVER xml_server OPTIONS (DIRECTORY_PATH '/home/myuser/xml');

• UNION ALL view BOOKS_VIEW that brings together the data from the local table BOOKS and nickname BOOKS_DB2.

CREATE VIEW books_view(isbn, book) AS
    SELECT isbn, book FROM books_db2
    UNION ALL
    SELECT isbn, book FROM books;

Next, you will walk through some typical use cases of pureXML features that WebSphere Federation Server supports for remote data.

Construct XML values from remote relational data

The following example shows how to construct XML values from remote relational data. If the data is stored in remote relational tables and needs to be published as XML, you can use the SQL/XML publishing functions such as XMLELEMENT, XMLAGG, and XMLATTRIBUTE.

Using the nickname INVENTORY_DB2, defined above, the following query groups together the books under each bookstore ID.

SELECT XMLELEMENT(NAME "bookstore",
    XMLATTRIBUTES(a.bookstore_id as "id"),
    XMLAGG(XMLELEMENT(NAME "book",
        XMLELEMENT(NAME "isbn", a.isbn),
        XMLELEMENT(NAME "quantity", a.quantity)))) as bookstores
FROM inventory_db2 a
GROUP BY a.bookstore_id;

A sample output is shown below.

```xml
<bookstore id="111">
    <book>
        <isbn>0-201-56317-7</isbn>
        <quantity>10</quantity>
    </book>
    <book>
        <isbn>0-201-63346-9</isbn>
        <quantity>5</quantity>
    </book>
</bookstore>

<bookstore id="222">
    <book>
        <isbn>0-738-49468-2</isbn>
        <quantity>4</quantity>
    </book>
</bookstore>
```
The "bookstore" element is built using the outermost XML_ELEMENT scalar function. The books are grouped under a bookstore ID using XMLAGG function. XMLAGG returns an XML sequence containing an item for each non-null value in a set of XML values.

**Extract remote XML values into relational format**

To do the inverse transformation, which is to extract the remote XML data and present it in the relational format to the application, you can use the XMLTABLE table function. XMLTABLE is an SQL table function that returns a table from the evaluation of XQuery expressions. The example uses XMLTABLE to extract the first and last names of the authors from the BOOKS_VIEW for those books published under publisher code '0-201' that cost more than $50.00.

```sql
SELECT a.isbn, b.first, b.last
FROM books_view a,
XMLTABLE('$BOOK/book[price>50.00]' COLUMNS
    First varchar(50) path 'author/first/text()',
    Last varchar(50) path author/last/text()') as b
WHERE isbn LIKE '0-201-%';
```

A sample output is shown below:

```
0-201-56317-7 Richard Stevens
```

**Full XML document retrieval using SQL-only queries**

The next example shows a simple, full XML document retrieval using a SQL-only query. This example queries the BOOKS_VIEW based on the ISBN column:

```sql
SELECT book
FROM books_view
WHERE isbn LIKE '0-201-%';
```

The output of the query is shown below:
Access XML data using XQuery

The next example finds the titles of the books published under "Addison-Wesley" publishing house after the year 1990 using XQuery. The XQuery language is designed to navigate through XML document trees and extract XML fragments. It includes expressions to create, manipulate, and iterate over sequences of XML items and construct new XML data.

XQUERY
FOR $book in db2-fn:xmlcolumn('BOOKS_VIEW.BOOK')/book
WHERE $book/publisher = 'Addison-Wesley' and $book/@year > 1990
RETURN <new year='{$book/@year}'>
  {$book/title/text()}
</new>;

The output of the query is shown below. The FOR clause iterates over the documents in the BOOKS_VIEW; the WHERE clause filters the rows based on the publisher and year. Then the RETURN clause generates the result of the XQuery. Collectively the clauses are part of the so-called FLWOR expression.

Combined access to XML and relational data

XQuery allows you to access XML data. If you need combined access to XML and relational data, you can use either XQuery embedded in SQL or XQuery with embedded SQL. A few examples are shown below.

For SQL-based applications that require XML fragments retrieval or XML predicates to be applied, you can use SQL/XML with XQuery embedded in SQL. In particular, a XMLQUERY scalar function is used to extract fragments from XML columns and a
XMLEXISTS predicate is used to express predicates over XML data.

The following query retrieves data from the BOOKS_VIEW UNION ALL view. The XMLEXISTS predicate restricts the selection to those books that cost more than $50 and the XMLQUERY function extracts the authors from the XML documents.

```sql
SELECT isbn, XMLQUERY('FOR $author in $BOOK/book/author
RETURN $author' returning sequence)
FROM books_view
WHERE XMLEXISTS ('$BOOK/book[price>50.00]')
and isbn LIKE '0-201-%';
```

A sample output is shown below.

```
0-201-56317-7
<author><last>Stevens</last><first>Richard</first></author>
```

In the above example, you can also use parameter markers when retrieving data from remote XML documents. You can do this by passing a SQL parameter marker as a variable into the embedded XMLEXISTS predicate. In the next example, instead of the actual price value variable $x is passed to the XMLEXISTS predicate.

```sql
SELECT isbn, XMLQUERY('FOR $author in $BOOK/book/author
RETURN $author' returning sequence)
FROM books_view
WHERE XMLEXISTS ('$BOOK/book[price>$x]'
passing cast(? AS double) as "x")
and isbn LIKE '0-201-%';
```

Now, if you have XML-based applications and want to process only a subset of the XML documents based on conditions on relational columns, you can use XQuery with embedded SQL instead, as shown in the next example.

```xquery
FOR $book in db2-fn:sqlquery("SELECT book FROM books_view
WHERE isbn like '0-201-%'")/book
WHERE $book/@year > 1990
RETURN <new year='{$book/@year}'>
  {$book/title/text()}
</new>;
```

The sqlquery function invoked in the FOR clause retrieves a sequence that is the result of an SQL select. This example is another variation of the previous examples. In this case, the query finds the titles published under publisher code '0-201' after the year 1990. A sample output is shown below:

```
<new year="1998">Advanced Programming in the Unix environment</new>
```
Join XML and relational data

To join XML and relational data, you can pass relational data as a variable into the XMLEXISTS predicate. The next query finds the reviews of the books published under publisher code '0-201'. The value of the isbn column is passed to XMLEXISTS predicates and is used to restrict the REVIEWS_XML documents.

```sql
SELECT b.review
FROM books_view a, reviews_xml b
WHERE XMLEXISTS ('$REVIEW/entry[isbn=$m]' passing a.isbn as "m")
    and isbn LIKE '0-201-%';
```

The output of the join is shown below.

```xml
<entry>
  <isbn>0-201-56317-7</isbn>
  <reviewer>
    <last>Parziale</last>
    <first>Lydia</first>
  </reviewer>
  <review>A clear and detailed discussion of UNIX programming</review>
</entry>
```

Join remote XML data

If you need to join remote XML data, you can use XQuery. The next query finds the authors who are also reviewers by iterating over the documents in the BOOKS_VIEW view and the REVIEWS_XML nickname.

```xquery
FOR $author in db2-fn:xmlcolumn('BOOKS_VIEW.BOOK')/book/author
FOR $reviewer in
db2-fn:xmlcolumn('REVIEWS_XML.REVIEW')/entry/reviewer
WHERE $author/last = $reviewer/last and
$author/first=$reviewer/first
RETURN $author;
```

The output is shown below. The values of the last and first name are compared in the WHERE clause. The query returns the content of the author XML element.

```xml
<author>
  <last>Parziale</last>
  <first>Lydia</first>
</author>
```

Group and aggregate of remote XML data using SQL/XML
You also can group and aggregate remote XML data. The next query provides an example. The XMLTABLE is used in conjunction with the SQL GROUP BY and aggregation functions. The query counts the books, grouped by author.

```
SELECT b.last, b.first, count(b.title)
FROM books_view a, XMLTABLE('"BOOK/book"
COLUMNS
  'author/last/text()' first varchar(50) path,
  'author/first/text()' last varchar(50) path,
  'title/text()' title varchar(50) path
) as b
GROUP BY b.last, b.first;
```

A sample output is shown below.

<table>
<thead>
<tr>
<th>Chamberlin</th>
<th>Don</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Klein</td>
<td>Scott</td>
<td>1</td>
</tr>
<tr>
<td>Parziale</td>
<td>Lydia</td>
<td>1</td>
</tr>
<tr>
<td>Stevens</td>
<td>Richard</td>
<td>2</td>
</tr>
</tbody>
</table>

Insert, update, or delete remote XML data

XML documents can be inserted, updated, and deleted using SQL data manipulation statements. WebSphere Federation Server 9.5 introduces support for insert, update, and delete operations on remote DB2 nickname columns that are type XML. Import and export commands are supported and new pureXML features (introduced in DB2 9.5) such as sub-document update are also supported.

The following is an insert example.

```
INSERT INTO books_db2 VALUES( '0-201-56317-7',
  '<book year="1998">
    <title>Advanced Programming in the Unix environment</title>
    <author><last>Stevens</last><first>Richard</first></author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>');
```

As an example for sub-document update, increase the prices of all books in the remote DB2 books table by five percent. The XQuery updating expressions allow you to delete, insert, replace, or rename nodes in an XML document.

```
UPDATE books_db2 SET book = XMLQuery ('transform
copy $r := $BOOK
modify (do replace value of $r/book/price with $r/book/price * 1.05 )
return $r');
```
You can also export data from an XML column in a nickname, or import data to an XML column in a nickname. The following simple statements show these capabilities:

```sql
EXPORT TO bookxml.del of del SELECT * FROM books_db2;
IMPORT FROM bookxml.del of del xml FROM /home/myuser
ALLOW WRITE ACCESS COMMITCOUNT 1 INSERT INTO books_db2;
```

**XML validation**

In DB2, XML column values are not constrained to an XML schema. The validation has to be done explicitly by the user with the XMLVALIDATE function. By default, the parser is invoked with no schema validation. It only checks for well-formedness. An XML schema repository is added to store XML schemas, DTDs, and other XML artifacts as objects in the database for use when processing (validating) the XML documents that reference them.

The XMLVALIDATE function validates the given XML value against the given XML schema, or the XML schema obtained from the schema specification in the instance document. It checks whether the structure, content, and data types of an XML document are valid. If XMLVALIDATE is explicitly called with the nickname column as an input, it sets the validation option to ON when calling the XML4C parser under the covers. The user can register a remote XML schema in the DB2 schema repository by using the DB2 XML schema registration process. For more information about schemas, see DB2 UDB SQL Reference, Volume 1 and The DB2 UDB SQL Reference, Volume 2.

The following cases can occur when invoking the XMLVALIDATE function:

- If the XMLVALIDATE is called without any schema specification, the schema is deduced from the "schemaLocation" attribute in the instance document itself. If the schema information doesn't exist, the parser issues an error message (SQL16196N).

- If XMLVALIDATE is called with a registered schema or a specific URI, the validation is performed against that specific schema. If an external schema is specified, it overwrites the internal schema specification. For federated statements, this is the mechanism to overwrite the internal schema specification in the document if the schema information is not accessible by the DB2 server.

For more information about XMLVALIDATE, see DB2 UDB SQL Reference, Volume 1 and The DB2 UDB SQL Reference, Volume 2.
To demonstrate using XMLVALIDATE, ensure that the books_view presents valid XML data. books_xsd is a local DB2 registered schema object, which may represent a remote data source XML schema. You can use the REGISTER XMLSCHEMA command to register the bookstore schema with DB2, as shown in the following example.

REGISTER XMLSCHEMA 'http://test.ibm.com/books.xsd'
FROM 'file:///home/myuser/books.xsd' AS books_xsd COMPLETE ENABLE DECOMPOSITION;

CREATE VIEW books_view (isbn, book) as
SELECT isbn, XMLValidate(book ACCORDING TO XMLSCHEMA ID books_xsd)
FROM books_db2
UNION ALL
SELECT isbn, XMLValidate(book ACCORDING TO XMLSCHEMA ID books_xsd)
FROM books;

Annotated XML schema decomposition

Decomposition is the process of storing content from an XML instance document in columns of relational tables, where the data type of the columns is not XML. In other words, an XML document can be "shredded" in columns of relational tables into one or more DB2 nicknames. The decomposition operates based on annotations specified in the XML schema. Here are the steps needed for decomposition:

1. Create the nicknames that you want to use for decomposition.
2. Annotate the schema documents with XML schema annotations.
3. Register the schema using the REGISTER XMLSCHEMA command.
4. Enable the schema for decomposition using the ALTER XSROBJECT statement.
5. Use the DECOMPOSE XML DOCUMENT command to decompose the XML instance document.

As an example of decomposition, extract the authors of all books and insert them into a remote table called AUTHOR by specifying the nickname AUTHOR in the annotated schema (note that the schema from the section above is reused, which has been enabled for decomposition).

DECOMPOSE XML DOCUMENT /home/myuser/books.xml XMLSCHEMA book_xsd
VALIDATE;
Conclusion

XML Federation in WebSphere Federation Server Version 9.5 is a new feature that provides transparent integration with XML data in XML documents and DB2 sources enabled with pureXML technology. The ability to combine remote XML data with other relational data helps users exploit the advantages of both relational and XML data models and allows users to:

- Create a simpler view of their information infrastructure
- Reduce programming costs in new application development
- Deliver needed business information faster
- Exchange data with ease between different systems, platforms, applications, and organizations

The rich and flexible XML data manipulation that the XQuery and SQL/XML languages provide, combined with the comprehensive pureXML support of remote XML data, makes WebSphere Federation Server Version 9.5 an ideal component in your enterprise solutions that require the coexistence of XML and relational data.

Appendix: Sample data and configuration scripts

The following table and data are created on the remote DB2 server.

```sql
CREATE TABLE BOOKS(ISBN CHAR(13), BOOK XML);
INSERT INTO BOOKS VALUES( '0-201-56317-7',
    '<book year="1998">
        <title>Advanced Programming in the Unix environment</title>
        <author>Richard Stevens</author>
        <publisher>Addison-Wesley</publisher>
        <price>65.95</price>
    </book>');

INSERT INTO BOOKS VALUES( '0-201-63346-9',
    '<book year="1994">
        <title>TCP/IP Illustrated, Volume 1</title>
        <author>Richard Stevens</author>
        <publisher>Addison-Wesley</publisher>
        <price>45.95</price>
    </book>');

INSERT INTO BOOKS VALUES( '0-738-49468-2',
    '<book year="2006">
        <title>TCP/IP Tutorial and Technical overview</title>
        <author>Lydia Parziale</author>
        <publisher>IBM Redbook</publisher>
        <price>49.99</price>
    </book>');

CREATE TABLE INVENTORY (BOOKSTORE_ID INT, ISBN CHAR(13), QUANTITY INT);
```
<table>
<thead>
<tr>
<th>SQL Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert into inventory values (111,'0-201-56317-7',10);</td>
</tr>
<tr>
<td>insert into inventory values (111,'0-201-63346-9',5);</td>
</tr>
<tr>
<td>insert into inventory values (222,'0-738-49468-2',4);</td>
</tr>
<tr>
<td>insert into inventory values (222,'1-55860-482-0',6);</td>
</tr>
<tr>
<td>insert into inventory values (222,'0-7645-9792-2',4);</td>
</tr>
</tbody>
</table>

The following tables, nicknames and data are created on the WebSphere Federation Server.

Create a DRDA wrapper, server, and nickname:

```sql
create wrapper drda_wrapper library 'libdb2drda.a';
create server drda_server type db2/udb version 9.1 wrapper
drda_wrapper
   authorization "myuser" password "mypassword"
   options (dbname 'v91data');
create user mapping for user server drda_server
   options (remote_authid 'myuser', remote_password
   'mypassword');
create nickname books_db2 for drda_server.myuser.books;
create nickname inventory_db2 for drda_server.myuser.inventory;
```

Create local table BOOKS:

```sql
create table books(isbn char(13), book XML);
insert into books values( '1-55860-482-0',
                       '<book year="1998">'
                       '<title>DB2 Universal database</title>
                       <author><last>Chamberlin</last><first>Don</first></author>
                       <publisher>Morgan Kaufmann</publisher>
                       <price> 40.00</price>
                       '</book>');
insert into books values( '0-7645-9792-2',
                       '<book year="2006">
                       <title>Professional SQL Server 2005 XML</title>
                       <author><last>Klein</last><first>Scott</first></author>
                       <publisher>Wiley Publishing Inc</publisher>
                       <price>49.99</price>
                       '</book>');
```

Create UNION ALL view BOOKS_VIEW:

```sql
create view books_view(isbn, book) as
select isbn, book from books_db2
union all
```

Integrate XML into your enterprise using WebSphere Federation Server Version 9.5
© Copyright IBM Corporation 1994, 2007. All rights reserved.
select ISBN, book from books;

Create XML wrapper, server, and nickname:

create wrapper xml_wrapper library 'libdb2lsxml.a';
create server xml_server wrapper xml_wrapper;
create nickname reviews_xml
(review xml)
for server xml_server
options (directory_path '/home/myuser/xml');

There are two files in this directory. The first file contains the following record:

<entry>
<reviewer><last>Parziale</last><first>Lydia</first></reviewer>
<review>A clear and detailed discussion of UNIX programming</review>
</entry>

The second file contains the following record:

<entry>
<reviewer><last>Thomas</last><first>Scott</first></reviewer>
<review>A good computer networking book.</review>
</entry>
Resources

Learn

- **Product Overview: IBM WebSphere Federation Server V9.5**: Visit this site to learn more about WebSphere Federation Server.

- "Using data federation technology in IBM WebSphere Information Integrator: Data federation design and configuration" (developerWorks, June 2005): Discover federated data concepts and read about design and configuration considerations for WebSphere Information Integrator (the former product name of WebSphere Federation Server).

- "Using data federation technology in IBM WebSphere Information Integrator: Data federation usage examples and performance tuning" (developerWorks, June 2005): Learn about federated query optimization, find usage examples, and familiarize yourself with performance tuning considerations for WebSphere Information Integrator.


- "Webcast: Integrating XML data using federation server" (developerWorks, Feb 2008): Learn how IBM's WebSphere Federation Server technology gives you the ability to integrate traditional tabular structured data with XML formatted data.

- "IBM WebSphere Information Integration Administration Guide for Federated Systems": Read the basic product documentation for WebSphere Federation Server.

- "The DB2 UDB SQL Reference, Volume 1" and "The DB2 UDB SQL Reference, Volume 2": Find the fundamental references for SQL use with DB2 for Linux, UNIX, and Windows.

- "Query DB2 XML Data with SQL" (developerWorks, March 2006): Learn how to query data stored in XML columns using SQL and SQL/XML.

- "Query DB2 XML data with XQuery" (developerWorks, April 2006): Learn how to query data stored in XML columns using XQuery.


- "DB2 9 pureXML Guide": This IBM Redbook offers a broad understanding of the DB2 9 pureXML.

- **developerWorks Information Management zone**: Learn more about DB2. Find technical documentation, how-to articles, education, downloads, product
information, and more.

- Stay current with developerWorks technical events and webcasts.

### Get products and technologies

- Build your next development project with IBM trial software, available for download directly from developerWorks.

### Discuss

- Participate in developerWorks blogs and get involved in the developerWorks community.

## About the authors

### Ioana Ursu

Ioana Ursu is an advisory software engineer in the Silicon Valley Laboratory in San Jose, Calif. She joined IBM Almaden in 1998 working for the Garlic research project. Since 1999, she has worked in many areas of federated query compilation, including query semantics, query rewrite, pushdown analysis, and query optimization. She currently works in the WebSphere Information Integrator Federated Query Compiler team, focusing on general federated query processing.

### Aakash Bordia

Aakash Bordia is an advisory software engineer at the Silicon Valley Laboratory in San Jose, Calif. After graduating from the University of Illinois in January 2000, he joined the IBM DB2 Warehouse Manager development team and contributed to SQL-based warehouse transformations. In 2002, he moved to the WebSphere Federation Server team where he has been a compiler/optimizer and QA engineer.

### Eileen Lin

Dr. Eileen Lin is a senior technical staff member in the Silicon Valley Laboratory in San Jose, Calif. She is one of the original members responsible for the success of DataJoiner, a federated database product that is the predecessor of the federation technology in DB2. Currently, she is the lead architect for WebSphere Federation Server. Dr. Lin has many patents covering areas such as federation technology, query optimization, and parallel query processing.