IBM DB2 XML Extender

An end-to-end solution for storing and retrieving XML documents.

An end-to-end solution for storing and retrieving XML documents.

Josephine Cheng, Jane Xu

IBM Silcon Valley

chengjm@us.ibm.com,
jxu@us.ibm.com

Abstract

The Extensible Markup Language (XML) is a key technology that facilitates information exchange and e-business transactions. Starting with IBM DB2® Universal Database™ Net.Data®, Version 1, an application can generate XML documents from SQL queries against DB2® or any ODBC-compliant databases. Today DB2 XML Extender not only serves as a repository for XML documents and their document type definitions (DTDs), but also provides data management functionalities, such as data integrity, security, recoverability and manageability. A user has the option to store an entire document as an XML user-defined column or to decompose the document into multiple tables and columns. Fast search using indices is provided for XML elements and attributes. Section search can be done against the content of the document. Query syntax adheres to W3C standards, such as Extensive Stylesheet Language Transformation (XSLT) and XML Path Language (XPath) specifications. A user can retrieve an entire document or extract XML elements and attributes dynamically in an SQL query. In addition, XML Extender provides stored procedures to generate XML documents from existing data. Together with Net.Data, one can browse the content of the XML documents through the Internet.

Introduction

The Extensible Markup Language (XML) is a simplified subset of the Standard Generalized Markup Language (SGML), a text-based data format for structured documents. XML is a tag language that looks similar in style to the HyperText Markup Language (HTML). However, it supports a much richer set of features, such as user-definable tags. XML is also a metalanguage for defining other markup languages specialized to the needs of an industry (for example, RELML, HL7) or discipline (for example, WML, VoiceML). The metadata is defined in document type definition (DTD), or XML grammars, which specifies a set of tags and the rules connecting the tags and their contents. There is a set of standards that is related to XML. Some of the better known standards include Document Object Model (DOM), Simple API for XML (SAX), Extensible Stylesheet Language (XSL), XML Linking Language (XLink), XML Pointer Language (XPointer), XML Query Language (XQL), Extensive Stylesheet Language Transformation (XSLT) and XML Path Language (XPath). An XML tutorial can be found at the IBM developerWorks™ Web site.

XML is perceived as the next wave of the Internet technology with the potential of replacing HTML for several reasons. First, XML allows customized tags to add semantics to the Web content page. Today's search engine is mostly based on words appearing in the content page rather than on the meaning of the search query. For example, a search on *Thinkpad*® will give you all documents which contain the word *Thinkpad* when the user is more interested in product information pages on *Thinkpad*. To solve this problem, one can use XML-customized tags to classify data with semantics such as *product* and *related-product*, among others. Second, different applications can communicate and extract information from the same XML document as long as they use the same DTD. Third, XML allows the separation of data and presentation directives, specified through XSL, in a document. As a result, the same data can be presented in different formats according to particular criteria, such as output device type or business purpose. Finally, the availability of IBM alphaWorks® XML Parser for Java™ as freeware promotes the popularity of XML usage. With the increase in popularity of the use of XML pages, one needs a repository to manage these documents and provide a fast search capability. IBM DB2 XML Extender is designed to provide these capabilities. The software is included in DB2 Universal Database, Version 7.1.

In today's just-in-time business paradigm, the ability to communicate efficiently among a variety of applications from suppliers, vendors and partners is vital. For example, in the automobile-manufacturing world, designers need to send design specifications to their suppliers, which in turn need to send a subset of the specifications to their subcontractors. Although these business entities all have different hardware and software environments, XML can serve as the common data format for data interchange. Another example is a real estate agency system that uses DB2 as its data repository. One of its requirements is to send multiple listings stored in DB2 to other agency systems which may use flat files, IMS® or other databases. RELML, XML for Real Estates, is used as the common data format among these heterogeneous systems. For new applications, you can store these RELML documents in XML Extender, which provides the capability to search listings based on a set of client criteria, and send the qualifying RELML documents to another real estate agency. For existing DB2 applications, XML Extender provides stored-procedures that can generate XML documents from existing DB2 data.

The browser has increasingly become a universal client for conducting e-business on the Internet. Recently released browsers, such as Internet Explorer V5 (IE5), support the presentation of XML documents with XSL stylesheets. You can invoke DB2 XML Extender in a data access JavaBeans™ through WebSphere® or in a Net.Data macro returning XML documents to IE5. You can also generate an XML document with queries against DB2 or any ODBC-compliant database using Net.Data.

XML documents and DTDs repository

XML Extender serves as a repository for XML documents as well as their DTDs. There are two options, XML column and XML collection, for storing an XML document. When using the XML column option, an entire XML document is stored as an XML user-defined type column. XML Extender provides three XML user-defined types (UDTs): XMLCLOB, XMLVARCHAR and XMLFile. XMLCLOB and XMLVARCHAR store an XML document as a CLOB and a VARCHAR, respectively in DB2 while XMLFile stores an XML document as a file on a local file system. Data access definition (DAD) is used to define which XML elements or attributes should be indexed. User-defined functions (UDFs) are provided to insert, select (either an entire document or a fragment) and update operations.

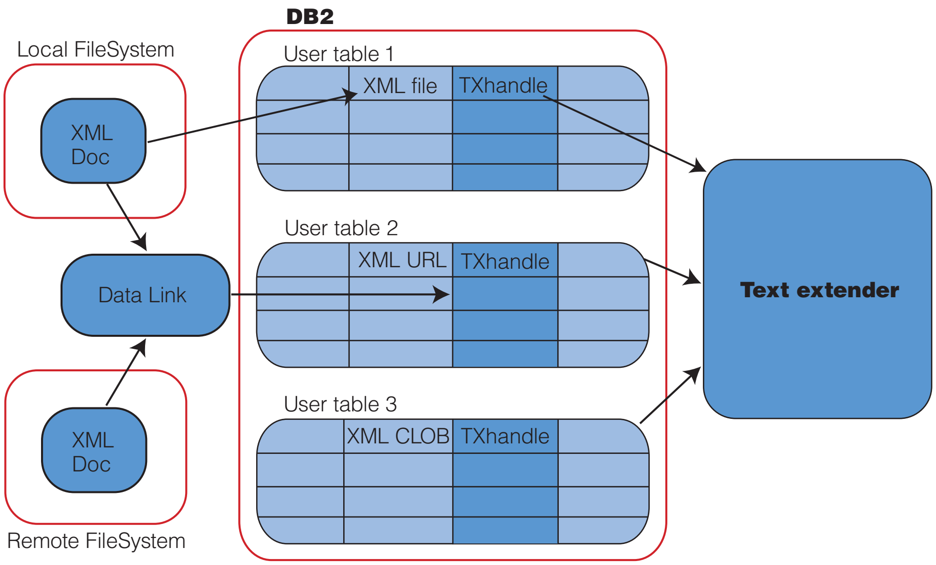

Figure 1. Overview of an XML document stored as an XML column.

An XML collection is a set of relational tables that contain data that is mapped to XML documents. The access and storage method allows you to compose an XML document from existing data, to decompose an XML document and to use XML as an interchange method. DAD is used to define the mapping of DTD to relational tables and columns. We use XSLT and XPath syntax for specifying the transformation and the location path. Unlike tags in an XML column, tags in an XML collection are not stored in the column of these tables. Stored procedures are provided for storing, retrieving, updating, searching and deleting data in an XML collection.

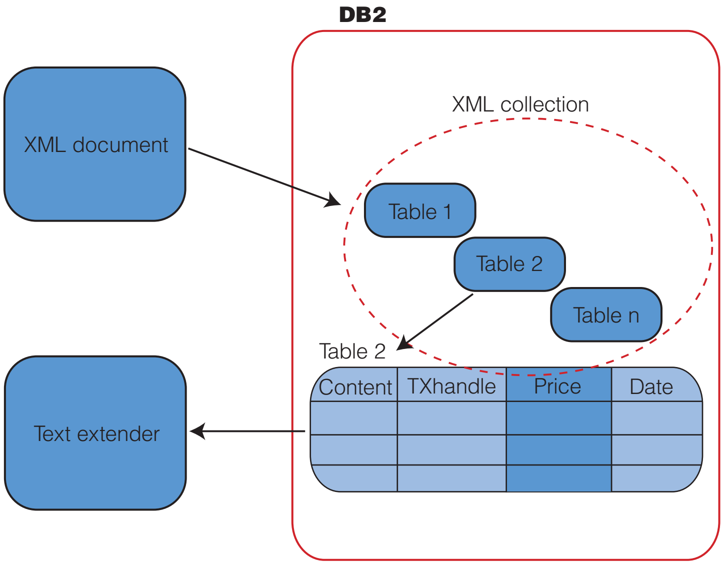

Figure 2. Overview of an XML document stored as an XML collection.

XML column provides a simple way to manage XML documents while providing fast search and indexing capabilities. It is ideal for data, such as purchases which do not require frequent updates. Also, if retrieval of an entire document is desired, performance is faster because there is no need to compose the XML document from DB2 data. On the other hand, XML collection provides a better mapping for an XML document which consists of multiple collections. For example, a book contains a set of chapters and each chapter contains a set of sections — both chapters and sections are XML collections. XML collection also supports existing DB2 data by providing a mechanism to generate an XML document for data interchange.

Searching XML documents

In this section, we show some examples of using SQL to search XML documents based on the elements and attributes of an XML document. The DAD defines PRICE element in the ORDER document for indexing and the side table is named PART_SIDETBL.

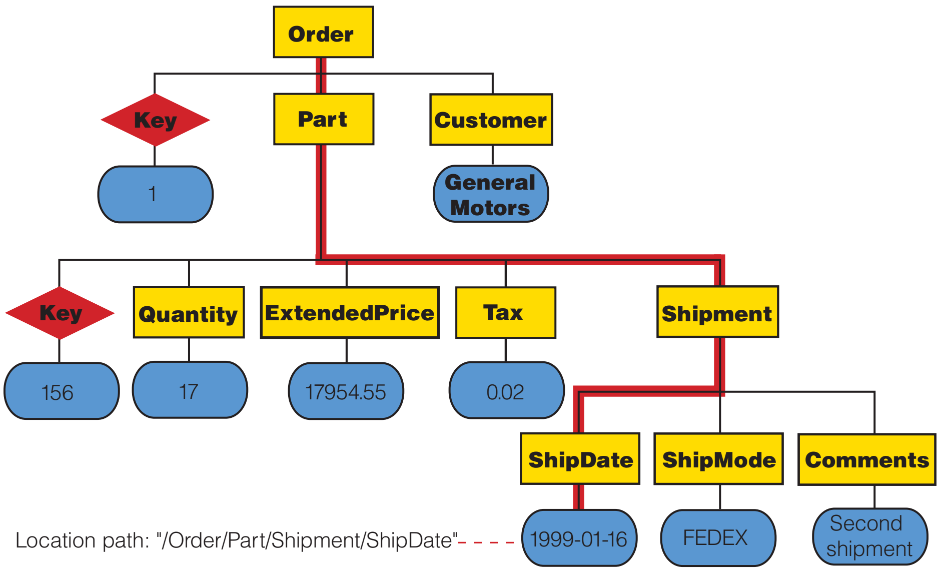

Figure 3. Example using SALES_TAB, a purchase order table with ORDER as an XML document and INVOICE_NUMBER as the primary key.

```
TABLE sales_tab
        invoice_number    CHAR(6) NOT NULL
                          PRIMARY KEY
        sales_person      VARCHAR(20)
        order             XMLVARCHAR
```

To find all sales persons who have purchase orders with a price over $2500.00, one can query directly against the side table, PART_SIDETBL as follows:

```
SELECT sales_person FROM sales_tab
        WHERE  invoice_number in
                      (SELECT invoice_number FROM part_sidetbl
                       WHERE extendedprice>2500.00)
```

You can also create a joined view combining the SALES_TAB and PART_SIDETBL so that query can be done directly against the joined view. The next example shows how one can query against the attribute to find all sales persons whose customer is *IBM* for invoice_number over 100.

```
SELECT sales_person FROM sales_tabl
        WHEREextractVarchar(order,
                      '/Order/Customer') LIKE "%IBM%'
               AND  invoice_number > 100
```

The above query is expensive because the UDF extractVarchar has to look into the content of the XML documents for CUSTOMER *IBM*. An alternative is to use DAD to define CUSTOMER as another index in the side table as to avoid using extractVarchar in the WHERE clause. Another alternative is to enable the XML column for the text extender so that text indexing can be used. The text extender provides section search (for example, / Order/Customer) with its UDF db2tx.contains.

For an XML document decomposed into XML collections, a search can be done directly against the SQL-based tables, or using the stored procedures dxxGenXML() and dxxRetrieveXML(). The DAD is used to specify whether to retrieve the entire document or a fragment, and the insertion can be conditional based on the values of XML elements or attributes. If the value of the element does not meet the condition specified in the DAD, the document will not be shredded and inserted into the table. Furthermore, the search criteria can be overridden dynamically at the invocation of the stored procedure as an optional parameter. Examples can be found in the XML Extender Administration and Programming Guide.

Generating XML documents from existing DB2 data

You can generate an XML document from SQL queries against DB2 or any ODBC-compliant databases using Net.Data. Figure 4 shows an example of Net.Data macro against an SQL query with user-defined tags around the results of the query.

```
%DEFINE DATABASE = “SAMPLE”
%DEFINE DTW_PRINT_HEADER = “NO”
%FUNCTION(DTW_SQL) QueryXML(){
   select * from staff where job = ‘Mgr’
     and years <= 5
%REPORT{
  <RowSet name=”QueryXML”>
   %ROW{
    <Row>
     <Column name=”$(N_ID)”>$(V_ID)</Column>
     <Column
      name=”$(N_NAME)”>$(V_NAME)</Column>
     <Column
      name=”$(N_DEPT)”>$(V_DEPT)</Column>
     <Column
      name=”$(N_JOB)”>$(V_JOB)</Column>
     <Column
      name=”$(N_YEARS)”>$(V_YEARS)</Column>
     <Column
      name=”$(N_SALARY)”>$(V_SALARY)</Column>
     <Column
      name=”$(N_COMM)”>$(V_COMM)</Column>
     </Row>
    %}
   </Rowset>
   %}
   %}
  %HTML(REPORT){
  Content-Type: text/xml
     <?xml version=”1.0"?>
   <?xml:stylesheet type=”text/xsl”
                 href=”http://malacandra.stl.ibm.com
           :8072/$(SS).xsl” ?>
     <XMLBlock version=”1" title=”Net.Data
           XML Demo”>
      @QueryXML()
     </XMLBlock>
%}
```

Figure 4. Net.Data Macro to generate XML Data

Figure 5 shows the generated XML data from the example in Figure 4. You can generate an XML document from heterogeneous data sources, such as files, DB2, Oracle and IMS Transaction Manager using Net.Data. It also supports *servlet direct* that allows users to specify the SQL statement or the stored procedure invocation directly in an Internet address, or URL.

```
Content-Type: text/xml
<?xml version=”1.0"?>
<?xml:stylesheet type=”text/xsl”

             Href=”http://malacandra.stl.ibm.com
             :8072/rowsetTable.xsl” ?>
<XMLBlock version=”1" title=”Net.Data XML Demo”>
    <RowSet name=”QueryXML”>
     <Row>
      <Column name=”ID”>30</Column>
      <Column name=”NAME”>Marenghi</Column>
      <Column name=”DEPT”>38</Column>
      <Column name=”JOB”>Mgr</Column>
      <Column name=”YEARS”>5</Column>
      <Column name=”SALARY”>17506.75</Column>
      <Column name=”COMM”></Column>
     </Row>
     <Row>
      <Column name=”ID”>240</Column>
      <Column name=”NAME”>Daniels</Column>
      <Column name=”DEPT”>10</Column>
      <Column name=”JOB”>Mgr</Column>
      <Column name=”YEARS”>5</Column>
      <Column name=”SALARY”>19260.25</Column>
    <Column name=”COMM”></Column>
     </Row>
    </RowSet></XMLBlock>
```

Figure 5. Result set of XML data generated by Net.Data

Another way to generate XML documents with automated tags is to use the stored procedures, dxxGenXML() and dxxRetrieveXML(), provided by IBM DB2 XML Extender. You need to define the mapping of DTD against relational tables (RDB_nodes) or queries (SQL_stmt) using DAD. Dynamic queries are supported by taking two optional parameters, *override* and *overrideType*. Based on the input *overrideType*, the application can override the <SQL_stmt> tag values for SQL mapping or the conditions in RDB_nodes for relational tables mapping in the DAD. dxxGenXML() and dxxRetrieveXML() provide the same function except dxxRetrieveXML is optimized for repeated tasks because the DAD is saved internally.

Conclusion

IBM DB2 XML Extender and DB2 Universal Database Net.Data provide an end-to-end solution for storing and retrieving XML documents for business-to-business and business-to-consumer (with a browser) processing. In particular, it is shown that an XML document can be stored as an XML column or decomposed into multiple DB2 tables and columns. In addition, XML documents can be generated from existing DB2 data. For those interested in experimenting with the software, IBM DB2 XML Extender and Net.Data can be downloaded at IBM Web sites.

For more information

To learn more about IBM developerWorks, visit:
ibm.com/developerWorks

To learn more about IBM DB2 XML Extender, visit:
ibm.com/software/data/db2/extenders/xmlext

To learn more about IBM DB2 Universal Database Net.Data, visit:
ibm.com/software/data/net.data

To learn more about IBM alphaWorks XML Parser for Java, visit:
ibm.com/alphaworks

♲

Printed in the United States on recycled paper containing 10% recovered post-consumer fiber.