

XML and DB2

Josephine Cheng , Jane Xu
IBM Santa Teresa Laboratory
chengjm@us.ibm.com, jxu@us.ibm.com

Abstract

The eXtensible Markup Language (XML) is a key technology that facilitates both information exchange and e-business transactions. Starting with DB2 UDB Net.Data V1, an application can generate XML documents from SQL queries against DB2 or any ODBC compliant databases. Today DB2 UDB XML Extender not only serves as a repository for both XML documents and their Document Type Definitions (DTDs), but also provides data management functionalities such as data integrity, security, recoverability and manageability. User has the option to store the entire document as an XML user-defined column or to decompose the document into multiple tables and columns. Fast search via indices is provided for both 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. User can retrieve the entire document or extract XML elements and attributes dynamically in an SQL query. In addition, XML Extender provides stored procedure to generate XML documents from existing data. Together with Net.Data, one can browse the content of the XML documents via the Internet.

1. 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 meta-language for defining other markup languages specialized to the needs of an industry (e.g. RELML, HL7) or discipline (e.g. WML,

VoiceML). The meta data is defined in Document Type Definition (DTD), or XML grammars, which specifies a set of tags and the rules connecting the tags and its contents. There is a set of standards that is related to XML. Some of the better known ones are Document Object Model (DOM), Simple API for XML (SAX), xXtensible Stylesheet Language (XSL), XML Linking Language (XLink), XML Pointer Language (XPointer), XML Query Language (XQL), Extensive Stylesheet Language Transformation (XSLT), XML Path Language (XPath) etc. An XML tutorial can be found at the IBM DeveloperWork site [1].

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 more semantic to the web content page. Today's search engine is mostly based on words appearing in the content page rather than 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 as "product", "related-product", etc. 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 via XSL, in a document. As a result, the same data can be presented in different formats according to such criteria as output device type or business purpose. Finally, the availability of an XML Parser as a freeware [2] promotes the popularity of XML usage. With the increase in popularity in the use of XML pages, one needs a repository to manage these documents and provide a fast search capability. DB2 UDB XML Extender is designed to provide these capabilities. It is available for web download at [3].

In today's just-in-time business paradigm, applications from suppliers, vendors and partners need to communicate with each other efficiently. 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 specification 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 amongst these heterogeneous systems. For new applications, one 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.

e-business on the Internet is very popular and the browser has become a universal client. Recently released browsers such as Internet Explorer V5 (IE5) support the presentation of XML documents with XSL stylesheets. One can invoke DB2 UDB XML Extender in a data access Java Bean via Websphere or in a Net.Data macro returning XML documents to IE5. One can also generate an XML document with queries against DB2 or any ODBC compliant database using Net.Data.

2. XML documents and DTDs repository

XML Extender serves as a repository for both XML documents as well as their DTDs. There are 2 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 four XML user-defined types (UDTs): XMLCLOB, XMLVARCHAR, XMLDBCLOB, and XMLFile. XMLCLOB, XMLVARCHAR and XMLDBCLOB store an XML document as a CLOB, VARCHAR, double byte CLOB 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 for insert, select (either an entire document or a fragment) and update operations. Figure 1 shows an overview of an XML document stored as an XML Column.

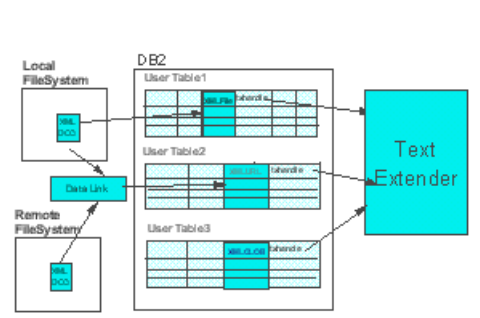


Figure 1. XML Column Overview

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 shows an overview of an XML document stored as an XML Collection.

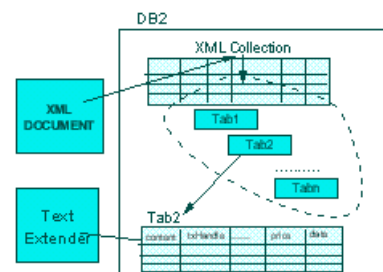


Figure 2. XML Collection Overview

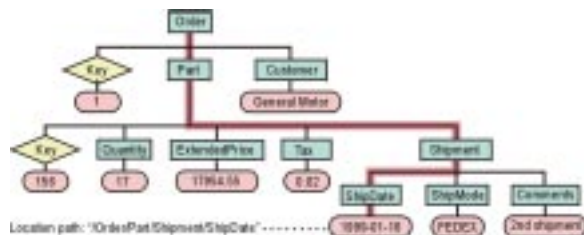
XML Column provides a simple way to manage XML documents while providing fast search and indexing capabilities. It is ideal for data, such as purchase which do not require frequent updates. Performance is also faster if the retrieval of an entire document is often desirable as there is no need to compose 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 set of chapters and each chapter contains 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.

3. 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. Figure 3 shows an example using SALES_TAB, a purchase order table with ORDER as an XML document and INVOICE_NUMBER as the primary key. The DAD defines PRICE element in the ORDER document for indexing and the side table is named PART_SIDETBL.

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

Figure. 3 XML Column Example



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

```
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
WHERE extractVarchar(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 so 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 (e.g. /Order/Customer) with its UDF db2tx.contains.

For XML document decomposed into XML Collections, 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 search criteria which can be based either on tables or SQL query. Furthermore the search criteria can be overridden dynamically at the invocation of the stored procedure as an optional parameters. Examples can be found in XML Extender Administration and Programming Guide [3].

4. 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 a 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)">$(V_COMM)</Column>
        </Row>
    %}
</Rowset>
%}
%}
%HTML(REPORT){
Content-Type: text/xml
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl"
    href="http://malacandra.stl.ibm.com:80
    72/$(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 Figure 4 example. Furthermore, 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” allowing users to specify the SQL statement or the stored procedure invocation directly in a url [4].

```

Content-Type: text/xml
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl"
    href="http://malacandra.stl.ibm.com:8072/row
    setTable.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>
    </Row>
</RowSet>

```

```

<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. Resultset of XML data generated by Net.Data

Another way to generate XML document with automated tags is to use the stored procedures, `dxxGenXML()` and `dxxRetrieveXML()`, provided by 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. Both `dxxGenXML()` and `dxxRetrieveXML()` provides the same function except `dxxRetrieveXML()` is optimized for repeated tasks because the DAD is saved internally.

5. Conclusion

An end-to-end solution for storing and retrieving XML documents for both business-to-business and business-to-consumer (via browser) processing using DB2 UDB XML Extender and DB2 UDB Net.Data has been described in this paper. In particular, it has been shown that an XML document can be stored as an XML column or decomposed into multiple DB2 tables and columns. In addition, XML document can be generated from existing DB2 data. Readers interested in experimenting with XML Extender and Net.Data can be downloaded at [3], [4], respectively.

References

- [1] IBM XML DeveloperWork at <http://www.ibm.com/developer/>
- [2] IBM Alphawork XML Parser at <http://www.alphaworks.ibm.com>.
- [3] DB2 UDB XML Extender web site at

<http://www-4.ibm.com/software/data/db2/extenders/xmlext>

[4]DB2 UDB Net.Data web site at

<http://www-4.ibm.com/software/data/net.data>.

[5] “Index design for structured documents based on abstraction”, Josephine Cheng, Jyh-Herng Chow, Jane Xu, 6th International Conference on Database Systems for Advanced Applications, Hsinchu, Taiwan on April 19-21, 1999.

[6] “Pluggng into XML”, Harold Treat, DB2 Magazine, 4Qtr, 1999.