

Comparing and contrasting XML features of DB2 and COBOL - When to use which?

Tom Ross
IBM

August 5, 2010

Session 7978



SHARE in Boston

Agenda

- What do you want to use XML for?
- Terminology: Do you speak COBOL or DB2?
- How to create XML documents
- How to get data out of XML documents into variables
- How to validate XML documents
- How to store XML documents
- When to use which XML features?

What do you want to do with XML?



- Communication between programs/applications?
 - Convert existing applications into Web Services?
 - Or maybe applications as services, even without Web access!
 - XML provides a very flexible interface compared to EDIFACT
- Archive information?
 - Save XML documents as is: DB2 BLOB or CLOB
 - Save XML documents in XML columns
 - Allows many DB2 relational features on XML data
 - Save scalar/relational data contained within the XML document
 - Allows all DB2 relational capabilities on data
- All of the above?
 - EX: Use XML as communication for Web Service requests
 - Use XML for subsequent invocation of related applications
 - Save XML in DB2 as record of transactions

Do you speak COBOL or DB2?

- DB2 XML support is called PureXML
- Create an XML document
 - COBOL: generate
 - XML GENERATE verb
 - DB2: publish and serialize
 - SQL/XML publishing functions (XMLDOCUMENT, etc) along with XMLSERIALIZE function to convert from DB2 internal representation into serialized string format
- Access data within XML documents
 - COBOL: XML PARSE verb
 - Data is scalar element and attribute values and tags
 - DB2: XMLPARSE SQL function
 - During *XML parsing*, the string representation of an XML document is transformed into an instance of the XPath model
 - After parsing the data is stored in XML columns or
 - Could do XMLTABLE or XMLQUERY directly

Do you speak COBOL or DB2?

- XML SCHEMA
 - COBOL: XML schema
 - Transformed into OSR (Optimized Schema Representation) and then
 - Can be used by XML System Services APIs or by COBOL XML PARSE statement
 - *The OSR can be in a file or variable for COBOL XML PARSE*
 - DB2: XML schema object
 - Must be registered in XML schema repository(XSR), and
 - Can only be used by DB2
- Validation of XML document against XML SCHEMA
 - COBOL: XML PARSE verb VALIDATING WITH clause
 - DB2: XMLPARSE function with SYSFUN.DSN_XMLVALIDATE function as argument
 - DB2 V10 will have SYSIBM.DSN_XMLVALIDATE built-in function

How to create XML documents

- Using PureXML
 - Use a string
 - Use SQL/XML publishing functions XMLDOCUMENT, XMLEMENT, etc,
 - Use XMLSERIALIZE
- Using COBOL
 - Use XML GENERATE simple
 - Use XML GENERATE with some fancy features

How to create XML documents

- Using PureXML with strings
 - Create a table named MYCUSTOMER that contains an XML column:
CREATE TABLE MYCUSTOMER (CID BIGINT, INFO XML)#
 - Create an index over XML data:
**CREATE UNIQUE INDEX MYCUT_CID_XMLIDX ON MYCUSTOMER(INFO)
GENERATE KEY USING XMLPATTERN 'declare default element namespace
"http://posample.org"; /customer_info/@Cid' AS SQL DECFLOAT#**
 - Insert XML document into that table:
**INSERT INTO MYCUSTOMER (CID, INFO) VALUES (1000,
'<customer_info xmlns="http://posample.org" Cid="1000">
 <name>Kathy Smith</name>
 <addr country="Canada">
 <street>5 Rosewood</street>
 <city>Toronto</city>
 <prov-state>Ontario</prov-state>
 <pcode-zip>M6W 1E6</pcode-zip>
 </addr>
 <phone type="work">416-555-1358</phone>
</customer_info>')#**

How to create XML documents

- Using PureXML with SQL/XML publishing functions
 - Create a table named EMPLOYEE that contains XML columns:

```
CREATE TABLE EMPLOYEE  
( EMP_ID          CHAR(4)          NOT NULL,  
  EMP_NAME        VARCHAR(20) NOT NULL,  
  EMP_CONTACT    XML              NOT NULL,  
  EMP_HOBBY       XML,  
  PRIMARY KEY (EMP_ID) );
```

- Create an index over XML data:
CREATE UNIQUE INDEX XEMPLID ON EMPLOYEE(EMP_ID ASC);

How to create XML documents

- Using PureXML with SQL/XML publishing functions

```
EXEC SQL INSERT INTO EMPLOYEE VALUES
('0123', 'MAPLE',
 (SELECT XMLDOCUMENT
  (XMLELEMENT
   (NAME "contact",
    XMLATTRIBUTES('0123' AS "id"),
    XMLCOMMENT('This is just a simple example'),
    XMLFOREST(c.CON_ADDRESS AS "address",
              c.CON_PHONE AS "phone",
              c.CON_EMAIL AS "email")
   )
  ) FROM CONTACT c WHERE c.CON_ID = '0123'
 ),
 (SELECT XMLDOCUMENT
  (XMLELEMENT
   (NAME "hobby",
    XMLFOREST('blue' AS "music",
              'football' AS "sports")
   )
  )
 ) ) ) FROM SYSIBM.SYSDUMMY1
END-EXEC.
```

How to create XML documents

- Some SQL/XML publishing functions:
 - **XMLDOCUMENT scalar function**
 - Returns an XML value with a single document node with zero or more children nodes
 - **XMLELEMENT scalar function**
 - Returns an XML value that is an XML element node
 - **XMLATTRIBUTES scalar function**
 - Constructs XML attributes from the arguments.
 - This function can only be used as an argument of the XMLELEMENT function
 - **XMLCOMMENT scalar function**
 - Returns an XML value with a single comment node with the input argument as the content
 - **XMLFOREST scalar function**
 - Returns an XML value that is a sequence of XML element nodes

How to create XML documents

- XML serialization
 - is the process of converting XML data from internal representation in a DB2 table to the serialized string format that it has in an application

Explicit:

```
SELECT XMLSERIALIZE (Info as BLOB(1M)  
      INCLUDING XMLDECLARATION  
      FROM Customer WHERE CID='1000'
```

Results in UTF-8 declaration:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

- when you use INCLUDING XMLDECLARATION, you need to ensure that the retrieved data is not converted from UTF-8 encoding to another encoding

How to create XML documents

- XML serialization
 - is the process of converting XML data from internal representation in a DB2 table to the serialized string format that it has in an application

Implicit:

```
EXEC SQL BEGIN DECLARE SECTION;  
SQL TYPE IS XML AS BLOB (1M) xmlCustInfo;  
EXEC SQL END DECLARE SECTION;  
  
...  
EXEC SQL SELECT INFO INTO :xmlCustInfo  
FROM Customer WHERE Cid='1000';
```

Results in declaration with code page of application:

```
<?xml version="1.0" encoding="IBM 1140" ?>
```

How to create XML documents

- Using COBOL verb XML GENERATE - simple case

```
01 customer_info.  
  05 name          Pic X(15) Value 'Kathy Smith'  
  05 addr.  
    10 country      Pic X(10) Value "Canada"  
    10 street       Pic X(10) Value '5 Rosewood'  
    10 city         Pic X(10) Value 'Toronto'  
    10 prov-state   Pic X(10) Value 'Ontario'  
    10 pcode-zip    Pic X(10) Value 'M6W 1E6'  
  05 phone         Pic X(15) Value '416-555-1358'  
  
77 xmldoc          Pic X(250).
```

```
XML GENERATE xmldoc FROM customer_info
```

- Note: COBOL GENERATE can have all attribute values or all element values

How to create XML documents



- Using COBOL verb XML GENERATE - simple case output

```
<customer_info>
  <name>Kathy Smith</name>
  <addr>
    <country>Canada</country>
    <street>5 Rosewood</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <PCODE-ZIP>M6W 1E6</PCODE-ZIP>
  </addr>
  <phone>416-555-1358</phone>
</customer_info>
```

How to create XML documents

- Using COBOL verb XML GENERATE - more features

```
01 customer_info.
```

```
    05 name                Pic X(15) Value 'Kathy Smith'.
```

```
    05 addr.
```

```
        10 country         Pic X(10) Value "Canada".
```

```
        10 street          Pic X(10) Value '5 Rosewood'.
```

```
        10 city            Pic X(10) Value 'Toronto'.
```

```
        10 prov-state      Pic X(10) Value 'Ontario'.
```

```
        10 pcode-zip       Pic X(10) Value 'M6W 1E6'.
```

```
    05 phone              Pic X(15) Value '416-555-1358'.
```

```
77 ns-var                Pic X(30) Value 'http://posample.org'.
```

```
77 xmldoc                Pic X(250).
```

```
XML GENERATE xmldoc FROM customer_info
```

```
    WITH XML-DECLARATION
```

```
    WITH ATTRIBUTES NAMESPACE IS ns-var
```

```
END-XML
```

How to create XML documents



- Using COBOL verb XML GENERATE - more features output

```
<?xml version="1.0" encoding="IBM-1140"?>
<customer_info xmlns="http://posample.org"
  name="Kathy Smith"
  phone="416-555-1358">
  <addr
    country="Canada"
    street="5 Rosewood"
    city="Toronto"
    prov-state="Ontario"
    pcode-zip="M6W 1E6">
  </addr>
</customer_info>
```

- Note: this example uses attribute values, no element values

How to get data out of XML documents into variables



- Using DB2 PureXML
 - XMLTABLE SQL table function with PATH option
 - Example document:

```
<customer_info xmlns="http://posample.org" Cid="1000">
  <name>Kathy Smith</name>
  <addr country="Canada">
    <street>5 Rosewood</street>
    <city>Toronto</city>
    <prov-state>Ontario</prov-state>
    <pcode-zip>M6W 1E6</pcode-zip>
  </addr>
  <phone type="work">416-555-1358</phone>
</customer_info>')
```

How to get data out of XML documents into variables



- Using DB2 PureXML
 - XMLTABLE SQL table function with PATH option
 - Example table definition:

```
CREATE TABLE CUSTADDR  
  (CUSTNAME      VARCHAR(30),  
   CUSTCOUNTRY   VARCHAR(30  
   CUSTSTREET    VARCHAR(30),  
   CUSTCITY      VARCHAR(30)  
   CUSTSTATE     VARCHAR(30),  
   CUSTZIP       VARCHAR(30))
```

How to get data out of XML documents into variables



- Using DB2 PureXML
 - Example: Get data out of XML doc stored in XML table CUSTOMER and store it in relational table CUSTADDR as XML data
 - Note: use @ for attribute values:

```
INSERT INTO CUSTADDR
SELECT X.*
FROM CUSTOMER,
XMLTABLE (XMLNAMESPACES(DEFAULT 'http://posample.org'),
'//customer_info'
PASSING CUSTOMER.INFO
COLUMNS
"CUSTNAME"      VARCHAR(30) PATH 'name',
"CUSTCOUNTRY"   VARCHAR(30) PATH 'addr/@country',
"CUSTSTREET"    VARCHAR(30) PATH 'addr/street',
"CUSTCITY"      VARCHAR(30) PATH 'addr/city',
"CUSTSTATE"     VARCHAR(30) PATH 'addr/prov-state',
"CUSTZIP"       VARCHAR(30) PATH 'addr/pcode-zip'
) as X
```

How to get data out of XML documents into variables - COBOL



- Using COBOL XML PARSE
 - PARSING PROCEDURE of XML PARSE statement
 - Example: Get data out of XML doc stored in COBOL variable CUSTOMER and store it in COBOL group CUSTADR as scalar data
 - Use ATTRIBUTE-CHARACTERS to attributes:

```
XML PARSE CUSTOMER
  PROCESSING PROCEDURE GET-DATA
END-XML
```

How to get data out of XML documents into variables - COBOL



GET-DATA.

EVALUATE XML-EVENT

When 'START-OF-ELEMENT'

Display 'Start element tag: {' XML-Text '}'

Move XML-Text to current-element

When 'CONTENT-CHARACTERS'

Display 'Content characters: {' XML-Text '}'

* ==> Transform XML content to operational COBOL data item..

EVALUATE current-element

When 'street'

Move XML-TEXT TO street

When 'city'

Move XML-TEXT TO city

When 'prov-state'

Move XML-TEXT TO prov-state

When 'pcode-zip'

Move XML-TEXT TO pcode-zip

When Other

Continue

End-evaluate

How to get data out of XML documents into variables - COBOL



GET-DATA.

EVALUATE XML-EVENT

When 'ATTRIBUTE-NAME'

Display 'Attribute name: {' XML-Text '}'

Move XML-Text to current-attribute

When 'ATTRIBUTE-CHARACTERS'

Display 'Attribute characters: {' XML-Text '}'

*** ==> Transform XML content to operational COBOL data item...**

EVALUATE current-attribute

When 'country'

Move XML-TEXT TO country

When Other

Continue

End-evaluate

How to validate XML documents

- Using DB2 PureXML
 - XMLPARSE function with SYSFUN.DSN_XMLVALIDATE function as argument

Insert into MyProduct

```
(pid, name, Price, PromoPrice,  
  PromoStart, PromoEnd,  
  description)  
values ( '110-100-01', 'Anvil', 9.99, 7.99,  
        '11-02-2004', '12-02-2004',  
        XMLPARSE(DOCUMENT  
                  SYSFUN.DSN_XMLVALIDATE(CAST ? AS CLOB),  
                  'SYSXSR.PRODUCT' )))
```

How to validate XML documents



- Using COBOL XML PARSE
 - VALIDATING WITH phrase of XML PARSE statement

```
//GO.DDSCHEMA DD  
DSN=TMROSS.XML.SCHEMA(MYSCHEMA), DISP=SHR
```

```
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SPECIAL-NAMES.
```

```
XML-SCHEMA my_Schema IS 'DDSCHEMA'.
```

```
XML PARSE CUSTOMER  
PROCESSING PROCEDURE GET-DATA  
VALIDATING WITH FILE my_Schema  
END-XML
```

How to store XML documents

- Use DB2 to store XML documents in XML form
 - Use SQL INSERT without PureXML into BLOB or CLOB
 - Use SQL INSERT into XML column (PureXML)
 - Use SQL INSERT with XMLPARSE to store into multiple XML columns (PureXML)
- Use COBOL XML features
 - Store as BLOB or CLOB without PureXML
 - Store data from XML documents in relational form without PureXML
 - Use XML PARSE to get XML data into host variables
 - Use data to store in relational form and to generate XML documents

How to store XML documents - DB2

- XML documents LOOK like they are just long strings of character data, this is not how DB2 stores them
 - You cannot therefore compare a CHAR column with an XML one
 - DB2 stores XML documents in a very similar way to LOBs
 - You don't have to create all the auxiliary-type objects
- When you create a DB2 table that contains an XML data type column, you also get (or you need to create yourself)
- A hidden column called DB2_GENERATED_DOC_ID_FOR_XML – this is like the LOB ROWID column
 - A unique index on the DOCID column
 - A table space to store the XML document(s)
 - A table in this XML table space
 - And an index on the XML table

When to use which XML features?

- Use DB2 to store XML documents in XML columns for later retrieval as XML
- Use DB2 or COBOL to get data out of XML documents
 - Use XML PARSE in COBOL to process the data in mid parse
 - Use DB2 (or COBOL + DB2) features to store data as relational data
- Use COBOL or DB2 to create XML documents to use as Web Service responses
 - COBOL is easier, DB2 more flexible (attributes vs. element values, for example)
 - COBOL XML support designed for SOA, DB2 for archiving, retrieval, and data analysis

When to use which XML features?

- Programs that use relational data from DB2 in host variables.
Using DB2 tables without XML columns:
 - Could use COBOL to parse incoming XML documents into DB2 host variables for insert into DB2
 - Could use COBOL to generate XML documents from retrieved DB2 relational data in host variables to send as responses to client requests for data
- Programs that retrieve relational data from DB2 in host variables. Using DB2 tables with XML columns:
 - Could change to use PureXML to parse incoming XML documents into DB2 XML columns
 - Could use PureXML features to create XML documents to send as responses to client requests for data

When to use which XML features?

- New web service code
 - Could use PureXML to save XML service requests in XML columns in DB2
 - Could write programs to retrieve XML data to satisfy requests for archived information
 - Could also use PureXML to analyze XML data for business patterns
 - One customer used PureXML to store data for transactions in a way that allowed easy populating of Web page
- If you need to use COBOL XML PARSE to get the data out of XML documents anyway, it might make sense to store data in relational DB2 tables
 - You can do more data analysis and
 - Performance of queries would be better

When to use which XML features?

- If you need to store and retrieve the exact original XML document
 - Do not store in XML column, store as BLOB or CLOB
 - In COBOL, you can recreate XML with the same values from data
 - In DB2, retrieval from an XML column will retrieve an equivalent reproduction

When to use which XML features?

- For cases where the XML documents might be malformed, COBOL can give better information about what is wrong
 - One customer used XML PARSE in COBOL to find out why DB2 would not accept their XML document. Each problem gives a return code that the COBOL program can use. DB2 normally just gives 'bad document' SQL code.
- In COBOL you can stop the parse partway through
MOVE -1 To XML-EVENT
- Both DB2 and COBOL use the XML System Services parsers
 - Validating and non-validating
 - COBOL also has a 3rd parser, high speed minimum function