The Ins and Outs of XML and DB2 for i5/OS

David Andruchuk
Sr. Architect
Computer Systems Design Associates, Inc.
August 20, 2008

What can *i* do…..*i* can do XML
The Ins and Outs of XML and DB2 for i5/OS

Table of Contents

• Part 1. Introduction
  Chapter 1. Introduction to XML integration with DB2 for i5/OS
  Chapter 2. Scenario overview

• Part 2. Programmatic Approach
  Chapter 3. Using SQL to compose XML
  Chapter 4. Using XSL Transformation and SQL
  Chapter 5. Using RPG for XML processing
  Chapter 6. Using SAX and Java to decompose XML
  Chapter 7. Advantages and Disadvantages of the Programmatic approach

• Part 3. Middleware Approach
  Chapter 8. DB2 XML Extender Overview
  Chapter 9. Performance related information regarding XML Extender
  Chapter 10. Shredding Methodology
  Chapter 11. Composing Methodology
  Chapter 12. Advantages and Disadvantages of the Middleware approach

• Part 4. Moving Forward
  Chapter 13. A look into the future
  Appendix A. Additional material
The Ins and Outs of XML and DB2 for i5/OS

From the Redbook:

- Extensible Markup Language (XML) represents a fundamental change in computing. It allows applications to move away from proprietary file and data formats to a world of open data interchange. XML has become ubiquitous not only because of its range of applications, but also because of its ease of use.

- Although XML solves many problems by providing a standard format for data interchange, some challenges remain. In the real world, applications need reliable services to store, retrieve, and manipulate data. These services have traditionally been offered by DB2® for i5/OS®.

- This Redbook deals with the challenges of representing XML hierarchies in the relational database model. It will provide an in-depth explanation of the three most popular approaches to bridge the hierarchy - relational model dichotomy:

  - Programmatically process the XML documents and map their hierarchy into a relational database.
  - Use database middleware to handle the XML parsing and XML-to-relational mapping.
  - Use XSL transformation to transform inbound XML documents directly to SQL scripts.
The Ins and Outs of XML and DB2 for i5/OS

What you need to get started:

- **XML Editing tool(s)**
  - WDSC (IBM)
  - XMLSpy (Altova)
  - Stylus Studio (Progress Software Corporation)

- **XML Reference(s)**
  - WDSC Step by Step (MC Press)
  - XML for eServer i5 and iSeries (MC Press)
  - Qshell for iSeries (MC Press)
  - XSLT 2nd Edition (WROX)
  - W3Schools Online Web Tutorials (www.w3schools.org)
What is XML?
- XML is short for Extensible Markup Language

Pros of XML?
- XML is flexible
- XML is self documenting
- XML can replace or extend legacy systems

Cons of XML?
- XML is verbose
- XML can significantly increases the size of your data
- XML works best with hierarchal data in a relational data base
There are three main components of XML

• XML Document
  – Document containing data

• XML Document Type Definition (DTD)
  – Document that defines the document structure of the XML Document

• XML Schema Definition (XSD)
  – Document that defines the document structure of the XML Document in an XML-based alternative to DTD
Structure of XML documents

• An XML document has a tree-like structure that is hierarchical
• The document must contain one and only one root element
• An element is the parent of all the elements it contains
• The elements that are inside a parent element are called its children
• Similarly, the elements that have the same parent element are called siblings
The Ins and Outs of XML and DB2 for i5/OS
The Ins and Outs of XML and DB2 for i5/OS
The Ins and Outs of XML and DB2 for i5/OS
The Ins and Outs of XML and DB2 for i5/OS

Structure of DTD documents

- Contains the list of tags which are allowed within the XML document and their types and attributes
- Defines how elements relate to one another within the document's tree structure and specifies which attributes may be used with which elements
The Ins and Outs of XML and DB2 for i5/OS
The Ins and Outs of XML and DB2 for i5/OS
Structure of XSD documents

- Contains the legal building blocks of an XML document similar to DTD
- Defines how elements relate to one another within the document's tree structure and specifies which attributes may be used with which elements
- Schema supports all data types used in most programming languages such as string, decimal, integer, Boolean, date and time
- Supports complexType elements that allows you to define an element type that can consist of sub-elements
The Ins and Outs of XML and DB2 for i5/OS
The Ins and Outs of XML and DB2 for i5/OS
The Ins and Outs of XML and DB2 for i5/OS

Key Points to Remember

XML is Hierarchical
- Hierarchical data is stored where the order and relationship of the elements is significant
- Elements are not related to one another by any key structure or relationship

DB2 is Relational
- Relational data is stored in rows of two dimensional tables where the physical order is insignificant
- Tables are related to one another by key
The Ins and Outs of XML and DB2 for i5/OS

Questions?
The Ins and Outs of XML and DB2 for i5/OS

Let’s explore 4 XML programmatic processing methods

SQL compose of an XML document
   – Compose a StoreSales XML document from 3 tables

RPG compose of an XML document
   – Compose a StoreSales XML document from 3 tables using SQLRPGLE & CGI

XSL & XSLT transform of an XML document to SQL script
   – Transform a composed StoreSales XML document to SQL script to populate 1 table

RPG decompose of an XML document
   – Use V5R4 RPG to decompose a CorpSales XML document to populate 1 table
The Ins and Outs of XML and DB2 for i5/OS

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StoreSales date="2006-04-06" xmlns:xsi="http://www3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="StoreSales.xsd">
  <StoreId>7</StoreId>
  <Transactions>
    <Transaction type="SALE">
      <SalesItem>
        <Brand name="Pepsi"/>
        <Name>Mt. Dew 20 oz.</Name>
        <Currency>USD</Currency>
        <Amount>1.19</Amount>
      </SalesItem>
    </Transaction>
  </Transactions>
</StoreSales>
```
The Ins and Outs of XML and DB2 for i5/OS

SQL Compose of an XML document
Coding the SQL stored procedure

create procedure GenStoreXML
  (IN dateval Char(10), IN storeval varchar(10))
language SQL

begin
declare string char(30000);
declare transid INTEGER;
declare libval varchar(50);
declare liblen integer;

set libval = 'Store' || storeval;
set liblen = (length(libval) + 18);
set string = 'call qsys/qemdxec("CHGCURLIB CURLIB(' || libval || ')", 00000000' || cast(liblen as decimal(15,5)) || ')';
prepare s1 from string;
execute s1;

create table qtemp/outfile(char1 char(1000));
insert into qtemp/outfile values('<?xml version="1.0" encoding="UTF-8"?>');

set string = '<StoreSales date="' || dateval || '" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="StoreSales.xsd">';

insert into qtemp/outfile values(string);

set string = '<storeId>' || storeval || '</storeId>'; 
insert into qtemp/outfile values(string);
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQL stored procedure (continued)

```sql
insert into qtemp/outfile values('<Transactions>');

FOR each_transaction AS cursor1 CURSOR FOR
select TransactionId, Type from Transactions
where dateval = char(date(TransactionTime), ISO)

DO
    set string = '<Transaction type="' || Type || '"';
    insert into qtemp/outfile values(string);

    set transid = TransactionId;

FOR each_salesitem AS cursor2 CURSOR FOR
select ItemName, BrandName, Currency, Amount from SalesItem
where TransactionID = transid

    DO
        set string = '<SalesItem><Brand name="' || replace(BrandName, '&', '&amp;') || '"/' || Name' || ItemName || '</Name><Currency>' || Currency || '</Currency><Amount>' || rtrim(char(Amount)) || '</Amount></SalesItem>';
        insert into qtemp/outfile values(string);

    END FOR;

    insert into qtemp/outfile values('</Transaction>');</

END FOR;

insert into qtemp/outfile values('</Transactions>');
The Ins and Outs of XML and DB2 for i5/OS

**Coding the SQL stored procedure** (continued)

```sql
insert into qtemp/outfile values('</StoreSales>');

set liblen = (length(libval) + 147);

set string = 'call qsys/qcmdexc("CPYTOIMPF FROMFILE(QTEMP/OUTFILE) TOSTMF(""/XMLRedbook/StoreXML/" || libval || substring(dateval,9,2) || ".xml") MBROPT(*REPLACE) STMFCODPAG(819) RCDDLM(*CRLF) DTAFMT(*FIXED) STRDLM(*NONE)", 0000000' || cast(liblen as decimal(15,5)) || ')';

prepare s2 from string;

execute s2;

drop table qtemp/outfile;

end;
```
Some considerations for SQL Composition

- SQL Stored Procedures contain two parts: a catalog entry and a program object. These two items can exist separate of each other but only work when both parts are there at run time and are not as easy to implement as using a SAVOBJ & RSTOBJ.

Work around for this limitation:
- Run the SQL Create Procedure statement on each box they are to be used on to create the catalog entry and program object
The Ins and Outs of XML and DB2 for i5/OS

RPG compose of an XML document
The Ins and Outs of XML and DB2 for i5/OS

Using RPG to compose XML

The ILE RPG Compiler as of the V5R4 release of i5/OS has no native operations codes and built in functions that facilitate composition of an XML document.

IBM through the Client Technology Center (CTC) has provided an open source solution for Web development, via an HLL, based on the Common Gateway Interface (CGI) language. While the main focus of the CGI functionality is to provide Web access to an HLL, in this example we have used the CGI toolkit to produce an XML document using RPG.

The biggest advantage of using the CGI toolkit is that the complexity of building an HTML interface, or in our case, an XML document and publishing it is removed. One service program supplied in the toolkit does all the work for us and provides simple procedure interfaces to access the features supplied by CGI. The developer can just concentrate on business logic required to generate an XML document.
<?xml version="1.0" encoding="UTF-8"?>
<StoreSales date="2006-04-06" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="StoreSales.xsd">
  <StoreId>7</StoreId>
  <Transactions>
    <Transaction type="SALE">
      <SalesItem>
        <Brand name="Pepsi"/>
        <Name>Ma. Dew 20 oz.</Name>
        <Currency>USD</Currency>
        <Amount>1.19</Amount>
      </SalesItem>
    </Transaction>
  </Transactions>
</StoreSales>
The Ins and Outs of XML and DB2 for i5/OS

CMD Interface

CMD     PROMPT('StoreSales XML Compose')

PARM    KWD(INPUT) TYPE(INPUT) MIN(1) PROMPT('Input')

INPUT:  ELEM     TYPE(*NAME) MIN(1) PROMPT('Schema')
        ELEM     TYPE(*CHAR) LEN(10) MIN(1) +
                  CHOICE('YYYY-MM-DD') PROMPT('Sales Date')
        ELEM     TYPE(*CHAR) LEN(10) MIN(1) PROMPT('Store Id')

PARM    KWD(OUTPUT) TYPE(OUTPUT) MIN(1) +
        PROMPT('Output')

OUTPUT: ELEM     TYPE(*CHAR) LEN(80) MIN(1) CASE(*MIXED) +
        PROMPT('Directory')
        ELEM     TYPE(*CHAR) LEN(80) MIN(1) CASE(*MIXED) +
                  PROMPT('Document')

        KWD(CODEPAGE) TYPE(*DEC) LEN(5) DFT(819) +
                  RANGE(1 32767) PMTCtrl(*PMTRQS) +
                  PROMPT('Code page')

        KWD(DEBUG) TYPE(*CHAR) LEN(4) RSTD(*YES) +
                  DFT(*NO) VALUES(*NO *YES) PMTCtrl(*PMTRQS) +
                  PROMPT('Debug')
The Ins and Outs of XML and DB2 for i5/OS

Coding the CGI XML template

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

<StoreSales date="/\%salesdate\%/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="StoreSales.xsd">
  <StoreId="/\%storeId\%/" />
  <Transactions>
    <Transaction type="/\%type\%/" />
    <SalesItem>
      <Brand name="/\%brand\%/" />
      <Name="/\%name\%/" />
      <Currency="/\%currency\%/" />
      <Amount="/\%amount\%/" />
    </SalesItem>
    <EndTransaction />
  </Transaction>
</StoreSales>
```
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program

// STRSLSGENR CMD parms prototype
D STRSLSGENR pr extpgm('STRSLSGENR')
D Input 33 Schema/Date/Store
D Output 163 Directory/Document
D CodePage 5 0 Codepage
D Debug 4 Debug function

// STRSLSGENR CMD parms procedure interface
D STRSLSGENR pi
D Input 33 Schema/Date/Store
D Output 163 Directory/Document
D CodePage 5 0 Codepage
D Debug 4 Debug function

// CGI Prototypes
/Copy cgidev2/qrpglesrc,PrototypeB CGI prototypes

// Error data structure
/Copy cgidev2/qrpglesrc,usec CGI error codes

// Execute SQL prototype
D ExecuteSql PR 10i 0 Execute SQL procedure
D 5000 value
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program (continued)

// Data structure for STORESALES cursor fields

Dv1_ds    ds        dim(1000) qualified
Dtransid   9b 0
Dtranstime  z
Dtranstype  30  varying
Dstrtransid  9b 0
Dsalesitemid  9b 0
Ditemname   80  varying
Dbrandname  30  varying
Dcurrency   30  varying
Damount     9p 2

Dv1_null_ds  ds        dim(1000) qualified
D v1_null_value  5i 0 dim(4)
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program (continued)

    // Set SQL table parameter(s)
    schema = %subst(input:3:10);
    date = %subst(input:13:10);
    store = %subst(input:23:10);

    // Set IFS parameter(s)
    ifs_folder = %subst(output:3:80);
    ifs_file = %subst(output:83:80);
    ifs_code_page = CodePage;

    // Check if user included ending / in folder
    //, If not included, insert when building IFS document location
    var_len = %len(ifs_folder);

    if %subst(ifs_folder:var_len:1) <> '/';
       ifs_document = %trim(ifs_folder) + '/' + %trim(ifs_file);
    else;
       ifs_document = %trim(ifs_folder) + %trim(ifs_file);
    endif;
Coding the SQLRPGLE program (continued)

    // Clear HTML Buffer
    ClrHtmlBuffer();

    // Load external XML template to memory buffer
    CallP GetHtml('QXMLSRC':'XMLREDBOOK':'STRSLSGENR');

    // Set SQL statement to drop previous alias over input schema/table and execute
    SqlStm = 'drop alias qtemp/TRANS_alias';
    eval sqlcod = ExecuteSql(SqlStm);

    SqlStm = 'drop alias qtemp/SALES_alias';
    eval sqlcod = ExecuteSql(SqlStm);

    // Set SQL statement to create alias over input tables and execute
    SqlStm = 'create alias qtemp/TRANS_alias for ' +
             %trim(schema) + '/TRANS00001';
    eval sqlcod = ExecuteSql(SqlStm);

    SqlStm = 'create alias qtemp/SALES_alias for ' +
             %trim(schema) + '/SALESTYPE';
    eval sqlcod = ExecuteSql(SqlStm);
Coding the SQLRPGLE program (continued)

```c
// Declare Cursor for STORESALES
C+ exec sql
C+ declare STORESALES cursor for
C+ select *
C+ from ( TRANS_alias left outer join
C+ SALES_alias
C+ on TRANS_alias.trans00001 =
C+ SALES_alias.trans00001 )
C+ where :date = char(date(TRANS_alias.TransactionTime), ISO)
C/end-exec

// Open Cursor for STORESALES
C+ exec sql
C+ open STORESALES
C/end-exec

/free
//Do all rows in STORESALES
dow sqlstmt = '00000';

// Fetch a block of rows from STORESALES cursor
C+ exec sql
C+ fetch next
C+ from STORESALES
C+ for 1000 rows
c+ into :v1_ds :v1_null_ds
C/end-exec
```
Coding the SQLRPGLE program (continued)

// Loop through multi occurrence DS for rows fetched
for i1 = 1 to sqldr3;

// Check if Header section has been written
// ·If not done write Header section
  if header_done = *off;
    eval header_done = *on;

// Output Header section
  Callp WrtSection('Header');

// Update StoreSales variable(s)
  Callp UpdHTMLVar('salesdate': date);
  Callp UpdHTMLVar('storeid': store);

// Output StoreSales section
  Callp WrtSection('StoreSales');
  endif;

// Check if Transaction Id changed
// ·If changed, handle Transaction processing
  if vl_ds(i1).transid <> sv_transid;

// Check if saved Transaction Id populated
// ·If not populate (not 1st time), output End Transaction section
  if sv_transid <> *zeros;
    Callp WrtSection('EndTransaction');
  endif;
// Save Transaction Id
  eval sv_transid = v1_ds(i1).transid;

// Update Transaction variable(s)
  CallIP UpdHTMLVar('type' : v1_ds(i1).transtype);

// Output Transaction section
  CallIP WrtSection('Transaction');
  endif;

// Update SalesItem variable(s)

// Check if brandname contains a &
// Convert embedded & to &amp; special literal since & not valid in XML
if %scan('&:', v1_ds(i1).brandname) > *zero;
  eval i2 = %scan('&:', v1_ds(i1).brandname) -1;
  eval wrk_brandname = %subst(v1_ds(i1).brandname:1:i2) +
    amp +
    %subst(v1_ds(i1).brandname:i2+2);
  CallIP UpdHTMLVar('brand' : wrk_brandname);
else;
  CallIP UpdHTMLVar('brand' : v1_ds(i1).brandname);
endif;
CallIP UpdHTMLVar('name' : v1_ds(i1).itemname);
CallIP UpdHTMLVar('currency' : v1_ds(i1).currency);
CallIP UpdHTMLVar('amount' : %char(v1_ds(i1).amount));
Cодирование SQLRPGLE программы (продолжение)

// Выход секции SalesItem
   Callp WrtSection('SalesItem');
endfor;
enddo;

// Когда обработаны все записи, выполните закрывающиеся разделы и вывод на IFS
Callp WrtSection('EndTransaction');
Callp WrtSection('Trailer');

Callp WrtHtmlToStmF(if$_document:ifs_code_page);

// Установить SQL-запрос для удаления alias'ов и выполнить
SqlStm = 'drop alias qttemp/TRANS_alias';
eval sqlcod = ExecuteSql(SqlStm);

SqlStm = 'drop alias qttemp/SALES_alias';
eval sqlcod = ExecuteSql(SqlStm);

P ExecuteSql b
   D ExecuteSql pi 10i 0
   D SqlStmI 5000 value

C/EXEC SQL
C+ Execute Immediate :SqlStmI
C/END-EXEC

C     return SqlCod

P ExecuteSql e
Some considerations for RPG Composition

• Each call to the `WrtHtmlToStmf` procedure will recreate the document specified by removing and creating each time it is called.

Work around for these limitations:
- Modify the CGI toolkit procedure to remove the limitations noted above. The good news is that since the CGI toolkit is open source, when you download and install the CGIDEV2 toolkit library on our system, the source code is included
XSL & XSLT transform of an XML document to SQL script
The Ins and Outs of XML and DB2 for i5/OS

Extensible Stylesheet Language (XSL) and XSL Transformations (XSLT)

• A stylesheet (XSL) provides formatting and presentation processing for raw underlying XML data
• XSL contains three different types of statements
  – XML Path Language (XPath)
  – XSL Formatting Object (XML-FO)
  – XSL Transformations (XSLT)
• XML-FO provides formatting of XML data for display such as HTML
• XSLT is a language that allows for very powerful manipulation of the data inside an XML document that can actually change the contents of the elements and much more
XPath and Location Path

XPath

– Is one of the cornerstones of the W3C's XSLT standard.
– It is a language that describes how to locate specific elements (attributes, comments, processing instructions, etc.) in an XML document.
– XPath expressions are used to return node sets such as an element, attribute, text, etc.

Location Path

– Is a special case of XPath expression
– It is used by many middleware products including DB2 XML Extender
The function of this XSL is to:
– Decompose the XML data using the default Xalan parser
– Generate an SQL script to insert XML data into our tables(s)

The following are guidelines when coding an XSL:
– Beginning with the Root Element of the XML document, code the transformation stylesheet following the XML hierarchy flow
– Match XML element and attribute names noting the case, spelling and tree structure path
– Use variables (identified by an $ prefix) to save XML data for later use and reuse
– Attributes in the XML data are identified with an @ prefix when coding the XPath directives
– Elements in the XML data have no prefix as do the Attributes when coding the XPath directives
The Ins and Outs of XML and DB2 for i5/OS

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StoreSales date="2006-04-06" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="StoreSales.xsd">
  <StoreId>7</StoreId>
  <Transactions>
    <Transaction type="SALE">
      <SalesItem>
        <Brand name="Pepsi"/>
        <Name>Mt. Dew 20 oz.</Name>
        <Currency>USD</Currency>
        <Amount>1.19</Amount>
      </SalesItem>
    </Transaction>
  </Transactions>
</StoreSales>
```

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Length</th>
<th>Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>StoreId</td>
<td>Integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BrandName</td>
<td>VarChar</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>SalesDate</td>
<td>Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>VarChar</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Currency</td>
<td>VarChar</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Amount</td>
<td>Decimal</td>
<td>9.2</td>
<td></td>
</tr>
</tbody>
</table>
The Ins and Outs of XML and DB2 for i5/OS

XML XPath name to Sales table column mappings

<table>
<thead>
<tr>
<th>XML name</th>
<th>Column name</th>
</tr>
</thead>
<tbody>
<tr>
<td>StoreSales/Date</td>
<td>salesdate</td>
</tr>
<tr>
<td>StoreSales/StoreId</td>
<td>storeid</td>
</tr>
<tr>
<td>StoreSales/Transactions/Transaction/type</td>
<td>type</td>
</tr>
<tr>
<td>StoreSales/Transactions/Transaction/SalesItem/Brand/name</td>
<td>brandname</td>
</tr>
<tr>
<td>StoreSales/Transactions/Transaction/SalesItem/Currency</td>
<td>currency</td>
</tr>
<tr>
<td>StoreSales/Transactions/Transaction/SalesItem/Amount</td>
<td>amount</td>
</tr>
</tbody>
</table>
The Ins and Outs of XML and DB2 for i5/OS

Coding the XSL Stylesheet

```xml
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="text" indent="no" encoding="UTF-8"/>

  <!-- Process complete XPATH tree starting from root-->
  <xsl:template match="/"/>  <!-- begin template match -->

  <!-- Set sales date variable with XML data from Date attribute -->
  <xsl:variable name="date">
    <xsl:value-of select="StoreSales/@date"/>
  </xsl:variable>

  <!-- Set store id variable with XML data from StoreId element -->
  <xsl:variable name="id">
    <xsl:value-of select="StoreSales/StoreId"/>
  </xsl:variable>

  <!-- Process each repeating Transaction element -->
  <xsl:for-each select="StoreSales/Transactions/Transaction">  <!-- begin Transaction -->
    <xsl:variable name="type">
      <xsl:value-of select="@type"/>
    </xsl:variable>
  </xsl:for-each>
```
The Ins and Outs of XML and DB2 for i5/OS

Coding the XSL Stylesheet (continued)

<!-- Process each repeating SalesItem element -->
<xsl:for-each select="SalesItem">
  <!-- begin SalesItem -->
  insert into Sales
  (salesdate,storeid,type,brandname,currency,amount)
  values (
    &apos;xsl:value-of select="$date"&apos;,
    &apos;xsl:value-of select="$id"&apos;,
    &apos;xsl:value-of select="$type"&apos;,
    &apos;xsl:value-of select="Brand/@name"&apos;,
    &apos;xsl:value-of select="Currency"&apos;,
    &apos;xsl:value-of select="Amount"&apos;);
  </xsl:for-each>  <!-- end SalesItem -->
</xsl:for-each>  <!-- end Transaction -->
</xsl:template>  <!-- end template match -->
</xsl:stylesheet>
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

public class Transform {
    public static void main(String argv[]) throws TransformerException, TransformerConfigurationException, FileNotFoundException, IOException {
        if (argv.length != 4) {
            System.err.println("Usage: java Transform sourcexml stylesheet targetxml guid");
            System.exit(1);
        }
        TransformerFactory tFactory = TransformerFactory.newInstance();
        Transformer transformer = tFactory.newTransformer(new StreamSource(argv[1]));
        // Set the global universal unique identified as input parameter guid
        transformer.setParameter("guid", argv[3]);
        // Use the Transformer to apply the associated Templates object to an XML document and write the output to a file
        transformer.transform(new StreamSource(argv[0]), new StreamResult(new FileOutputStream(argv[2])));
        System.out.println("*************** The result is in " + argv[2] + " ***************");
    }
}
The Ins and Outs of XML and DB2 for i5/OS

CMD Interface

CMD  PROMPT('StoreSales XML Decompose')

PARM  KWDC(INPUT) TYPE(INPUT) MIN(1) PROMPT('Input')

INPUT:  ELEM  TYPE(*CHAR) LEN(80) MIN(1) CASE(*MIXED) +
        PROMPT('Directory')
        ELEM  TYPE(*CHAR) LEN(80) MIN(1) CASE(*MIXED) +
        PROMPT('Document')

PARM  KWDC(OUTPUT) TYPE(OUTPUT) MIN(1) +
       PROMPT('Output')

OUTPUT:  ELEM  TYPE(*NAME) MIN(1) PROMPT('Schema')

PARM  KWDC(CLASSPATH) TYPE(*CHAR) LEN(80) +
       DFT(('/XMLRedbook/classes') +
       SPCVAL((*DIRECTORY *DIRECTORY) (*ENVVAR +
       *ENVVAR)) CASE(*MIXED) PROMPT('Classpath')

PARM  KWDC(DEBUG) TYPE(*CHAR) LEN(4) RSTD(*YES) +
       DFT(*NO) VALUES(*NO *YES) PMTCTL(*PMTRQS) +
       PROMPT('Debug')
The Ins and Outs of XML and DB2 for i5/OS

Coding the CLLE program

/* check if classpath s/b resolved to directory name */
B_Classpth: IF COND(&CLASSPATH *EQ '*DIRECTORY') THEN( +
   CHGVAR VAR(&CLASSPATH) VALUE(&DIRECTORY))

/* execute Transform */
B_Debug_1: IF COND(&DEBUG *EQ '*YES') THEN( +
   RUNJAVA CLASS(Transform) +
   PARM(&XMLFILE +
      &XSLFILE +
      &SQLFILE +
      '') +
   CLASSPATH(&CLASSPATH) +
   PROP((java.version 1.4)))

B_Debug_2: IF COND(&DEBUG *EQ '*NO') THEN( +
   RUNJAVA CLASS(Transform) +
   PARM(&XMLFILE +
      &XSLFILE +
      &SQLFILE +
      '') +
   CLASSPATH(&CLASSPATH) +
   PROP((java.version 1.4)) +
   OUTPUT(*NONE))
The Ins and Outs of XML and DB2 for i5/OS

Coding the CLLE program (continued)

;/* create temporary source file for SQL script execution */
CRTPSRCPF FILE(QTEMP/QSQLSRC) RCDLEN(80)
MONMSG MSGID(CPF0000)

;/* copy SQL script to temporary source file member */
CHGVAR VAR(&FRMSTRMF) VALUE(&DIRECTORY |< &DOCUMENT |< '.sql')

CPYFROMSTMF FROMSTMF(&FRMSTRMF) +
   TOBJR('/qsys.lib/qtemp.lib/qsqlsrc.file/Sto+
   reSales.mbr') MBROPT(*REPLACE)

;/* execute SQL script */
RUNSQLSTM SRCFILE(QTEMP/QSQLSRC) SRCMBR(STORESALES) +
   DFRTRDCOL(&SCHEMA)

;/* remove generated SQL script */
CHGVAR VAR(&SQLFILE) VALUE(&DIRECTORY |< &DOCUMENT |< '.sql')

RMVFLNK OBJLNK(&SQLFILE)
Generated SQL Script

insert into Sales
  (salesdate,storeid,type,brandname,currency,amount)
values (
  '2006-04-06',
  '7',
  'SALE',
  'Pepsi',
  'USD',
  1.19);
The Ins and Outs of XML and DB2 for i5/OS

Some considerations for XSL Decomposition

• The `RUNSQLSTM` function has a limit as to the size of an SQL script that can be handled. Currently this limit is 16Mb and must be taken into account if you are attempting to decompose large XML documents.

Work around for this limitation:
- Create an HLL application that reads the generated SQL script and processes each insert one at a time
- Use the DB2 command as part of QShell to execute the SQL script

• The generated SQL script currently can not be executed directly from the IFS directory it is created in.

Work around for this limitation:
- Copy the generated SQL script to a source file and execute using `RUNSQLSTM`
The Ins and Outs of XML and DB2 for i5/OS

RPG decompose of an XML document
Using RPG to decompose XML

The ILE RPG Compiler as of the V5R4 release of i5/OS has new native operations codes and built-in functions that facilitate decomposition of an XML document into data structures defined in the program using a non-validating parser. For more information on the XML parser used by ILE RPG, please refer to Chapter 11 in the *WebSphere Development Studio ILE RPG Programmer’s Guide*, SC09-2507.

The advantage of using the new operation codes and built-in functions added to RPG is that once the hierarchy of the XML document is replicated using data structures in our RPG program code, the decomposition of the XML data and population of data structure fields is done automatically and efficiently without the developer having to manipulate storage to get the XML data into workable form. The complexity has been removed from the XML decomposition process and the developer now can just concentrate on business logic required to implement the XML data. No pointers, API interfaces, buffer manipulation, just straight forward RPG coding.
The Ins and Outs of XML and DB2 for i5/OS

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE CorpSales PUBLIC "CorpSalesId" "CorpSales.dtd">
<CorpSales Date="2006-04-06">
  <CountryInfo>
    <Name>USA</Name>
  </CountryInfo>
  <Brand>
    <Name>Pepsi</Name>
    <Sales>
      <Currency>USD</Currency>
      <Amount>2.38</Amount>
    </Sales>
    <Returns>
      <Currency></Currency>
      <Amount></Amount>
    </Returns>
  </Brand>
</CorpSales>
```

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Length</th>
<th>Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>CountryName</td>
<td>VarChar</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>BrandName</td>
<td>VarChar</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>SalesDate</td>
<td>Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>VarChar</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Currency</td>
<td>VarChar</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>Amount</td>
<td>Decimal</td>
<td>9.2</td>
<td></td>
</tr>
</tbody>
</table>
The Ins and Outs of XML and DB2 for i5/OS

XML XPath name to CorpSales table column mappings

<table>
<thead>
<tr>
<th>XML name</th>
<th>Column name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CorpSales/Date</td>
<td>salesdate</td>
</tr>
<tr>
<td>CorpSales/CountryInfo/Name</td>
<td>countryname</td>
</tr>
<tr>
<td>CorpSales/SalesByBrand/Brand/Name</td>
<td>brandname</td>
</tr>
<tr>
<td>CorpSales/SalesByBrand/Sales/Currency</td>
<td>currency</td>
</tr>
<tr>
<td>CorpSales/SalesByBrand/Sales/Amount</td>
<td>amount</td>
</tr>
<tr>
<td>CorpSales/SalesByBrand/Returns/Currency</td>
<td>currency</td>
</tr>
<tr>
<td>CorpSales/SalesByBrand/Returns/Amount</td>
<td>amount</td>
</tr>
</tbody>
</table>
 CMD Interface

CMD PROMPT('CorpSales XML Decompose')

PARM KWD(INPUT) TYPE(INPUT) MIN(1) PROMPT('Input')

INPUT: ELEM TYPE(*CHAR) LEN(80) MIN(1) CASE(*MIXED) +
       PROMPT('Directory')
       ELEM TYPE(*CHAR) LEN(80) MIN(1) CASE(*MIXED) +
       PROMPT('Document')

PARM KWD(CLASSPATH) TYPE(*CHAR) LEN(80) +
       DFT('/XMLRedbook/classes') +
       SPCVAL((DIRECTORY DIRECTORY) (ENVVVAR +
       ENVVVAR)) CASE(*MIXED) PROMPT('Classpath')

PARM KWD(DEBUG) TYPE(*CHAR) LEN(4) RSTD(*YES) +
       DFT(*NO) VALUES(*NO *YES) PMTCTL(*PMTRQS) +
       PROMPT('Debug')
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program

//CORSLSSHRD CMD parms prototype
D CORSLSSHRD   PR     extpgm('CORSLSSHRD')
D Input        163    Directory/Document
D Class         80     classpath
D Debug         4      Debug function

//CORSLSSHRD CMD parms procedure interface
D CORSLSSHRD   PI
D Input        163    Directory/Document
D Class         80     classpath
D Debug         4      Debug function

//Write table via sql prototype
d WriteTbl     PR     10i 0  Write to table w/Sql

//System command prototype
d qcmdexec      pr     extpgm('QCMDEXC')  Execute CL command
d command       100a
d length        15p 5

// Data structure for external table add
d CorpSales_ds   e DS   extname(CorpSales)  CorpSales table
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program (continued)

// Data structure(s)/column(s) for XML document parsing

<table>
<thead>
<tr>
<th>Field</th>
<th>DS</th>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CorpSales</td>
<td>DS QUALIFIED</td>
<td>XML CorpSales Elem</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>10d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CountryInfo</td>
<td>likeys(CountryInfo)</td>
<td>XML CountryInfo Elem</td>
<td></td>
</tr>
<tr>
<td>SalesByBrand</td>
<td>likeys(SalesByBrand)</td>
<td>XML SalesByBrand Elem</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>DS</th>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>50a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brand</td>
<td>dim(10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>likeys(Sales)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>likeys(Returns)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>DS</th>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brand</td>
<td>DS QUALIFIED</td>
<td>XML Brand Elem</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>80a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>likeys(Sales)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>likeys(Returns)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>DS</th>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales</td>
<td>DS QUALIFIED</td>
<td>XML Sales Elem</td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td>30a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amount</td>
<td>9p 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>DS</th>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>DS QUALIFIED</td>
<td>XML Returns Elem</td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td>30a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amount</td>
<td>9p 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program (continued)

eval XSLdocument = '/XMLRedbook/CorpXML/RemoveEmpty.xsl' + Null;

// Check if user specified special value classpath
select;
when Class = '*DIRECTORY';
    eval XMLclasspath = apos + %trim(%subst(Input:3:80)) + apos;
when Class = '*ENVVAR';
    eval XMLclasspath = (Class);
when Class <> '*DIRECTORY' and Class <> '*ENVVAR';
    eval XMLclasspath = apos + %trimr(Class) + apos;
endsl;

// Check if user specified debug option
// Set RUNJVA command string based on debug option, then execute
select;
when Debug = '*NO';
cmd = 'RUNJVA CLASS(Transform) +
    PARM(' +
        apos + %trimr(XMLdocumentin) + apos + '' +
        apos + %trimr(XSLdocument) + apos + '' +
        apos + %trimr(XMLdocumentout) + apos + '' +
        apos + '' + apos + ') +
    CLASSPATH(' +
        %trimr(XMLclasspath) + ') +
    PROP((java.version 1.4)) +
    OUTPUT(*NONE) ;
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program (continued)

when Debug = '*YES';
cmd = 'RUNJAVA CLASS(Transform) +
   PARM(' +
   apos + %trimr(XMLdocumentin) + apos + '' +
   apos + %trimr(XSLdocument) + apos + '' +
   apos + %trimr(XMLdocumentout) + apos + '' +
   apos + '' + apos + ') +
CLASSPATH(' +
   %trimr(XMLclasspath) + ') +
PROP((java.version 1.4))';
endsl;

cmdlen = %len(%trim(cmd));

callp qcmdexec(cmd:cmdlen);
// Decompose XML into column(s)/data structure(s)
xml-into CorpSales %XML(XMLdocumentout :
   'doc=file +
   allowextra=yes +
   allowmissing=yes +
   case=any');
The Ins and Outs of XML and DB2 for i5/OS

Coding the SQLRPGLE program (continued)

    // Process decomposed XML data from data structure(s)
    eval  salesdate = CorpSales.date;
    eval  count00001 = CorpSales.CountryInfo.Name;

    // Loop through multi occurrence DS
    for i1 = 1 to 10; // b-Brand DS array loop

    // Process Brand DS array XML data
    if CorpSales.SalesByBrand.Brand(i1).name <> *blanks; // b-Brand.name <> *blanks
        eval  brandname = CorpSales.SalesByBrand.Brand(i1).name;

    if CorpSales.SalesByBrand.Brand(i1).Sales.currency <> *blanks; // b-Sales.currency <> *blanks
        eval  type = 'SALE';
        eval  currency = CorpSales.SalesByBrand.Brand(i1).Sales.currency;
        eval  amount = CorpSales.SalesByBrand.Brand(i1).Sales.amount;

        eval returncode = WriteTbl(); // Insert XML data to table(s)
    endif;

    if CorpSales.SalesByBrand.Brand(i1).Returns.currency <> *blanks; // b-Returns.currency <> *blanks
        eval  type = 'RETURN';
        eval  currency = CorpSales.SalesByBrand.Brand(i1).Returns.currency;
        eval  amount = CorpSales.SalesByBrand.Brand(i1).Returns.amount;

        eval returncode = WriteTbl(); // Insert XML data to table(s)
    endif;
    endif;
    endfor;
Coding the SQLRPGLE program (continued)

P WriteTbl b export
D WriteTbl pi 10i 0

// Write out table contents via SQL
C/exec sql
  C+ insert into CorpSales
  C+  (COUNTRYNAME,
  C+    BRANDNAME ,
  C+    SALESDATE ,
  C+    TYPE    ,
  C+    CURRENCY ,
  C+    AMOUNT)
  C+ values(
  C+    :COUNT00001,
  C+    :BRANDNAME ,
  C+    :SALESDATE ,
  C+    :TYPE    ,
  C+    :CURRENCY ,
  C+    :AMOUNT)
  C+ with nc
C/end-exec

  C        return sqlcode

P WriteTbl e
Some considerations for RPG Decomposition

• Use of the Java Virtual Machine (JVM) on a System i requires some simple tuning to optimize performance. Since we use the JVM to run our Transform class it is important to note the memory pool that your JVM executes in be set appropriately to improve performance.

• XML-INTO operation cannot handle empty elements or attributes that contain zero length numeric, date, time or timestamp fields during decomposition.

Work around for this limitation:
- Use a stylesheet (XSL) as we did to remove the empty limitations noted above
The Ins and Outs of XML and DB2 for i5/OS

Some considerations for RPG Decomposition (continued)

- The RPG compiler limits character variables to 65535 characters in length. Data Structures, being considered character fields, are subject to this length limitation as well. You must be aware when replicating the XML tree using data structures, especially data structures that are dimensional, that the total size of the data structure tree does not exceed this limit as well.

Work around for this limitation:
- break apart the data structures without nesting them in side each other and
  use multiple XML-INTO operations to decompose specifying the specific path of the XML document that matches the data structure names.
The Ins and Outs of XML and DB2 for i5/OS

The Future:

While much can be done on the System i with XML already, there are some capabilities that the System i does not yet have.

The future of XML is wide open and should include:
- XQuery
- SQL/XML (but you can create a UDTF w/RPG parser to fake it now)
- XLink
- XML Encryption
- Hybrid DBMS w/native XML support

Using Zend PHP and RPG on System i to compose & decompose XML

Stay tuned for more exciting XML on a System i near you!

You now know What can i do.....i can do XML
The Ins and Outs of XML and DB2 for i5/OS

Additional Materials:

The Redbook this session is based on refers to additional materials that you can download from the Internet at ftp://www.redbooks.ibm.com/redbooks/SG247258.

In addition I recommend you visit www.partner400.com to read through Jon Paris and Susan Gantner’s presentations on RPG, XML, PHP and CGI. The have a wide assortment of examples to get you started.
The Ins and Outs of XML and DB2 for i5/OS

Grok

To understand so thoroughly that the observer becomes a part of the observed—to merge, blend, intermarry, lose identity in group experience. It means almost everything that we mean by religion, philosophy, and science—and it means as little to us (because we are from Earth) as color means to a blind man.
The Ins and Outs of XML and DB2 for i5/OS

Questions?