pureXML Database Technology: An Introduction  
[Part 1 – Of a Two Session Presentation]

Susan Malaika, IBM  
malaika@us.ibm.com  
March 2009 
List of articles to get started:  
http://www.ibm.com/developerworks/wikis/display/db2xml/Technical+Papers+and+Articles#TechnicalPapersandArticles-GettingStarted

Google alphaWorks pureXML  
Google wiki pureXML  

Copyright IBM
pureXML Database Technology : An Introduction

- Learn how you can store, index, and query XML data in IBM DB2 easily without first mapping to relational structures and achieve considerable design and development savings. Schema evolution becomes easier as there is no longer any need to re-structure the way data is stored in the face of XML structure changes. There is no need to associate exactly one schema with the stored XML; the appropriate version of the schema can be used. New SOA, Web Services, Web 2.0, Mashup and Forms solutions for XML are easier to build. A pureXML solution will be described and illustrations will be provided. The session will include a brief introduction to basic XML principles if required by the audience
pureXML Bootcamp – Chicago 19-21 May 2009

• A three day pureXML Bootcamp will be held in Chicago 19-21 May 2009 to include tutoring on XQuery and SQL/XML, XML storage and indexing, XML Schema handling.
  – The Bootcamp has plenty of hands on labs to practice what you’ve learnt – including building a complete XML application.
  – Feedback we’ve had from attendees includes “Fantastic”, “I really enjoyed the Bootcamp”

• See photos from other bootcamps here:

Please add your title, name, company, email address, phone, and IBM rep (if you know who it is) to the sign-up sheet if you would like to be considered for this bootcamp or send e-mail to malaika@us.ibm.com. Places are limited.
Agenda

- Introducing XML
- Storing what you Exchange
- DB2 pureXML Features
- A Simple Scenario
- Getting Started and Next Steps

Demonstrations and Downloads are (available on the Internet) that include HR-XML, IMS, FpML, FIXML, ISO 20022, TWIST, XBRL GL and many other formats

Google alphaWorks pureXML
Introducing XML
eXtensible Markup Language
Some XML Terminology

An XML Document, or Instance, or Message

<book>
  <authors>
    <author id="47">John Doe</author>
    <author id="58">Peter Pan</author>
  </authors>
  <title>Database systems</title>
  <price>29</price>
  <keywords>
    <keyword>SQL</keyword>
    <keyword>relational</keyword>
  </keywords>
</book>

Start Tag

Data or Content or Text Node

End Tag

Attribute

Element
XML Document Representation

XML Parsers often support:
• DOM: Document Object Model
• SAX: Simple API for XML

XML Parsing

<book>
  <authors>
    <author id="47">John Doe</author>
    <author id="58">Peter Pan</author>
  </authors>
  <title>Database systems</title>
  <price>29</price>
  <keywords>
    <keyword>SQL</keyword>
    <keyword>relational</keyword>
  </keywords>
</book>

Serialization

XML Representations include:
Text, Tree, Streams....

Copyright IBM
## Well-formed XML Documents

An XML document is well-formed, if:

<table>
<thead>
<tr>
<th></th>
<th>Not Well-Formed Example</th>
<th>Well-Formed Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>The document has exactly one root element</td>
<td><code>&lt;b&gt;bla&lt;/b&gt;</code> <code>&lt;c&gt;blub&lt;/c&gt;</code></td>
<td><code>&lt;a&gt; &lt;b&gt;bla&lt;/b&gt; &lt;c&gt;blub&lt;/c&gt; &lt;/a&gt;</code></td>
</tr>
<tr>
<td>Each opening tag is matched by a closing tag</td>
<td><code>&lt;a&gt;&lt;b&gt;bla&lt;/a&gt;</code></td>
<td><code>&lt;a&gt;&lt;b&gt;bla&lt;/b&gt;&lt;/a&gt;</code></td>
</tr>
<tr>
<td>All elements are properly nested</td>
<td><code>&lt;a&gt;&lt;b&gt;bla&lt;/b&gt;&lt;/a&gt;</code></td>
<td><code>&lt;a&gt;&lt;b&gt;bla&lt;/b&gt;&lt;/a&gt;</code></td>
</tr>
<tr>
<td>Attribute values are quoted</td>
<td><code>&lt;a id=15&gt;&lt;/a&gt;</code></td>
<td><code>&lt;a id=“15”&gt;&lt;/a&gt;</code></td>
</tr>
<tr>
<td>Does not use disallowed characters in tags or values</td>
<td><code>&lt;a&gt; 3&lt;5 &lt;/a&gt;</code></td>
<td><code>&lt;a&gt; 3&amp;lt;5 &lt;/a&gt;</code></td>
</tr>
</tbody>
</table>

Note: `xml header <?xml version=“1.0”?>` is NOT required for well-formedness.  
See [http://www.w3.org/TR/REC-xml](http://www.w3.org/TR/REC-xml) for full definition.
“Well-formed” XML or “Valid” XML

• An XML document is **well-formed**, if…
  – …it complies with the rules on the previous page
  – i.e., it can be parsed by an XML parser without error

• An XML document is **valid**, if…
  – …it is well-formed AND
  – …it complies with a specific DTD or XML Schema
    • XML Parsers can optionally perform “validation”

• **DTDs (Document Type Definitions) and XML Schemas define** a specific XML document structure
XML Namespaces

A **prefix** identifies domain (“namespace”), and distinguishes between duplicate element names and names from different domains

```xml
<job:title>Database Administrator</job:title>

<person:title>Mr</person:title>

<movies:title>Gone with the wind</movies:title>
```

Namespaces need to be uniquely identified: URIs are used
URI = Universal Resource Identifier

URI Examples:
  http://www.ibm.com/db2xml
  http://abcdefghijklmn.xyz

• URIs uniquely identify a namespace
• URIs typically look like a URL
• URIs are just an identifier, they may point to a web page, but don’t have to!

For more details on URIs see http://www.ietf.org/rfc/rfc2396.txt
Namespace Declaration and URIs

• Example:
  – element name: person
  – namespace URI:  http://www.foobar.org
  – namespace prefix:  foo
    <foo:person xmlns:foo="http://www.foobar.org">
      <foo:name>John Doe</foo:name>
    </foo:person>

The reserved attribute `xmlns` defines namespaces, and (optionally) assigns them to a namespace prefix.

*The namespace applies to the current element and all sub-elements and attributes that it contains.*
Multiple Namespaces

<cust:person xmlns:cust="http://www.foobar.com/customer">
  <cust:name>John Doe</cust:name>
    <prod:name>Thinkpad T40</prod:name>
    <prod:orderdate>2004-11-18</prod:orderdate>
  </prod:product>
</cust:person>

*The namespace applies to the current element and all sub-elements and attributes that it contains – unless it’s overridden!*
Default Namespaces

A namespace declaration without prefix defines a default namespace. The namespace is implicit for all elements in scope, without using a prefix.

```xml
<person xmlns="http://www.foobarn.org">
  <age>45</age>
  <name>
    <first>John</first>
    <last>Doe</last>
  </name>
</person>
```

The default namespace applies to the current element and all sub-elements that it contains.
XML Schema

- Defines structure, content, data types for XML documents
- Consists of 1 or more schema documents, sometimes known as XSDs
- A schema document can define a namespace (optionally)
- Example:
  - 1 XML Schema, 3 Schema Documents, 2 Namespaces

```
Order

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ORDER.XSD

include

Lineitem

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LINEITEM.XSD

import

parts.xsd
```

[http://www.w3.org/TR/xmlschema-0/](http://www.w3.org/TR/xmlschema-0/)

Copyright IBM
<xsd:schema targetNamespace="http://www.mycompany/products"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:simpleType name="PriceType">
    <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="100000"/>
      <xsd:totalDigits value="9"/>
      <xsd:fractionDigits value="3"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="StockPriceType">
    <xsd:sequence>
      <xsd:element name="Ask" type="PriceType"/>
      <xsd:element name="Bid" type="PriceType"/>
      <xsd:element name="P50DayAvg" type="PriceType"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="StockPrice" type="StockPriceType"/>
</xsd:schema>
XML Schema: Simple & Complex Types

PriceType: derived from “Decimal” by defining additional restrictions

Once defined, PriceType can be used multiple times!

A valid instance document

Copyright IBM
Why XML?

• XML: a notation for data exchange between systems and applications that have not necessarily been formally introduced to each other.
• XML: enables the creation of precise descriptions for admissible content - you can define the tags
  – Encoding: Allowable characters and allowable character encodings - to support diverse platforms (operating environments and hardware).
    • Ways to discover the particular encoding
  – Schemas and Constraints: Methods for defining domain specific and general purpose content -
    • Ways to discover, access and process the schemas and constraints
  – Namespaces: Mixing of domain specific content from different sources
    • Ways to use different schemas in a single XML document
Storing What You Exchange
## The Architecture of Standards

### Composite Industry Standards

### Industry-Specific Standards

### Horizontal Industry Standards (cross-industry)

### Information Technology Standards (cross-industry)

### Examples

<table>
<thead>
<tr>
<th>Formats:</th>
<th>IHE, ARTS, RosettaNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formats:</td>
<td>ACORD, FpML, FIXML, MISMO, HL7, HR-XML, NewsML, MDDL, MusicXML, OTA, STAR, TWIST, UBL, UNIFI (ISO 20022), XBRL</td>
</tr>
<tr>
<td>ebXML, OAGIS, RFID</td>
<td></td>
</tr>
<tr>
<td>Formats:</td>
<td>XML, Web Services, Atom</td>
</tr>
<tr>
<td><strong>Communication:</strong></td>
<td>TCP/IP, HTTP</td>
</tr>
<tr>
<td><strong>Addressing:</strong></td>
<td>URI</td>
</tr>
<tr>
<td><strong>Programming:</strong></td>
<td>Java, SQL</td>
</tr>
</tbody>
</table>

Copyright IBM
Some XML Industry Formats Snapshot

- 15 Industry Formats are listed in the next two charts
- The numbers of schema files range from 1-600 (FpML has 23)
- The numbers of elements range from 84 to 77319 (FpML has 1887)
- The numbers of attributes range from 20 to 4300 (FpML has 196)

A Tip:
IBM has an XML Schema Flattener here
http://www.alphaworks.ibm.com/tech/xfg
select an xsd file, right click mouse and see Schema>Flattener
## XML Schema Characteristics for Some Industry Formats [1]

<table>
<thead>
<tr>
<th>Industry Format</th>
<th>Version</th>
<th>Types</th>
<th>Elements</th>
<th>Attributes</th>
<th>XSD's</th>
<th>Max. XSD size in kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACORD (Association for Cooperative Operations Research and Development)</td>
<td>(XMLife) 2.16.01</td>
<td>1369</td>
<td>9378</td>
<td>1275</td>
<td>4</td>
<td>743</td>
</tr>
<tr>
<td>ARTS (Association for Retail Technology Standards)</td>
<td>1.0 - 3.0</td>
<td>4825</td>
<td>6305</td>
<td>2011</td>
<td>32</td>
<td>625</td>
</tr>
<tr>
<td>CDISC (Clinical Data Interchange Standards Consortium)</td>
<td>00-9-03</td>
<td>98</td>
<td>84</td>
<td>71</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>FpML (Financial Products Markup Language)</td>
<td>4.2</td>
<td>686</td>
<td>1867</td>
<td>196</td>
<td>23</td>
<td>130</td>
</tr>
<tr>
<td>FIXML (Financial Information Exchange)</td>
<td>4.4</td>
<td>1310</td>
<td>619</td>
<td>2593</td>
<td>41</td>
<td>797</td>
</tr>
<tr>
<td>HL7CDA (Clinical Document Architecture)</td>
<td>3</td>
<td>1953</td>
<td>945</td>
<td>477</td>
<td>6</td>
<td>749</td>
</tr>
<tr>
<td>IRS1120 (IRS e-File Form 1120)</td>
<td>2006v3.3</td>
<td>3415</td>
<td>11591</td>
<td>2632</td>
<td>600</td>
<td>214</td>
</tr>
<tr>
<td>MISMO (Mortgage Industry Standards Maintenance Organization)</td>
<td>2.3 - 2.4</td>
<td>2899</td>
<td>1087</td>
<td>13733</td>
<td>31</td>
<td>2865</td>
</tr>
</tbody>
</table>
### XML Schema Characteristics for Some Industry Formats [2]

<table>
<thead>
<tr>
<th>Industry Format</th>
<th>Version</th>
<th>Types</th>
<th>Elements</th>
<th>Attributes</th>
<th>XSD's</th>
<th>Max. XSD size in kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCJE (NIEM) (Minnesota Criminal Justice Event (National Information Exchange Model))</td>
<td>1</td>
<td>415</td>
<td>936</td>
<td>46</td>
<td>7</td>
<td>1661</td>
</tr>
<tr>
<td>OTA (OpenTravel Alliance)</td>
<td>2003/5</td>
<td>27293</td>
<td>24893</td>
<td>43141</td>
<td>234</td>
<td>538</td>
</tr>
<tr>
<td>STAR (Standards for Technology in Automotive Retail – OAGIS)</td>
<td>5.0.4</td>
<td>5846</td>
<td>77319</td>
<td>625</td>
<td>192</td>
<td>1200</td>
</tr>
<tr>
<td>TWIST (Transaction Workflow Innovation Standards Team)</td>
<td>3.1</td>
<td>1016</td>
<td>2314</td>
<td>20</td>
<td>19</td>
<td>154</td>
</tr>
<tr>
<td>UBL (Universal Business Language)</td>
<td>2</td>
<td>682</td>
<td>2665</td>
<td>253</td>
<td>43</td>
<td>938</td>
</tr>
<tr>
<td>UNIFI (ISO20022 UNiversal Financial Industry message scheme)</td>
<td>1.01 - 2.01</td>
<td>5082</td>
<td>9747</td>
<td>127</td>
<td>71</td>
<td>42</td>
</tr>
<tr>
<td>XBRL - GL (Extensible Business Reporting Language - Global Ledger)</td>
<td>2006-10-25</td>
<td>1858</td>
<td>2847</td>
<td>383</td>
<td>45</td>
<td>39</td>
</tr>
</tbody>
</table>

*These tables are intended to give an idea of the different design styles for schemas – The schemas may have moved on since these tables was created.*
XML Data Exchange

Storing what is exchanged – benefits include

- Storing and querying XML without needing to re-map exchange data to relational format
- Providing common integrity, recovery, security, management interfaces and mechanisms
- Providing flexibility and speed in the face of schema evolution
- Supporting easy access through Web Services and Web 2.0
- Increasing the likelihood that business users understand the stored data
- Simplifying database design
XML At The Edges

Format
Addressing
Communication

The External World

[Structured to support the needs of the business or users]

XML

Relational Tables
[Normalized to support the needs of the computer system]

The Internal World

Copyright IBM
What is Exchanged is Transformed and Stored

Universal Client

Format Specific Client

Agreed Exchange Format

Communication

<Address>

Agreed Exchange Format

Custom Code to map between the exchange and stored formats

Stored Data

Copyright IBM
Service Oriented Architecture

• A key aspect of SOA is the use agreed interfaces between consumers and service providers
• Often custom programming on the server side of SOA is devoted to mapping between relational data formats and XML formats
• The implications of storing what is exchanged include the increased ability to create universal services (on the server side) leading to:
  – Simpler design
  – Simpler programming and increased re-use for both service consumer and the service provider
  – Fewer services
  – Increased automation
What is Exchanged is What is Stored

Universal Client

Format Specific Client

Agreed Exchange Format

Communication

<Address>

Agreed Exchange Format

Universal Server

Stored Data

Agreed Exchange Format

Copyright IBM
XML Storage Options

Unstructured XML storage: XML as text

Extract selected elements/attr.

XML DOC

CLOB Column

Side Tables
or Indexes

Any sub-document level access requires XML parsing – slow.

Shredding: XML → Relational

Fixed Mapping

(regular relational tables)

Mapping prevents XML schema changes, and is often too complex. XML reconstruction is slow.

DB2 pureXML: XML as XML

XML Column

Maximum flexibility and performance

Copyright IBM
DB2 pureXML
With DB2 9 you can:

• Store well-formed XML messages in the database as collections of XML (in an XML column of type XML)
  – Query stored XML messages
    • Using XQuery including update – UNIX and Windows only
    • Using SQL/XML
    • Using some combination of XQuery, XPath, and SQL/XML
  – Optionally index the XML messages
  – Optionally validate the XML messages with respect to an XML schema
  – Store documents with different XML structures in the same column
    • Helps with schema evolution
  – Get started with industry formats such as FpML through pureXML industry bundles
    – In summary, you can process many XML messages with one declarative request – [contrast with XML storage in files]

• Store XML messages as relational data
  – Shedding technology makes it possible to transform XML into relational data by using either the XMLTABLE function or the annotated schema shred
DB2 pureXML Storage

- DB2 stores XML in parsed hierarchical format (~DOM)
  
  ```sql
  create table OTCderiv (FpMLKey char(8), ..., FpML xml);
  ```

- Relational columns are stored in relational format (tables)

- XML columns are stored natively

- No XML parsing for query evaluation!

- XML document in one cell of a table
DB2 pureXML APIs

- Native XML data type (server & client side)
  - (not Varchar, not CLOB, not object-relational !)
- XML Capabilities in all DB2 components
- Applications combine XML & relational data
Ways of Querying XML Data in DB2

• XQuery
  – XQuery as the primary language
  – Optional: SQL embedded in XQuery

• SQL/XML
  – SQL as the primary language
  – Optional: XQuery embedded in SQL
XQuery Construct: FLWOR Expression

• **FOR**: iterates through a sequence, bind variable to items
• **LET**: binds a variable to a sequence
• **WHERE**: eliminates items of the iteration
• **ORDER**: reorders items of the iteration
• **RETURN**: constructs query results

FOR $movie in collection('movies')
LET $actors := $movie//actor
WHERE $movie/duration > 90
ORDER by $movie/@year
RETURN <movie>
   {$movie/title, $actors}
</movie>
XQuery with SQL

create table dept (deptID char(8) primary key, deptdoc xml);

- Identify XML data by a SELECT statement
- Leverage predicates/indexes on relational columns

for $d$ in \texttt{db2-fn:sqlquery}('select deptdoc from dept
where deptID = "PR27" ')… (single document)

for $d$ in \texttt{db2-fn:sqlquery}('select deptdoc from dept
where deptID LIKE "PR\%" ')… (some documents)

for $d$ in \texttt{db2-fn:sqlquery}('select dept.deptdoc from dept, unit
where dept.deptID=unit.ID
and unit.headcount > 200')..... (some documents)
XMLTABLE: Make a Table from XML

```
<dept bldg=101>
  <employee id="901">
    <name>
      <first>John</first>
      <last>Doe</last>
    </name>
    <office>344</office>
  </employee>
  <employee id="902">
    <name>
      <first>Peter</first>
      <last>Pan</last>
    </name>
    <office>216</office>
  </employee>
</dept>
```

```
SELECT X.* FROM dept,
XMLTABLE ('$d/dept/employee' passing deptdoc as "d")
COLUMNS
  empID INTEGER PATH '@id',
  firstname VARCHAR(30) PATH 'name/first',
  lastname XML PATH 'name/last',
  office INTEGER PATH 'office') AS X
```

<table>
<thead>
<tr>
<th>empID</th>
<th>firstname</th>
<th>lastname</th>
<th>office</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>John</td>
<td>Doe</td>
<td>344</td>
</tr>
<tr>
<td>902</td>
<td>Peter</td>
<td>Pan</td>
<td>216</td>
</tr>
</tbody>
</table>
Publishing: Make XML from Tables

```
SELECT
    XMLELEMENT(NAME "Department",
                XMLATTRIBUTES (e.dept AS "name" ),
                XMLAGG( XMLELEMENT (NAME "emp", e.firstname) )
     ) AS "dept_list"
FROM employee e
WHERE ..... 
GROUP BY e.dept;
```

default   lastname  dept
SEAN       LEE       A00
MICHAEL    JOHNSON   B01
VINCENZO   BARELLI  A00
CHRISTINE  SMITH    A00

```
XQuery Transform – for Updating

• Subdocument update
• The XQuery *transform* expression:
  – Creates modified copies of existing nodes, documents, or sequences
  – Typically 3 (4) clauses:
    • transform (<- optional keyword)
    • copy
    • modify (replace, delete, rename, insert)
    • return
  – Example:

```xml
update xmlcustomer
set info = xmlquery( 'transform
  copy $new := $i
  modify do replace value of $new/customerinfo/addr/pcode-zip
  with 90111
  return $new'
passing info as "i")
where cid = 1000;
```
Regions index facilitates access to document regions in the XML data area.

Like LOBs, XML data is stored separately from the base table.

Unlike LOBs, XML data is buffered in the buffer pool.
Page Size for XML

- Larger documents get split into regions
- Max. doc size: 2GB, spans many pages

Fewer regions per document are better for performance.
Choose page size depending on document size, larger pages are better!
DB2 pureXML Indexing Examples

create unique index idx1 on customer(info)
generate key using
xmlpattern '/customerinfo/@Cid'
as sql double;

create index idx2 on customer(info)
generate key using
xmlpattern '/customerinfo/name'
as sql varchar(40);

create index idx3 on customer(info)
generate key using
xmlpattern '//name'
as sql varchar(40);

create index idx4 on customer(info)
generate key using
xmlpattern '/customerinfo/phone'
as sql varchar(40);

create table customer(info XML);

easy to index repeating elements

<&customerinfo Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <state>Ontario</state>
    <pcode>M3Z-5H9</pcode>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone type="home">416-555-3376</phone>
  <assistant>
    <name>Peter Smith</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
DB2 pureXML Schema Flexibility

Document validation for zero, one, or many schemas per XML column:
Always Well Formed XML

(a) No Schema
(b) One Schema
(c) Schema V4.2 & Schema V4.3
(d) Documents w/ and w/o schema
(e) Any mix you want!
IBM Data Studio XML Capabilities

- Easily perform the following XML operations for DB2 pureXML
  - Create XML documents and schemas
  - Import and export XML documents and schemas
  - Insert XML documents into XML columns
  - Generate XML Schema registration scripts
  - Register XML Schema

XML Editors
pureXML Summary

- Hierarchical XML data storage & processing
- No fixed XML schema per XML column
  - Well formed XML
- Schema validation is optional & per document
- XML indexes for specific elements/attributes
- XQuery and SQL/XML Integration

create table dept (deptID char(8),..., deptdoc xml);
DB2 9.5 “Transactional pureXML”

Half the storage of XML Flat Files/CLOBs
Double the throughput of DB2 9 XML Files, CLOBs

DB2 9.5 pureXML
DB2 9.5 Transactional XML

• **Inlining and compression**
  – *Major NA bank* gets *6x* storage savings and *2x* faster inserts than DB2 9
  – *Transaction Processing over XML ( TPOX )* benchmark – based on *FIXML* : 
    • *1/3* storage, *2x* throughput than DB2 9

• **Industry first XQuery Update**
  – *2-3 times* faster than the approach in DB2 9

• **Faster insert with schema validation**
  – *Up to 5 times* faster than DB2 9

• **Instant compatible schema evolution**

• **Enable existing customer base**
  – *Non-Unicode, Offline Load, Replication, Federation*

• **Richer tool support:**
  – *IBM Data Studio, Rational Data Architect, DB2 Warehouse, and many more*
  – *Info 2.0*
  – *Altova, Skytide, and many more*
Financial OLTP with FIXML
In DB2 9.5 - 1/3 Storage, 2 Times Throughput – compared to DB2 9
100 concurrent users, 12-way AIX, TPOX benchmark [http://tpox.sourceforge.net/]

![TPoX throughput by workload type with inlining and compression]

<table>
<thead>
<tr>
<th>Page Type</th>
<th>V9</th>
<th>Vanilla V9.5</th>
<th>Inlined V9.5</th>
<th>Compressed V9.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Tables</td>
<td>21,178</td>
<td>21,178</td>
<td>628,518</td>
<td>259,706</td>
</tr>
<tr>
<td>XDA</td>
<td>955,600</td>
<td>632,448</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>User Indexes</td>
<td>88,388</td>
<td>88,233</td>
<td>88,122</td>
<td>88,312</td>
</tr>
<tr>
<td>Path/Region Indexes</td>
<td>17,961</td>
<td>15,458</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>1,083,127</td>
<td>757,317</td>
<td>716,896</td>
<td>348,274</td>
</tr>
<tr>
<td>GB (= 1024**3)</td>
<td>16.5</td>
<td>11.6</td>
<td>10.9</td>
<td>5.3</td>
</tr>
</tbody>
</table>
A Simple Solution
XML End-to-End Approach: Setting the Scene

A query-able store for well-formed XML that match external entities
A collection of simple services to manipulate the stored XML
Characteristics of Data Layer

**Reduce Server Side Customization**
Data storage structures match the structures in the external world

**Prototype Applications Quickly**
A detailed data model is unnecessary to get started

**Decrease Complexity**
Reduced need for special mappings to a custom storage format

**Respond to Data Structure Changes with Agility**
New data structures can be incorporated easily
Universal Services

• The services are a set of operations, such as insert, update, delete, and query, exposed as Web service operations that operate on stored well-formed XML that represent business objects.
• The services can be set up quickly, for example for prototyping.
• The services allow users or software to query and modify XML data stored through REST or SOAP, which can serve as a basis for further development of various applications, such as forms or SOA applications.
XML Forms Generator (XFG)

- Analyze WSDL of Web Service
- Retrieve sample XML document stored in DB2 pureXML database
- Analyze structure of XML document
- Generate XForms based on structure of XML document

XML Database containing XML documents

http://www.alphaworks.ibm.com/tech/xfg
Characteristics of Forms Layer

Reduce Server Side Customization

Significant customization can take place in the form

Prototype Applications Quickly

The universal services can be used directly

Decrease Complexity

Much of the complexity is isolated in the form or user interface

Respond to Data Structure Changes with Agility

New data structures can be incorporated into the forms as needed
Atom Feeds Demonstration with pureXML
Aggregator (also known as feed reader) retrieves feeds through feed links and displays feeds.
pureXML and Lotus SameTime - Example

Google same-time pureXML
Getting Started

Next Steps
Getting Started: The pureXML Industry Bundles and Demos

• You can read about the demos and bundles here:
  Get started with Industry Formats and Services with pureXML
• You can download the bundles here - Google pureXML alphaWorks
  • Select Download Now to locate the public Industry Bundles
  • Includes FpML, FIXML, TWIST and UNIFI (ISO 20022) bundles
• You can try the interactive demo that includes query, feeds, forms Web and RESTFUL services here - Google pureXML alphaWorks
  • Select View Demo to locate the public interactive demo

Please follow the “Before You Begin” instructions in the “Getting Started with the Demo” PDF
Getting Started: Try pureXML Industry Format Demo

UI

Web Browser XQuery, etc
XForms Plugin
Specialized Format Plugins (e.g., SVG, MusicXML)
Feed Reader

Services

Universal “Quick” Services
Feeds

Data

pureXML

XML: End to End

Copyright IBM
Getting Started: Industry Formats Demo

Structure

1. All the demos have the same table structure that includes:
   - A single pureXML column called document

2. The services layer is the same for all demos
   - The same set of Web, RESTful, and feed services that are exposed on the pureXML column

3. Web browsers and plug-ins provide the UI
   - Some formats have special plug-ins or are recognized by regular Web browsers, e.g., Feed Formats, SVG etc
   - Some formats have customized format specific UIs through XForms
     - Regular browser plug-in used

   All the UI uses the services layer
   - No format specific customization is needed on the server side
Getting Started: Try a pureXML Industry Bundle

1 Register industry schemas
2 Insert and query XML industry messages
3 Produce query results
4 Transform some XML into relational tables
5 **End Result:** XML & relational tables, with suitable queries for the industry format
Getting Started: Build A Small Application

• Step 1 - Set up an Industry Bundle

• Step 2 - Set up the Universal Services

• Step 3 – Set up some Forms or an interface of some kind, e.g.,
Summary of Benefits of Storing XML

- Simpler designs (exchange format same as storage format)
- Quicker implementation (no need to code mappings)
- Faster execution (no need to assemble and disassemble the XML)
- Straightforward schema evolution (no need to re-design and re-implement mapping code each time XML schema changes)
- Easier to understand systems (more people understand the internals of the systems - only a few people understand the relational designs)
To Find Out More

- DB2 9: pureXML Overview and Fast Start

  - Celebrating 10 years of XML

- To locate the Interactive Demo and Industry Bundles and accompanying article
  - [Google pureXML alphaWorks](http://www.alphaworks.ibm.com/tech/purexml)

- To find out more about pureXML:
  - [Google pureXML wiki](http://www.ibm.com/developerworks/wikis/display/db2xml/Home)

- To find out about customers who are using pureXML
More Reading Materials

- Improving the Derivatives Trading Process with pureXML (FpML based)
  http://www.ibm.com/industries/financialservices/doc/content/bin/fss_improving_derivatives_trading_process.pdf
- Automating the confirmation of derivatives trades (FpML based)
- Supporting SEPA with DB2 pureXML (ISO20022 - UNIFI based)
- Exposing DB2 9 pureXML using WID
- pureXML in mashups http://www.youtube.com/watch?v=ckGfhIZW0BY
Performance Articles

• DB2 9 XML Performance Characteristics (FIXML based)

• 15 best practices for pureXML performance in DB2 9

• A performance comparison of DB2 9 pureXML and CLOB or shredded XML storage

• Exploit XML indexes for XML query performance in DB2 9
If you need a DB2 9 System with pureXML and don’t have one …

DB2 Express-C home:

DB2 Express-C Forum:

DB2 Express-C Download:
http://www.ibm.com/software/data/db2/express/download.html
The Industry’s first 1TB XML Benchmark

The System Under Test
- IBM DB2 pureXML 9.5 FP2
- Linux64, SLES 10
- Intel® Xeon® 7400 Processor Server
- 4 CPUs, 6 cores/CPU, 2.67 GHz
- 64GB memory
- 135 disk drives

The Benchmark
- Open Source Benchmark: TPoX (Transaction Processing over XML Data)
  - http://tpox.sourceforge.net/
- 1TB raw data, 360M XML Documents
- 70% Queries, 30% Insert/Update/Delete,
- 200 concurrent users

The Score Card
- DB2 Compression Ratio: 64%
- Mixed Workload:
  → 6,763 Transactions per second
- 1.48x speed-up: Intel® Xeon® 7300 to Intel® Xeon® 7400 Processor
- Insert-only workload (600 users):
  → 11,900 inserts per second
  → Ingestion rate: ~100GB/hour
- All tests performed by Intel at Intel Labs

TPoX Transactions per Second (TTPS)

Intel® Xeon® Processor 7300 Series and Intel® Xeon® Processor 7400 Series are trademarks of Intel Corporation in the U.S. and other countries.

Copyright IBM
Trademarks for Benchmark

- Intel® Xeon® Processor 7300 Series and Intel® Xeon® Processor 7400 Series are trademarks of Intel Corporation in the U.S. and other countries.
- Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations