XML and Relational Database Integration

Stephen A. Brodsky, Ph.D.
IBM
Change and Improvement Have Been Daunting

How do I deliver business flexibility?

Can Information Become a Service?
XML Solves Business Problems Today

- **Business to Business Integration**
  - Platform independent transport mechanism
  - *Purchase order triggers transactions flowing over a service oriented architecture*

- **Document Management**
  - Government and legal industry require digital signature
  - *Tax forms require signature & change year to year*
  - Documents often contain sub-documents
  - *Literary materials contain books, chapters, and sub-chapters*

- **Business Intelligence**
  - Universal representation from multiple sources
  - *Claims adjustor reviews damage estimates from multiple garages with consideration of original format*
XML Today and Tomorrow…

SOA Reference Architecture
Supporting your SOA Lifecycle

**XML is the Fabric of Services Oriented Architectures**
Information as a Service
Moving to a Flexible Architecture (SOA)

Standards-based
- e.g., XQuery, SDO
- JDBC, Web Services...

Information as a Service
Optimize, Virtualize, Integrate, Accelerate

Extracted or Real-time

Heterogeneous Applications & Information

Master Data, Entity Analytics, Decision Portals, Executive Dashboards, Industry Data Models

DB2
IBM Content Manager
Oracle

and more...
Information as a Service
Moving From a Project-Based to a Flexible Architecture

Sources:
Heterogeneous Applications & Information

Destinations:
Business functions
(roles, applications)

Standards based: e.g.,
XQuery, SDO, JDBC,
Web Services...

Services Oriented
Architecture

Metadata
Management

XML

Delivery &
Messaging

Real Time...
& or Extracted

Data Access
& Integration
XML Is Central to Delivering Information as a Service

**Analysis**
- XML formats... PMML
- Metadata
- Reporting formats
- Search, UIMA

**Information Integration**
- Data Transformation
- Data Interchange
- Metadata Interchange
- Publishing

**Business Standards**
- HIPAA (Insurance / HC)
- RFID / EPC
- HL7 (Healthcare)
- Basel II, SOX...

**Master Data**
- XML Master Data
- Web Services
- Search
- Metadata

**Content Mgmt.**
- XML documents
- XPath in JSR 170
- Text Search
- Web Services
- Metadata

**Data Services**
- XQuery and SQL XML
- Web Services
XML Schemas by Industry

**Banking**
- ISO 15022, SPIFA, SPARCS, etc...
- IFX Standard
- OFX Standard
- STEP2
- MISMO Standard

**Financial Markets**
- FIX protocol, FIXML
- Message Standard for Post Trade & Pmts
- Market Data Definition Language (MDDL)
- Research Information ML (RIXML)
- Financial Products ML (FPML)

**Insurance**
- XML for P&C, Life Insurance, etc...

**Chemical & Petroleum**
- Chem eStandards, CyberSecurity
- PIDX Standard

**Healthcare**
- HL7 Standard
- DICOM Standard
- SNOMED Standard
- LOINC Standard
- HIPAA
- SCRIPT, Mfg. Rebate Stdts.
- DoD XML, others
XML Schemas by Industry (Continued)

**Life Sciences**
- numerous technical standards
- MIAME, MAGE, etc...
- LSID, others...
- HL7 Standard
- DICOM Standard
- CDISC Standards for ODM, LAB, ADaM, etc..

**Retail**
- UCCNET including EAN-UCC
- many existing standards
- ePC Network & standards
- ARTS XML for Retail (IXRETAIL)

**Electronics**
- PIPs, RNIF, Business Directory, etc...
- Open Access Standards

**Automotive**
- ebXML, and other B2B Standards
- STAR XML

**Telecommunications**
- eTOM, NGOSS, etc...
- Parlay Specification

**Energy & Utilities**
- IEC working group14, multiple standards
- CIM
- Multispeak

**Cross Industry**
- PDES/STEP ISO 13003
- SMPL Standards
- Radio Frequency ID (RFID)

**Global Grid Forum (GGF)**
**Microarray Gene Expression Data Society (MGED)**
**Interoperable Informatics Infrastructure Consortium (I3C)**
**Health Level 7**
**Digital Imaging and Communication in Medicine**
**Clinical Data Interchange Standards Consortium**

**Uniform Code Council, Inc**
**Global Commerce Initiative (GCI)**
**Auto-ID Center**
**Association of Retail Technical Standards (ARTS)**

**Rosetta Net**
**OpenEDA.Org**

**Automotive Industry Action Group**
**Standards for Technology in Automotive Retail**

**Telemangement Forum (TMF)**
**The PARLAY Group**

**International Electrotechnical Commission**
**Distributed Management Taskforce (DMTF.ORG)**
**Multispeak2.ORG**

**PEDS Inc**
**Voluntary Interindustry Commerce Standards Association**
**EPCGlobal is a subsidiary of ECCnet**
XML Is Changing The Database Industry:  *Inflection Point*

**Performance/Scale, Availability, Security…**

**Query Flexibility**

- **XML RDB Columns**
- **Relational Databases**
- **Hierarchical Databases**

**Application Independence**

- **Files**

**Hybrid Capabilities**

- **Flexible Infrastructure**
  - XQuery and/or SQL
  - XML and/or Relational Data

**Complex Schema Evolution**

- **Sequences & Hierarchies**

**XML Documents, Interchange**

- **XML**
- **Databases**

- **Content Repositories**

- **Digitized Content**
Contrasting the Models – XML and Relational

**Relational**

- **Strength:** Static data
  - Strict schema ensures data integrity
  - High performance indexing on fixed data

- **Strength:** ‘Set-based’ data
  - Multiple results returned
  - Retrieving rows

**XML**

- **Strength:** Semi-structured, frequently changing data
  - Self-describing, flexible schema
  - Easily modified format

- **Strength:** Retrieving sequences
  - Documents, subdocuments, related documents

*Over $20B Annual Customer Technology Investment in RDB Alone…*

*XML database investments growing twice as fast as total database investment…*
XML Data Needs Relational Maturity

Complementing XML Processing

- **XML Data Needs Protection**
  - Backup and recovery features to ensure continuity
  - Data is protected using database security

- **Simplified XML Data Access**
  - Centrally store and access difficult to retrieve data
  - SQL or XQuery can be used to retrieve data
  - Join XML data with its related relational data

- **Search Speed**
  - Search documents quickly and efficiently using proven search optimization engine of mature database

- **Optimize Existing Investments**
  - Use existing technology infrastructure and skills to store and manage both relational and XML
Is it either / or?
A New Model Is Emerging
Optimized Storage, Mature Services, Familiar Programming

XML Developer
“I see a sophisticated XML repository that also supports SQL.”

Familiar Programming Models

SQL Developer
“I see a sophisticated RDBMS that also supports XML.”

Optimized Storage Models

Mature Services

Familiar Tooling

Optimized Performance & Scale
SQL 2006 Standard: SQL+XML

- ISO/IEC 9075-14:2006 defines SQL + XML
  - Importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form.
  - Applications integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.
What Is XQuery? – A W3C Standard

A query language standard designed for XML data…
…and supported in DB2.

XQUERY

XML Schema
w3c.org/ XML/Schema

XML
w3c.org/XML

Expressions
w3c.org/TR/xquery

Functions & Operators
w3c.org/TR/xquery-operators

XPath 2.0
w3c.org/ TR/xpath20

XQuery 1.0 & XPath 2.0 Data Model
w3c.org/TR/query-datamodel
Integration of XML & Relational Capabilities

- Publish Relational Data as XML
- Decompose XML into Relational Data
- Store, update XML
- Query, index, and search, and discover with XML
- Join Relational and XML

SERVER

CLIENT

SQL/X

Application
- Object
- Relational
- XML

XQuery

Data Storage:
Relational
Data Engine

XML
Interface

Relational
Interface
XML – A First Class Citizen
A 20 foot view

- Data Definition
  create table dept(deptID int, deptdoc xml);

- Insert
  insert into dept(deptID, deptdoc) values (?,?)

- Retrieve
  select deptID, deptdoc from dept

- Select
  select deptID, xmlquery('$/dept/name' passing deptdoc as "d") from dept where deptID <> "PR27"

- XQuery
  xquery for $d in db2-fn:xmlcolumn('T.DOC')//address return $d
Native XML

- Flexibility because that is what XML is all about...
  - Any document, Any schema, Not just the ones that are mapped to relational tables

- XML storage, several options
  - Native: XML is parsed and stored hierarchical
  - Shredded: Still very useful...
  - BLOB: Sometimes it still makes sense...

- Sophisticated XML indexing
  - Join Relational and XML together using optimized indexes

- Broad XQuery support
  - Both embedded in SQL and as a primary language

- Supports Digital Signatures
  - Signatures can be validated on retrieved documents
Native XML Storage

- XML data is stored in XML-typed columns in tables

```sql
create table dept (deptID char(8), ..., deptdoc xml);
```

- XML is stored in a parsed hierarchical format

- Relational columns are stored in relational format
DB2 Native XML Storage

- Node hierarchy of an XML doc stored on DB2 pages
  - Large documents split into pages/regions

- Nodes are physically connected
  - Query performance

- Regions are logically connected
  - Regions index is a system component

- XML validation is always optional
- Validate against any XML Schema
  - Flexibility

- Parse-once paradigm:
  - Documents XML-parsed at insert time only (never at query time)
  - Query performance
Publish with XQuery: The 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 xmlcolumn('movies')
LET $actors := $movie//actor
WHERE $movie/duration > 90
ORDER by $movie/@year
RETURN <movielist>
    {$movie/title, $actors}
</movielist>
```

```
<movie year="2002">
    <title>Chicago</title>
    <duration>92</duration>
    <actor>Renee Zellweger</actor>
    <actor>Richard Gere</actor>
    <actor>Catherine Zeta-Jones</actor>
</movie>
```
XQuery Access to XML Values

- **Identifying XML data by column**
  FOR $d$ in `db2-fn:xmlcolumn("DEPT.DOC")`… always operates on entire column!

- **Identifying XML data via a select statement**
  Leverage predicates/indexes on relational columns
  
  FOR $d$ in `db2-fn:sqlquery("select doc from dept")`…
  FOR $d$ in `db2-fn:sqlquery("select doc from dept where deptID = 'PR27' ")`…
  FOR $d$ in `db2-fn:sqlquery("select doc from dept where deptID LIKE 'PR%' ")`…
  FOR $d$ in `db2-fn:sqlquery("select dept.doc from dept, unit
  where dept.deptID=unit.ID and unit.headcount > 200")`…
XMLQUERY: Embed XQuery in SQL

```sql
create table dept(deptID char(8) primary key, doc xml);
create table unit(unitID char(8), headcount integer, bldg integer);

select deptID, xmlquery('for $d in $doc/dept
    where $d/@bldg = 101
    return $d/name' passing doc as "doc")
from dept
where deptID <> "PR27";

select d.deptID, u.headcount,
    xmlquery('$doc/dept/name' passing d.doc as "doc")
from dept d, unit u
where d.deptID=u.unitID
and u.headcount > 200
and xmlexists('$doc/dept/employee[@name = "Joe"]' passing d.doc as "doc")
```
XMLTABLE: Make a Table from XML

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

Stephen A. Brodsky — XML and Relational Database Integration
Publish with SQL/XML: Use SQL to Produce XML from Relational Data

```
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;
```

Start With

Available Functions:
- XMLELEMENT
- XMLATTRIBUTES
- XMLFOREST
- XMLCONCAT
- XMLAGG
- XML2CLOB
- XMLNAMESPACES
- XMLCAST

Produce

```
<department name="A00">
    <emp>CHRISTINE</emp>
    <emp>VINCENZO</emp>
    <emp>SEAN</emp>
</department>

<department name="B01">
    <emp>MICHAEL</emp>
</department>

<department name="A00">
    <emp>LEE</emp>
    <emp>SEAN</emp>
    <emp>BARELLI</emp>
</department>

<department name="B01">
    <emp>JOHNSON</emp>
    <emp>MICHAEL</emp>
</department>

<department name="A00">
    <emp>SMITH</emp>
    <emp>CHRISTINE</emp>
</department>
```

<table>
<thead>
<tr>
<th>firstname</th>
<th>lastname</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEAN</td>
<td>LEE</td>
<td>A00</td>
</tr>
<tr>
<td>MICHAEL</td>
<td>JOHNSON</td>
<td>B01</td>
</tr>
<tr>
<td>VINCENZO</td>
<td>BARELLI</td>
<td>A00</td>
</tr>
<tr>
<td>CHRISTINE</td>
<td>SMITH</td>
<td>A00</td>
</tr>
</tbody>
</table>
New SQL/XML Functions in SQL 2006

- **XMLPARSE** parses character/BLOB data, produces XML value
- **XMLSERIALIZE** converts an XML value into character/BLOB data
- **XMLVALIDATE** validates XML value against an XML schema and type-annotates the XML value
- **XML_EXISTS** determines if an XQuery returns a results (i.e. a sequence of one or more items)
- **XMLQUERY** executes an XQuery and returns the result sequence
- **XMLTABLE** executes an XQuery, returns the result sequence as a relational table (if possible)
- **XMLCAST** cast to or from an XML type
XML Schema Repository (XSR)

- Defines structure, content, data types of XML documents
- XML Schemas are used to "validate" an XML document

- Register XML Schemas with XSR

  \[
  \text{REGISTER XMLSCHEMA} \ 'http://myOrderSchema/order.xsd' \\
  \text{FROM} \ 'file://c:/TEMP/order.xsd' \\
  \text{AS} \ \text{user1.myOrderSchema} \ \text{COMPLETE}
  \]

- Validation is always optional
- XMLValidate is the only way to validate

  \[
  \text{insert into orders(doc) values (xmlvalidate(?))} \\
  \text{insert into orders(doc) values (xmlvalidate(?
  \text{according to xmlschema id user1.myOrderSchema))}
  \]
Text Search

- Full XML-aware index of entire doc
  - Create index myIndex for text on T (doc) format xml

- Simple usage
  - select i, doc from T where contains (doc, `sections("/a/b") "zero"`) = 1
  - Complex search criteria also supported

- Is not transactional
  - Manual or scheduled updates
  - Update index myIndex for text
Accessing XML with JDBC
JDBC Review

- **Connection** – open a connection to the database
  
  ```java
  connection = DriverManager.getConnection(url, user, pass);
  ```

- **Statement** – the SQL to execute
  
  ```java
  PreparedStatement stmt = connection.prepareStatement(sql);
  ```

- **ResultSet** – the results of execution
  
  ```java
  ResultSet resultSet = stmt.executeQuery();
  ```

- **Stream** – an XML value
  
  ```java
  InputStream inputStream = resultSet.getBinaryStream(1);
  ```

- **or DB2XML or SQLXML** – an XML value
  
  ```java
  DB2Xml db2xml = (DB2Xml) resultSet.getObject(1);
  ```
Example – Select

- Select an XML product from the database

```java
String sql = "SELECT PID, DESCRIPTION from XMLPRODUCT where PID = ?";
PreparedStatement stmt = connection.prepareStatement(sql);
stmt.setString(1, "100-105-09");
ResultSet resultSet = stmt.executeQuery();

String xml = resultSet.getString("DESCRIPTION"); // or
InputStream inputStream = resultSet.getBinaryStream("DESCRIPTION"); // or
Reader reader = resultSet.getCharacterStream("DESCRIPTION");
```
Example – Insert

- Insert an XML product from the database from a file

```java
String sql = "INSERT INTO xmlproduct VALUES(?, ?)"
PreparedStatement stmt =
    connection.prepareStatement(sql);
stmt.setString(1, "100-105-09");
File xmlFile = new File("productBinIn.xml");
InputStream inBin = new FileInputStream(xmlFile);
stmt.setBinaryStream(2, inBin, (int) binFile.getLength());
stmt.execute();
```
JDBC and XML Coming Soon
(JDBC v4) (JSR 221)

- SQLXML object added to JDBC specification
  - getSQLXML() returns SQLXML object
  - getObject() returns SQLXML object
  - SQLXML objects are also DB2XML objects – full compatibility

- Represents an XML value

- Similar to BLOB/CLOB

- Access from a ResultSet
  - SQLXML getSQLXML(int columnIndex)
  - SQLXML getSQLXML(String columnName)
  - void updateSQLXML(int columnIndex, SQLXML xmlObject)
  - void updateSQLXML(String columnName, SQLXML xmlObject)
### SQLXML Example

- Access the XML value of the SQLXML instance

```java
ResultSet resultSet = stmt.executeQuery();
SQLXML sqlxml = resultSet.getSQLXML(column);
- String xml = sqlxml.getString("DESCRIPTION"); // or
- InputStream inputStream = sqlxml.getBinaryStream("DESCRIPTION"); // or
- Reader reader = sqlxml.getCharacterStream("DESCRIPTION");
```
SQLXML Methods

- `get/setBinaryStream()` – InputStream/OutputStream
- `set/setCharacterStream()` – Reader/Writer
- `get/setString()` – String
- `getSource()/setResult()` – uses the JAXP Source/Result
  - DOM – DOMSource / DOMResult
  - SAX – SAXSource / SAXResult
  - StAX – StAXSource / StAXResult
  - Stream – StreamSource / StreamResult
Java's XML APIs

- **DOM – Document Object Model – JDK 1.4**
  - An object tree that represents an XML document
  - Most common XML API in use
- **SAX – Simple API for XML – JDK 1.4**
  - A series of events generated while a parser processes XML
  - Faster than DOM -> Build your own object tree
- **StAX – Streaming API for XML – JDK 1.6 / JSR 173**
  - Pull XML events from a parser
  - Similar to SAX but newer and simpler
  - Adds a streaming writer API
- **XSLT – XML Stylesheet Transform – JDK 1.4**
  - Transform XML formats from **Source** to **Result**
  - Transform XML from DOM to SAX to StAX to Stream and back
Source and Result Examples

- **DOM get a Document**
  - DOMSource domSource = resultSet.getSQLXML(1).getSource(DOMSource.class);
  - Document document = (Document) domSource.getNode();

- **DOM set a Document**
  - DOMResult domResult = resultSet.getSQLXML(1).setResult(DOMResult.class);
  - domResult.setNode(myNode);

- **Run an XSLT on an XML result**
  - File xsltFile = new File("my.xslt");
  - File myFile = new File("result.xml");
  - Transformer xslt = TransformerFactory.newInstance().newTransformer(new StreamSource(xsltFile));
  - Source source = resultSet.getSQLXML(1).getSource(null);
  - Result result = new StreamResult(myFile);
  - xslt.transform(source, result);
Source and Result Background

- JAXP general approach to unifying the many ways to access XML information
- Source and Result are Java's strategic XML direction
- Source and Result represent in memory or stream XML (i.e. InputStream, Reader, DOM, SAX ContentHandler, etc.)
- Transform converts a Source to a Result
  - transform(source, result)
- Transforms built into Java include XSLT
- Enables optimizations for database directly to/from Source/Result by avoiding parsing on the client and server
  - Could convert high speed XML format directly to/from DOM or SAX
create table dept (id char(8), deptdoc xml)

```java
PreparedStatement stmt = con.prepareStatement(
    "Select deptdoc from dept where id='001'");
ResultSet rs = stmt.executeQuery();
rs.next();
XSDHelper.INSTANCE.define("<schema ....> ");

XMLHelper xmlHelper = XMLHelper.INSTANCE;
InputStream stream  = rs.getBinaryStream("deptdoc");
    // Load XML stream into SDO XMLDocument
XMLDocument deptDoc = xmlHelper.load(stream);
DataObject dept = xmlDoc.getRootObject();
int salary = dept.getInt("employee[SN='123']/salary");
```
The following slides are from Microsoft’s September 2005 Professional Developers Conference, as posted on their web site.
MS XQuery And XML-DML In SQL Server 2005

- Subset of XQuery implemented
- Is aligned with July 2004 XQuery working draft
- Added XML Data Modification
- Applies to single XML data type instance
- Methods on XML data type:
  - query(), value(), exist(), modify(), nodes()
- Use SQL to iterate over collection of instances (XML-typed column)
- Can refer to relational data
- Take advantage of Schema-collection information to operate on typed XML data
- Will make use of XML indices for optimization
Map SQL value and type into XQuery values and types in context of XQuery or XML-DML

**sql:variable()**: accesses a SQL variable/parameter

```
declare @value int
set @value=42
select * from T
where T.x.exist(''/a/b[@id=sql:variable("@value")]')=1
```

**sql:column()**: accesses another column value

```
tables: T(key int, x xml), S(key int, val int)
select * from T join S on T.key=S.key
where T.x.exist(''/a/b[@id=sql:column("S.val")]')=1
```

**Restrictions in SQL Server 2005:**

- No XML, CLR UDT, datetime, or deprecated text/ntext/image
MS XML Indices

- Create XML index on XML column
  ```sql
  CREATE PRIMARY XML INDEX idx_1 ON docs (xDoc)
  ```
- Create secondary indexes on tags, values, paths
- Speed up queries
  - Results can be served directly from index
  - SQL’s cost based optimizer will consider index
- Primary and Secondary Indices will be efficiently maintained during updates
  - Only subtree that changes will be updated
MS Architectural Blueprint: Indexing

XML Column in table T(id, x) → Primary XML Index (1 per XML column) Clustered on Primary Key (of table T), XID

<table>
<thead>
<tr>
<th>id</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Binary XML</td>
</tr>
<tr>
<td>2</td>
<td>Binary XML</td>
</tr>
<tr>
<td>3</td>
<td>Binary XML</td>
</tr>
</tbody>
</table>

Non-clustered Secondary Indices (n per primary Index)

- Value Index
- Property Index
- Path Index
Take-Away: XML Indices

- **PRIMARY XML Index** – use when lot’s of XQuery

- **FOR VALUE** – useful for queries where values are more selective than paths such as `//*[.=‘Seattle’]`

- **FOR PATH** – useful for Path expressions: avoids joins by mapping paths to hierarchical index (HID) numbers. Example: `/person/address/zip`

- **FOR PROPERTY** – useful when optimizer chooses other index (e.g., on relational column, or FT Index) in addition so row is already known
Summary

- SOA makes IT flexibility possible
- SOA is built on XML and Web Services
- XML allows enterprises to implement highly standardized service descriptions and message structures
- XML and SQL integration standardized in the SQL 2006 specification
- XML is now in the core of the Database tier
Summary (Continued)

- Native XML support with more to come
- Seamless integration with the relational world

And all the relational stuff
DB2 Viper Downloads

http://www.ibm.com/db2/viper
Free Resources

IBM developerWorks Tutorials and Training
- Tutorials and training
- over 36 titles for IBM Rational software and more titles being added all the time: http://ibm.com/developerworks/training

Downloads
- Easy access to IBM trial software: http://ibm.com/developerWorks/downloads

2005 developerWorks Software Evaluation Kit
- Over 14GB of the latest trial software, both development and testing tools as well as middleware on DVD: http://ibm.com/developerWorks/offers/sek