DB2 pureXML - History, Features, Details

Jena, July 9th, 2009

Dr. Henrik Loeser
DB2 for Linux, UNIX, and Windows
Agenda

- A Look Back
- Status Quo
- What's next?

DB2 pureXML

pureXML Storage
pureXML Indexing
XQuery
SQL/XML

DB2 pureXML
Beginning of XML

- 1960s: IBM's Generalized Markup Language (GML, Goldfarb/Mosher/Lorie)


- Mid-1990s: Work started on XML

- February 10th, 1998: XML 1.0

- February 4th, 2004: XML 1.1
XML Data is Everywhere

- XML-based data exchange

- SOA

- Industry Standards: XBRL, FpML, FIXML, HL7, ACORD, MISMO, UNIFI, SVG, NewsML, STAR, OTA, …

- Applications with need for flexibility
XML Usage

- Document size?
- How many?
- Workload mix?
- Query complexity?
- Validation?
What is a XML Database?

<xmlldb>
  <features>
    <f id="1">defines (logical) model for XML document</f>
    <f id="2">XML document as unit of storage</f>
    <f id="3">store data optimally for querying (XPath, XQuery) and flexibility</f>
  </features>
  <!-- not much, but hard to do -->
</xmlldb>
XML Database Types

- Native / XML only
- XML-enabled
- Hybrid
The Early Players

- 1999: eXcelon (Object Design Inc., ObjectStore)
- 2000: Tamino (Software AG)
- 2001: DB2 XML Extender
pureHistory

- "Native XML in DB2" project started in 2001
- Moved from AdTech to regular DB2 development in 2003
- "pureXML 1.0" as part of DB2 9 in 07/2006
- XQuery 1.0 – 01/2007
- "pureXML 2.0" as part of DB2 9.5 in 10/2007
- "pureXML 3.0" as part of DB2 9.7 in 06/2009
pureXML in DB2

SQL Person... "I see a sophisticated RDBMS that also supports XML"

XML Person... “I see a sophisticated XML repository that also supports SQL"

XML integrated in all layers of DB2!

New XML applications benefit from:
• Ability to seamlessly leverage relational investment
• Proven Infrastructure that provides enterprise-class capabilities
Relational Data vs. XML Data

- Flat
- Needs metadata
- Unordered
- Regular, Homogeneous
- Hierarchical, nested
- Self-describing
- Order of elements important
- Extensible, heterogeneous

➡ New query language for XML data needed
What is XQuery?

- A query language designed for XML data sources
- Developed by a W3C Working Group over five years
- Became "Recommendation" January 23rd, 2007
XQuery on One Slide

- Prolog: declares namespaces and sets up the query environment
- Composable expressions:
  - Literals & variables
  - Function calls
  - Path expressions
  - Predicates
  - Arithmetic
  - Comparisons
  - and, or
  - Union, intersect, except
  - Constructors
  - FLWOR expressions
  - if ... then ... else
  - some, every
  - cast
  - XQuery Update Facility
SQL/XML on one slide

- Part 14 of ISO/IEC 9075
- XML as SQL data type

- XMLELEMENT
- XMLATTRIBUTES
- XMLCOMMENT
- XMLPI
- XMLDOCUMENT
- XMLFOREST
- XMLCONCAT
- XMLAGG
- XMLQUERY
- XMLEXISTS
- XMLTABLE
- XMLSERIALIZE
- XMLPARSE
- XMLVALIDATE
- IS VALIDATED
- XMLNAMESPACE
- XMLCAST
Putting it together

SELECT XMLELEMENT(NAME "purchaseOrder",
    XMLNAMESPACES(DEFAULT 'http://555gadgets.biz/po'),
    XMLCOMMENT('coded by HL'),
    XMLCONCAT(XMLFOREST(o.poid AS "poid",o.odate AS "date"),
    XMLELEMENT(NAME "customer",
        XMLELEMENT(NAME "id"),
        XMLELEMENT(NAME "name", c.name)),
    (SELECT XMLELEMENT(NAME "items",
        XMAGG(XMELIST(NAME "item"),XMLATTRIBUTES(p.id AS "id"),
        XMELIST(NAME "quantity",i.quantity))))
FROM products p,orderitems l
WHERE i.pid=p.id
GROUP BY i.poid),
XMLQUERY(('[for $i in $l//DidYouReallyReadSoFar let $j:=(<hi>Still not
tired?</hi>) where $i mod 100=0 return <end>{$j}</end>}' PASSING l.logs AS "l")
FROM customers c,orders o, logs l
WHERE o.cid=c.id
    AND XMLQUERY(('[@status="reviewed"]/details/city
[contains(.,"San Jose")]' PASSING l.logs AS "l")
WHERE $i mod 100=0 return <end>{$j}</end>' PASSING l.logs AS "l")
Putting it together

```
SELECT XML blah blah
  XML blah blah
XML blah (XML blah blah),
XML blah blah
XML blah blah
 (SELECT XM blah blah
XML blah(XML blah blah
XML blah blah
XML blah blah
FROM products p,orderitems I
WHERE i.pid=p.id
GROUP BY i.poid)),

XML blah
blah
 blah
FROM customers c,orders o, logs l
WHERE o.cid=c.id AND XML blah blah
blah blah
blah
```
DB2 – SQL and XQuery Execution

SQL Statement → SQL Parser

XQuery Statement → XQuery Parser

SQL Parser → DB2 Compiler / Optimizer → QGM

QGM → DB2 Package
The XML Data Type

- Table with XML- and non-XML columns

```sql
CREATE TABLE dept (deptID CHAR(8), ..., deptdoc XML);
```

- Rows hold relational data
- XML values in pureXML (XDM) storage
- Descriptor pointing from row to XML storage
XML Storage

- Documents are stored in parsed representation
  - IBM’s version of open-source Xerces is used.
  - The XQuery Data Model (XDM) is persisted
  - Each document consists of many nodes.

- All data is stored in UTF-8, regardless of the document encoding

- All XML nodes are type annotated, according to the XQuery specification
  - XML Schema types if validated.
  - Default types otherwise.
Use of stringIDs/pathIDs

- NCNames (attribute, element, nsURI, prefix) replaced with 32 bit IDs (stringIDs)
  - Compresses documents
  - Enables faster processing
- `<a:name>`Henrik`</a:name>`
- `<a:name>` → 200:417

- stringIDs used in many places
  - XQuery compiler, runtime, storage, indexing

- stringIDs are used to encode paths
  - Paths are replaced with pathIDs
  - XML Column Path index can be used for wildcard resolution, estimations, etc.
The Document Tree

```
<dept>
  <employee id="901">
    <name>John Doe</name>
    <phone>408 555 1212</phone>
    <office>344</office>
  </employee>
  <employee id="902">
    <name>Peter Pan</name>
    <phone>408 555 9918</phone>
    <office>216</office>
  </employee>
</dept>
```

Each node has a path.
Strings Table & Paths Table

- String & Path table per database
- Database wide dictionary...
- ...for all documents in all XML columns

Path table

<table>
<thead>
<tr>
<th>Path table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Namespaces are omitted for clarity

SYSIBM.SYSXMLPATHS

SYSIBM.SYSXMLSTRINGS

...new in the DB2 catalog.
XML Storage

DAT Object

<table>
<thead>
<tr>
<th>ID</th>
<th>DEPTDOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR27</td>
<td></td>
</tr>
<tr>
<td>PR28</td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td></td>
</tr>
</tbody>
</table>

INX Object

Region

Path

/dept
/dept/employee
/dept/employee/@id
...

XDA Object
The Xml Data Object

- Xml Data (XDA) object
  - Holds XML data pages ("XML Storage")
  - Logically pointed to from data descriptor stored in the row – resolved through the Region Index.
DB2 Bufferpool

All data pages, including XDA (CSL) pages go through the bufferpool

<table>
<thead>
<tr>
<th>ID</th>
<th>…</th>
<th>DOC (XML)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR27</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>PR28</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td>…</td>
<td></td>
</tr>
</tbody>
</table>
pureXML Processing

- Relational processing: by value
  - Data copied from disk
  - Carried thru the system

- XML processing: by reference
  - Only descriptors are moved throughout the system
  - Usually beneficial because of document size
  - Temporary results need to be "temp'ed" to have reference
New System Indexes

- Entries in SYSCAT.INDEXES with the following INDEXTYPE
  - XRGN: XML Region Index
    - Created once for table with XML column(s)
    - Maps logical pointers to XML data pages
  - XPTH: XML Path Index
    - Created for each XML column
    - Holds local subset of global path/pathID mapping information / path table
    - Can be used for wildcard resolution
XML Node Storage Layout

- Node hierarchy of an XML doc stored on DB2 pages
- Documents that don’t fit on 1 page: split into pages/regions
- No architectural limit for size of XML documents
- NodeIDs used to identify individual nodes (1.2.4.3)

Example:

Document split into 3 regions, stored on 3 pages
XML Data – as Trees on DB2 pages

All benefits of DB2 tablespaces:

- Buffered in Buffer Pools
- Prefetching
- Logging & Recovery
- etc.
Index over XML Data

- Specify the XPath and Data Type

- `CREATE INDEX idx1 ON myTable(doc) GENERATE KEY USING XMLPATTERN '/account/balance' AS SQL DOUBLE REJECT INVALID VALUES`

- B-Trees with special keys

- Multiple (hundreds) of indexes possible on single XML column
How is an XML index different?

- Index elements/attributes inside the document
  - 0, 1, or multiple index entries per document
  - 0 if missing elements/attributes
  - Multiple if repeating elements

- XMLPATTERN '/*siblings*/name'
Three Conditions for XML Index Eligibility

- An XML index can be used for a query predicate if...
  
1. Index “contains” the query predicate, i.e. is equally or less restrictive than the predicate

2. Query predicate matches the index data type

3. /text() is used consistently in predicates and index definitions
Benchmarks

- XMach-1, XMark, XPathMark, XOO7, XBench, MBench, ...
- TPoX (Transaction Processing over XML)
  http://tpox.sourceforge.net
What's next?

- Extended Usage
  - Paradigm shift
  - XML moving into more domains
  - Data volume
What's next?

- Performance, Scalability
  - Parsing Technology
  - Integration between application and database layer
  - ...

What's next?

- Tooling, APIs
- Closing functional gaps in XQuery and SQL
- Best Practices, Simplification
Conclusions

- XML has old roots
- XML Databases already decade "old" technology, but still young
- XML databases still in early usage phase

Exciting field to work in
(and so is Information Management in general)