hybrid applications using relational and xml together

Bob Beauchemin

Outline

- **Data Type Choices**
  - XML in a Relational Database
    - Semi Structured Data
    - Unstructured Data
  - Relational Data and XML
  - Blob Storage
    - Large Value Types and Filestream
- **Hybrid XML-Relational Apps**
  - Composition and Decomposition
    - SELECT... FOR XML
    - OpenXML and xml.nodes
  - Mixing XQuery and Full-Text Search

XML in a relational database

- **XML can be stored as text**
  - loses much of value of XML representation
    - storing XML documents with lexical integrity
    - no query, transform without casting to XML type
- **XML can decomposed into multiple relational tables**
  - allows use of relational technologies
    - T-SQL programmer familiarity
    - better integration with rest of the tables
    - ease of query
    - document order not maintained
- **XML can be stored as an xml data type**
  - allows use of XML technologies
    - XML programmer familiy
    - can join using XQuery value method
    - maintains document order

Data Type Options

- **The relational data types serve enterprise applications well but...**
  - There's always been a tension with large data
    - In database or file systems?
  - XML becoming common for all industries
    - In B2B, B2C, data exchange
    - XML is a standard for data on the web
    - To evolve and integrate your business you may need to support XML
  - Domain-specific types used by some industries

Unstructured Data

- **Byte stream**
  - Office documents
  - Images (bitmap, PNG, JPEG etc)
  - Compiled binaries
- **DB engine has no intrinsic understanding of semantics or internal structure of the stream.**
- **Two database categories for unstructured data**
  - CLOB: Character large “objects”, byte stream with collation/codepage information
  - BLOB: Binary large “objects”, just byte stream
- **Operations on them through extensions to SQL**
  - FullText using a specialized filter
  - User-defined functions

Semi-structured Data

- **Self-describing data**
  - Metadata and data in same data instance
  - No explicit schema
  - Extensible
- **Heterogeneous data**
  - Structure varies from instance to instance
  - Structure varies over time
- **Sparse data**
  - One-off annotations, Rare properties
- **Markup data**
  - documents/content markup (XHTML, WordML etc.)
  - Explicit document order
- **Schema/Type system**
  - Schema-less
  - Optional Schema: semi-structured

XML Characteristics

- **Self-describing**
- **Complex data**
  - Trees, recursive, graph
  - Structured Data: highly regular, homogeneous structures
  - Semi-structured Data: heterogeneous, sparse data
  - Markup Data: documents/content markup
- **Document ordering**
- **Schema/Type system**
  - Schema-less, semi-structured, structured
- **Extensible**
  - Annotations, multiple schemas

XML Scenarios

- **Data Exchange**
  - Business to business (B2B), business to consumer (B2C), application to application (A2A)
  - XML is ubiquitous, extensible, platform independent transport format
- **Document Management**
  - XHTML, Office XML Documents
- **Messaging**
  - Simple Object Access Protocol (SOAP), REST
- **Mid-Tier Collaboration**
- **Ad-hoc modeling**
  - storing objects with sparse and multi-valued properties that do not fit well in the traditional relational schemata

Data Type Enhancements (SQL Server 2005, 2008)

- **Relational is native for SQL Server**
  - Third or fifth normal form schema design preferred
  - Some designs are difficult in 3NF
    - "Open-schema" helped by PIVOT, sparse cols
      - Assists sparse population & name-value pair queries
    - Hierarchy support using XML, hierarchyid type, adjacency, and nested sets
      - XML and hierarchyid type both use ORDPATH
      - or query adjacency model with recursive common table expressions
- **XML is a built-in alternative**
  - Through XML data type, schema, and query
  - XML values can be promoted to persisted computed columns with xml.value
- **Large value type support is better since SQL Server 2005**
  - MAX data types subsume TEXT and IMAGE
  - Filestream allows file system storage
- **Custom types and aggregates available**
  - Through SQLCLR UDT for custom scalars
    - Hierarchyid, spatial are built-in types implemented through SQLCLR
  - Through SQLCLR custom aggregates

Large value data types

- **Using TEXT or IMAGE data type had limitations**
  - not allowed as procedure variables
  - not updateable directly
  - most varchar functions don't work on TEXT
- **Large variable-length data types don't have these limits**
  - VARCHAR(MAX), NVARCHAR(MAX)
    - support all SQL functions that VARCHAR supports
    - this is NOT just variation of VARCHAR
  - VARBINARY(MAX)
  - new types preferred over TEXT and IMAGE

Filestream storage

- **Storing large binary objects in databases is suboptimal**
  - Large objects take buffers in database memory
  - Updating large objects cause database fragmentation
    - In file system however, "update" is delete and insert
    - "Before image" in an update is not deleted immediately
- **Data over 1mb should be stored in file system**
  - Break-even point is between 256kb – 1mb
- **Storing all related data in a database adds**
  - Transactional consistency
  - Integrated, point-in-time backup and restore
  - Single storage and query vehicle
- **Enter filestream storage...**

Filestream Implementation

- **A filegroup for filestream storage is declared using DDL**
  - Filestream storage is tied to a database
- **The filegroup is mapped to a directory**
  - Must be NTFS file system
  - Caution: Files deleteable from file system if you have appropriate permissions
- **VARBINARY(MAX) columns can be defined with FILESTREAM attribute**
  - Table must also have UNIQUEIDENTIFIER column
  - Filestream storage not available for other large types (no nvarchar(max) or XML data type)
- **Data is stored in the file system**
  - Transactionally manipulated through T-SQL or T-SQL + File IO

Data Types and Applications

Scenarios	XML	BLOB	Relational
Relational Data Exchange	Use as transport, shred to relational	-	Storage and Query
Document Management	Use as markup, store natively if query, update or validation needed	Store natively (even XML if no query, update, or validation needed)	Provides framework to manage collections and relationships; provides Full-text search
Semi-structured Data	Represent semi-structured parts	-	Represent structured parts
Message audit	Store natively	Store natively (if part of message)	Use for querying over promoted properties
Object serialization	Use for accessing structure and interoperability. Store natively	Use for "dumb", efficient storage.	Use for querying over promoted properties

Hybrid XML-Relational Apps

- **You can create hybrid apps by**
  - Using composition and decomposition
    - SELECT ... FOR XML
    - xml.nodes
    - OpenXML
  - Extracting often-used element and attribute values to computed columns
  - Using XQuery in T-SQL statements
  - Using XML in middle-tier or client (OData)
  - Using System.Xml in .NET procs
  - Combining XQuery and Full-Text Search
  - Using SOAP for messaging
    - SQL Server 2005 introduced SOAP as adjunct to TDS protocol
      - SOAP operations mapped to stored procedure invocation
    - Deprecated in SQL Server 2008

Composition and Decomposition

- XML data can be stored as relational (decomposition)
- Relational data can be delivered as XML (composition)

SELECT...FOR XML

- **SELECT FOR XML is an extension to Transact-SQL**
  - usually produces a stream
  - in SQL Server 2005, it can also produce an XML data type
    - use TYPE keyword after FOR XML
    - can be used to query a "collection" of documents

```
DECLARE @x xml
SET @x = SELECT * FROM authors
    FOR XML AUTO, TYPE
```

FOR XML PATH

- **SELECT FOR XML PATH allows shaping of output**
  - FOR XML AUTO, RAW allow little shaping
  - FOR XML EXPLICIT is complex to write
  - FOR XML PATH uses path-syntax in column aliases
    - allows "creation" of elements
    - simpler than XML EXPLICIT
    - can specify namespaces with XMLNAMESPACES function

XML Schema and FOR XML

- **XMLSCHEMA keyword prepends XML Schema to FOR XML**
  - can be used with other keywords
  - schema namespace can be specified
  - standard schema for SQL Server types
    - http://schemas.microsoft.com/sql/2004/types
  - FOR XML AUTO and FOR XML RAW only
  - produces typed XML if TYPE used
    - schema must have been defined using CREATE XML SCHEMA COLLECTION

```
DECLARE @x xml
SET @x = (SELECT * FROM authors
 FOR XML AUTO, TYPE, XMLSCHEMA('urn:authors')
```

More FOR XML Enhancements

- **Many extensions to FOR XML in 2005**
  - element-centric XML using FOR XML RAW
  - generate xsi:nil for NULL database values
    - FOR XML AUTO/RAW, ELEMENTS XSINIL
  - nested FOR XML queries
  - can specify ROOT element
  - can name <row> element in FOR XML RAW
  - changes to nesting algorithm in XML AUTO
  - supports new data types
    - varchar(max), nvarchar(max), varbinary(max)
    - UDTs (must be cast/converted to XML in query)
  - WITH XMLNAMESPACES

XML Data Type and OpenXml

- **OpenXml is a SQL Server system function**
  - originally used (N)VARCHAR or (N)TEXT input
  - available since SQL Server 2000
  - used with sp_xml_preparedocument
  - permits user-defined fill into existing tables
  - no performance problem with parent axis
  - convenient overflow column for any XML not matching column predicates
- **xml.nodes method usually preferred after 2005**
  - best choice for performance and memory utilization
- **OpenXml function can use XML data type (2005 and Above)**
  - still must be parsed with sp_xml_preparedocument
  - overflow column may be xml data type

Using XML through SQLCLR

- **SQLCLR sprocs and functions can use XML**
  - Any code in System.Xml is available
- **SQL Server 2008 adds LINQ to XML**
  - System.Core.dll, System.Xml.Linq available as "approved" assemblies
- **XML data type maps to System.Data.SqlTypes.SqlXml**
  - Use CreateXmlReader static method

Full-Text Indexing and XML

- **SQL Server 2005 adds XML filter**
  - Index and query XML instances
    - Markup serves as token boundary
    - Markup tags (including attributes) removed
  - Syntax - same as for other columns
    - CREATE FULLTEXT INDEX ON docs (xDoc)
  - Can be combined with XQuery:
    - Use full-text search as filter, then XQuery search
    - Uses full-text index first
    - Uses XML index on tags, values, paths
    - FTS contains is not the same as XQuery contains

Scenario for XML Development

- **Good Scenario:**

  - Data is semi-structured, small core of fixed data with many, sparsely populated extended attributes
    - Multi-value Property bags
    - Complex Property bags
    - "WordXML"
    - Fixed data can be stored as relational columns
  - Documents are small-medium size, but rarely updated
    → Indexing will pay off
  - Data is hierarchical
    → path expressions are well suited for finding data

- **Bad Scenario:**
  - "Database in a Cell"
  - Documents are large and/or updated frequently
  - Document update contention is likely
  - Data is fully structured & populated

    → candidate for conversion to relational schema
  - Data contains large binary objects (2GB limitation)

SQL Server Supports...

	Model	Schema	Query	Extension
Strict Relational	Tables and Relations	Relational Schema	SQL	T-SQL SQLCLR
Hierarchical	hierarchyID XML	Relational / XML Schema	SQL and HID methods	HierachyID CTE
Sparse Attribute	Tables or XML	Name/Value XML Schema	SQL or XQuery	Sparse Columns, PIVOT
Semi-structured Or Markup	XML	XML Schema	XQuery, XPath FullText	T-SQL SQLCLR
Unstructured	MAX Datatypes Filestream	IFilter	FullText	Filestream
Custom Scalars	UDT	Custom	SQL	Custom Methods

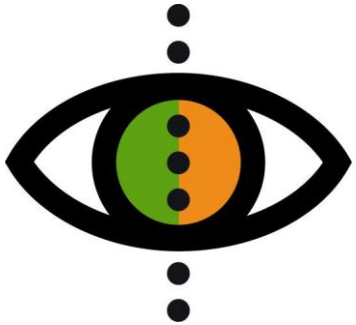
Summary

- **XML is a first class type in SQL Server**
  - Useable in columns, variables, and parameters
  - Queryable with XQuery or Full-text search
  - Use with SQLCLR and .NET XML stack
- **TEXT & IMAGE replaced by MAX data types in 2005**
- **Filestream allows file system storage**
- **SQLCLR UDTs can contain some structure**
- **Mix functionality - relational and XML**
  - Composition (SELECT...FOR XML)
  - Decomposition (xml.nodes and OpenXML)
  - Full-text search and XML
- **Best solution is based on requirements**

References

- A Developer's Guide to SQL Server 2005 - Ch 7,9 and 12, Addison-Wesley, Bob Beauchemin and Dan Sullivan
- To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem, Russell Sears, Catharine Van Ingen, and Jim Gray
- XML Best Practices for Microsoft SQL Server 2005, Shankar Pal et al., MSDN Online

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust