An Introduction to DB2 and XML

Meir Zohar, Senior Consultant
IBM Certified DBA for DB2 V8 for Z/OS
CA-Israel

Who am I

- Meir Zohar
  - 21 years in IT
  - 15 years working with DB2
  - 8 years at CA
  - Speaker at IDUG / Israel RUG (co-chairman) and Israel z/OS UG
- Small country, small market

Agenda

- An Introduction to XML
- XML and your Database
- XML and DB2
- DB2 futures – XML in vNext

<Markup> Languages </Markup>

- Humans are good at understanding words in context (sometimes …)
  - Does anyone care to guess what "*mai mai mai mai, mai ?"* means in Thai ?
  - *Mai (New) mai (wood) mai (doesn't) mai (burn), mai (does it) ?*
- And software doesn't usually understand the concept of "context"

XML – *History*

- IBM GML – General Markup Language
- SGML – Standard General Markup Language
  Document Storage for US Government Projects

  ISO 8879:1986 Information processing—Text and office systems—Standard Generalized Markup Language (SGML)
- HTML – The birth of the web – 1991
  foundation of W3C – 1994
- XML - Derived from SGML
  W3C project from 1995
  XML 1.0 Recommendation 2/98

XML – *extensible markup language*

- Multiplatform and multi-system information sharing
  - Standard extensible data formats
  - EDI, banking, eCommerce etc.
  - Enterprise Application Integration
- Intra-application data transmission
  - Flexible information transfer within the application
- Content delivery
  - Simple standard for combining data from multiple sources

XML – *What it is (and what it's not)*

- A language for defining a tree based structure to information
- Can be used as a base for more strictly defined markup languages
- Not an application programming language
- Not a DBMS

XML – *Benefits*

- Data reuse
  - Since the format is standardized – simple to repackage data into a new application (search engines)
- Separation of data and display
  - Renewal of the UI, doesn't require redesigning the data models
- Extensibility
  - Standardization of "add-ons"
  - Changes to the language do not necessarily require changing the applications
- Semantic Information
  - The context of the information can be included in the XML doc.
  - "Simple" text format, doesn't require complex development tools and training

XML – *Challenges*

- Redundant syntax
  - <challenge1>Redundant Syntax</challenge1>
- Complex Parsing
  - Recursive Nesting and duplicate definitions
- Limited Datatypes
  - Is <pi>3.1415926</pi> a number or a string ?
- Mapping to relational data structures can be complex

XML – *Concepts*

- *Well-formedness*
  - <Start> and </Start> tags
  - Valid and defined attribute names
  - Use “<“ only where allowed etc..
- Validity
  - XML documents are associated to DTD/XSD and adhering to the definitions in the DTD/XSD
- Parser checks well-formedness first, validity second

XML – *The DTD (doc type declaration/definition)*

- *Optional (but can dramatically reduce the development/maintenance effort)*
- *Simple definitions (limited syntax and validity checks)*
- *Can be internal or external*
  - Reuse common DTDs across applications
- *List of legitimate tags, attributes and entities within a document*
  - *Defines the various elements within the page and rules that control them (See the handout for a list of valid symbols and attribute types)*
  - *Defines entities (user defined constants, parameters).*
  - *Notation (external data sources)*
  - *Comments <!-- This is a comment -->*

Multiple DTDs - namespaces

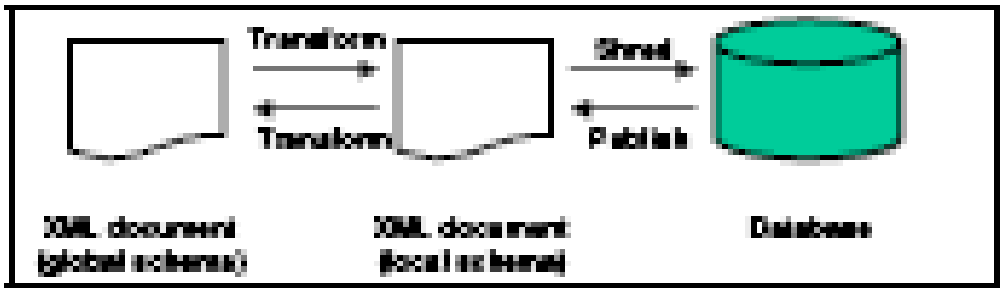
- Handling of elements with same name by context

```
<?xml version=”1.0”?>
<library-entry xmlns:authr=”authors.dtd”
                xmlns:bk=”books.dtd”>
    <bk:book>
                <bk:title>XML Sample</bk:title>
                <bk:pages>210</bk:pages>
                <bk:isbn>1-868640-34-2</bk:isbn>
    <authr:author>
                <authr:firstname>Joe</authr:firstname>
                <authr:lastname>Bloggs</authr:lastname>
                <authr:title>Mr</authr:title>
    </authr:author>
    </bk:book>
</library-entry>
```

- Add a namespace prefix to differentiate between types
- Complex definitions should be left to an XML schema

XML Schemas

- Local XML Schema used to manipulate data to/from local DBMS
- Global XML Schema used to transfer data to/from external applications
- Determining which XML schemas (local and Global) to use is an important part of the implementation (and depends on the type of application and data storage you need).



Schema Transformations

- When using both Local and Global XML schemas, the local application must transform the incoming/outgoing XML docs from the global schema to the local schema
  - Using XSLT
  - SAX Applications
  - 3rd party package
- Transformations can be expensive, so if possible, use a single schema

More XML concepts

- Xpath
  - URI (Universal Reference Indicator) notation for navigation in an XML docs (mapping the tree)
- XSL (eXtensible Stylesheet Language)
  - Output formatting information
- XSLT (transformation language)
  - Transforming data from one XML document (a source tree) into another (a result tree)

XML Applications

- Java, the natural choice
  - Portability and Unicode
  - But, can be any language with an XML parser and Unicode support
- APIs to XML
  - SAX (document access and data extraction)
    - Parser includes event handling
    - No changes to the data or underlying structure and processing is sequential only
  - DOM (Document Object Model)
    - Tree structure access
    - Data and structure creation and update
    - But .. Requires parsing the entire XML document to run

XML and your Database

- What do we need ?
  - Data storage
  - Data Access
  - Data Security
  - Utilities
  - Application Independence
  - Etc.

XML and your Database

- Can XML be used as a Database ?
  - Hierarchical Model
  - Unicode
  - Text Format
  - Schemas (DTD, XML Schemas)
  - Query languages
  - APIs
  - *So why not ?*

XML and your Database

- Can XML be used as a Database ? Why not ?
  - Inefficient
  - Verbose (need to parse every document to access the data)
  - No indexes
  - No Security
  - No Logs
  - No RI
  - *So, if you want to write your own DBMS …*

XML and your Database

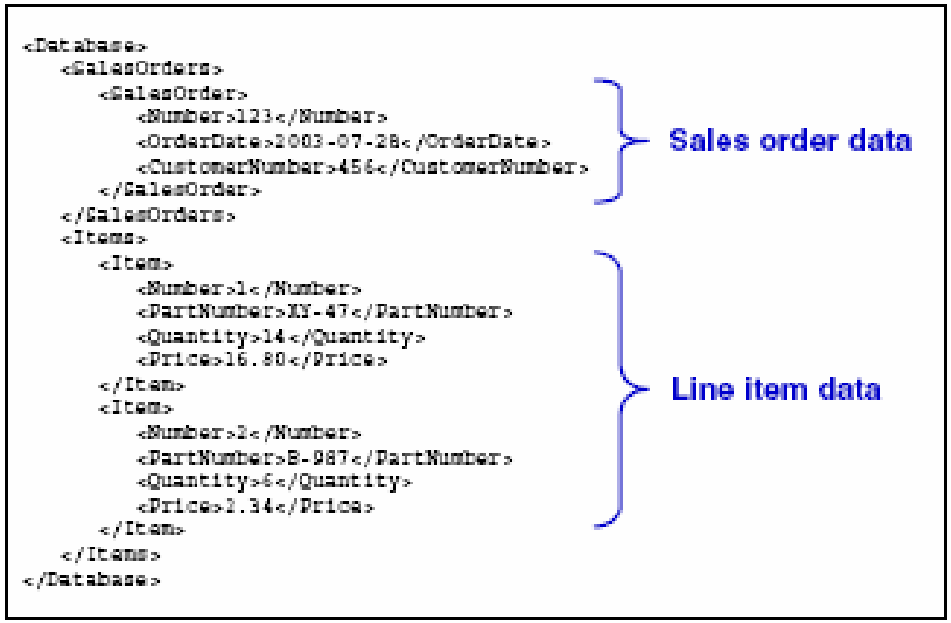
- XML enabled DB
  - Data is parsed into rows / columns (shredding)
  - Documents are constructed from rows / columns (publishing)
  - No need to change the existing data model and existing applications
- Native XML DBs
  - XML documents are stored as records in a DB
  - Data from the document can be stored on various tables according to the XML data model

XML and your Database

- XML Enabled DB – using XML to exchange data
  - No change to existing applications/data model
  - Uses DB extensions to transfer data between XML and DB data model (SQL/XML, XML Extender, XML wrapper etc.)
  - Limited to the existing design of the DB
  - Only retains data that the relational model considers important (data can be lost and the original document cannot be restored).

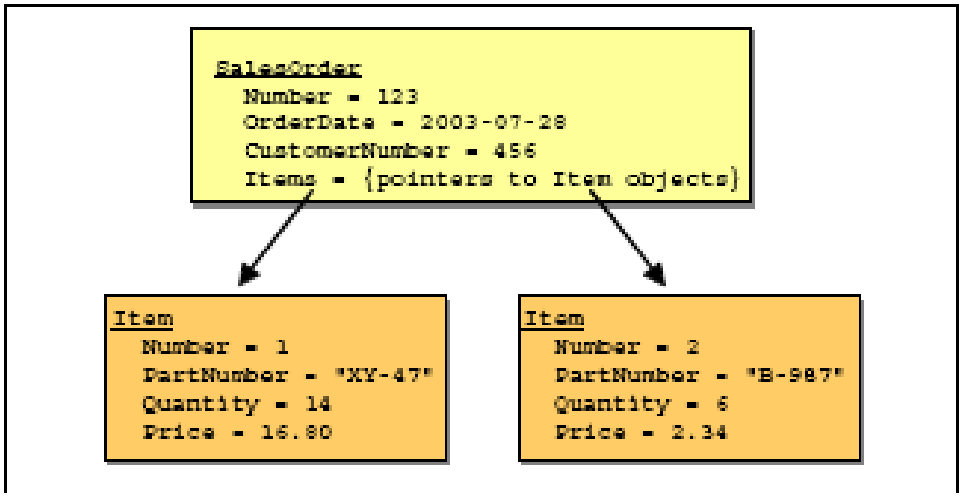
XML and your Database

- XML Enabled DB – mapping to/from XML
  - Table based mapping
    - The XML doc can be “viewed” as a set of rows in a table(s)



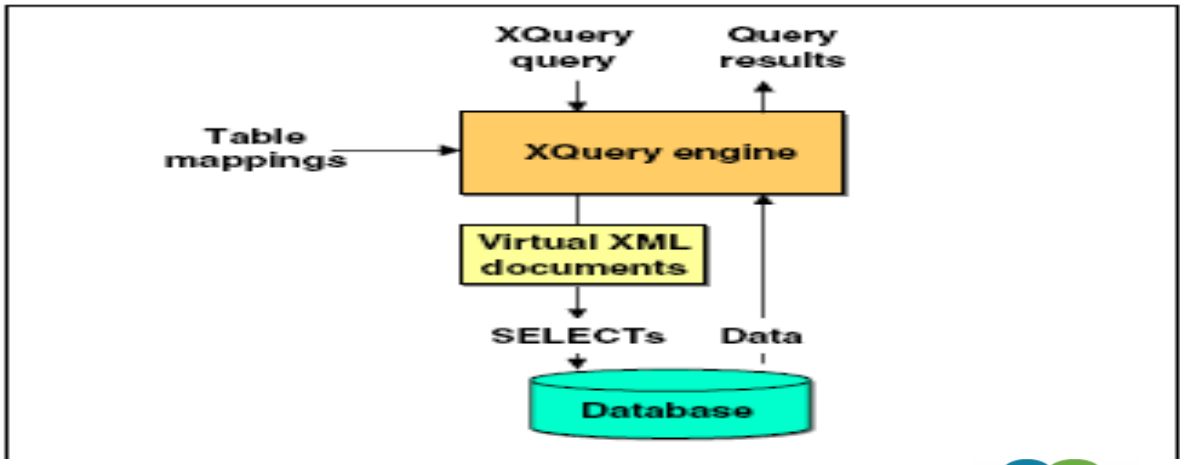
XML and your Database

- XML Enabled DB – mapping to/from XML
  - Object-relational mapping
    - The XML doc is parsed in into objects, properties and relationships



XML and your Database

- Query Languages
  - SQL/XML
    - XML extension to create XML docs from relational data
    - X/Query
      - **A query language that directly accesses XML documents**



XML and your Database

- Native XML DB
  - Defines a data model for XML (elements, attributes, text, document order etc.)
  - XML doc is the basic “logical” unit of data
    - But what is considered a doc is a design decision
    - Docs can be any size
  - Document schemas not known
    - Can be partially normalized (i.e. medical records, bank statements)
  - Difficult to model to a relational model

XML and your Database

- Native XML DB – various implementations
  - Storing complete docs in an RDBMS, indexing elements and attributes (DB2 XML extender)
    - Best for applications that query complete docs (but not for apps that update the docs)
  - Parse XML docs into fixed sets of tables retaining the logical relationships between elements
  - Parse XML docs as DOM trees in a OODBMS
  - Parse XML into indexed hash tables
- *What should you use ? It depends ….*

XML and your Database

- Native XML DB – best uses
  - Document management
  - Semi-structured data
  - Long running transactions
  - Document archival
- XML Enabled DB – best uses
  - XML used as a medium of transport
  - Context and structure insensitive data

XML and DB2

- SQL / XML
- XML Extender
- Net Search Extender
- XML Wrapper
- Websphere MQ

XML and DB2 – SQL / XML

- DB2 Internal “XML” data type
  - contains XML content (NULL, XML doc with an XML declaration or element content)
  - Publishing functionality only
- Scalar functions
  - XMLELEMENT
  - XMLATTRIBUTES
  - XMLFOREST
  - XMLCONCAT
  - Aggregation function – XMLAGG
  - XMLNAMESPACES

SQL XML is part of the ISO SQL specification (Information technology - Database languages - SQL - Part 14: XML-Related Specifications (SQL/XML) ISO/IEC 9075-14:2003).

XML and DB2 – SQL / XML

```
SELECT XML2CLOB(
        XMLELEMENT(NAME Customer, XMLATTRIBUTES(customers.id AS ID),
                XMLELEMENT(NAME Name, customers.name),
                XMLELEMENT(NAME Street, customers.street),
                XMLELEMENT(NAME City, customers.city),
                XMLELEMENT(NAME State, customers.state)))
        AS CustomerXML
FROM Customers
```

```
<Customer ID="customer id">
        <Name>customer name</Name>
        <Street>street address</Street>
        <City>city name</City>
        <State>state name</State>
</Customer>
```

XML and DB2 – DB2 XML Extender

- DB2 Extender for support of both XML enabled and native XML under DB2
  - XML enabling is provided using *XML Collections*
    - *XML Collections* are sets of tables that are mapped to from the DBMS to XML docs using DAD documents
    - Can be either SQL mapping (for select only) or RDB mapping (bi-directional XML nodes to RBD elements and vice versa)

XML and DB2 – DB2 XML Extender

- Native XML is provided using *XML Columns*
  - Special data types (XMLVARCHAR, XMLCLOB(XMLDBCLOB), XMLFILE)
  - All documents in the XML column must use the same XML Schema
  - Additional data is stored and indexed in *Side Tables* (maintained automatically by the DB) defined in a DAD document.
  - After definition the XML Column must be enabled.
- Partial update of a document based on XML column can be expensive
- Location based extraction may also be resource intensive

XML and DB2 – DB2 XML Extender

- *Side Tables*
  - Assist in indexing the contents of the XML docs
  - Must be defined when the XML column is defined (and created with the XML column is enabled).
  - Can be used to filter the returned data directly (in the WHERE clause).
  - Automatically updated on INSERT or UPDATE (invoked by triggers on the XML column).
  - If updated directly can potentially corrupt the data

XML and DB2 - Net Search Extender *

- Native XML searches
  - On text values only
  - On specific elements
  - Must be enabled for specific databases
- Requires Net Search Extender Indexes (not DB2 indexes)
  - Indexes contain a FORMAT, Code Page, Conversion Function and a document model

* DB2 LUW 8.1 and up

XML and DB2 - Net Search Extender *

- Scalar Functions
  - CONTAINS
  - NUMBEROFMATCHES
  - SCORE
- Structural Queries (on parts of the document)
- SELECT Document
  FROM OrderDocuments
  WHERE CONTAINS(Document, '"wrench"') = 1
- SELECT Document
  FROM OrderDocuments
  WHERE CONTAINS(Document,
  'THESAURUS "MechanicThesaurus“
  EXPAND SYNONYM TERM OF "wrench" |
  "Automotive Suppliers"') = 1

XML and DB2 - XML Wrapper *

- Enables users to treat native XML data as relational data
  - Shreds the document according to object-relational mapping and returns data as a table
  - Must parse each accessed document

XML and DB2 - XML Wrapper

- Mapping an XML Schema
  - Use NICKNAME to create mapped table to Schema
  - FOR SERVER defines the registered XML Wrapper Server
  - XPATH defines the mapped element name
  - PRIMARY/FOREIGN Key map relationship between internal XML Wrapper table (remember the tree structure)
  - The relevant XML document is identified using
    - DIRECTORY_ PATH
    - FILE_PATH
    - DOC column containing the “URI” keyword

XML and DB2 - XML Wrapper

```
CREATE NICKNAME OrdersNN
    (ID VARCHAR(16) NOT NULL OPTIONS(PRIMARY_KEY 'YES'),
    Number INTEGER NOT NULL OPTIONS(XPATH './@Number'),
    OrderDate DATE NOT NULL OPTIONS(XPATH './OrderDate'),
    CustNum INTEGER NOT NULL OPTIONS(XPATH './CustomerNumber'))
    FOR SERVER xml_server
    OPTIONS(DIRECTORY_PATH 'c:\OrderDocs\', XPATH '/SalesOrder')

  CREATE NICKNAME ItemsNN
    (ParentID VARCHAR(16) NOT NULL OPTIONS(FOREIGN_KEY 'OrdersNN'),
    Number INTEGER NOT NULL OPTIONS(XPATH './@Number'),
    PartNum VARCHAR(10) NOT NULL OPTIONS(XPATH './PartNumber'),
    Quantity INTEGER NOT NULL OPTIONS(XPATH './Quantity'),
    Price DECIMAL(8,2) NOT NULL OPTIONS(XPATH './Price'))
    FOR SERVER xml_server
    OPTIONS(XPATH '/SalesOrder/Item')
```

XML and DB2 - XML Wrapper

Querying through the XML wrapper

SELECT Number FROM OrdersNN WHERE CustNum = 456 (for root nickname)

SELECT SUM(Quantity)

FROM OrdersNN, ItemsNN

WHERE OrdersNN.ID = ItemsNN.ParentID AND

CustNum = 456 AND

PartNum = ‘XY-47‘ (for multi-level join)

XML and DB2 - XML Wrapper

Shredding a document using the XML wrapper

```
INSERT INTO Orders (Number, Date, Customer)
  SELECT Number, OrderDate, CustNum FROM OrdersNN

INSERT INTO Items (SONumber, Number, PartNumber, Quantity,
Price)
  SELECT OrdersNN.Number, ItemsNN.Number, PartNum, Quantity, Price
  FROM OrdersNN, ItemsNN
  WHERE OrdersNN.ID = ItemsNN.ParentID
```

Ascertain that data is retrieved and inserted in the correct order

XML and DB2 – Websphere MQ

- Using DB2 XML Extender user can send and retrieve messages directly to/from message queues
  - Publish to queue
  - Retrieve
  - Receive
  - Stored Procedures to Publish / Shred documents to/from MQ

DB2 futures – XML in vNext

- Moving functionality to the DB Engine
  - Not all functions identical across the family
- Storing XML documents natively
- Changes to DDL
  - CREATE/ALTER with XML data type (automatically create the auxiliary objects)
  - Indexing support using XPATH to determine which elements comprise the index
  - INSERT/UPDATE/DELETE
    - INSERT with VALUES and SUBSELECT
    - Current document only

DB2 futures – XML in vNext

- New SQL/XML Functions and Predicates
  - XMLPARSE – Convert and XML text to an XML Value
  - XMLSERIALIZE – Convert XML data type to Non XML
  - XMLQUERY – execute an XPATH expression against and XML value
  - XMLCAST – cast XML to other types and vice versa
  - XMLEXISTS – return TRUE if an XPATH expression evaluates to a non empty sequence

DB2 futures – XML in vNext

- XPATH
  - XPATH 2.0 feature support
- Utility support for XML data and structures
  - Load/Unload, Check Data/Index, Copy, Rebuild, Recover, Reorg etc.
- XML Schema Support
  - XSR – XML Schema Repository
  - Catalog tables to store XML schemas
  - Stored procedures to register XML schemas
- XMLVALIDATE SQL/XML function
  - Test XML for Validity against XML schema
  - Obtain default and schema normalized values from XML schema

Agenda

- An Introduction to XML
- XML and your Database
- XML and DB2
- DB2 futures – XML in vNext

Appendix A *DTD symbols*

Symbol	Meaning	Example	Description
, (comma)	Means "and" in specified order	TITLE, AUTHOR	TITLE and AUTHOR in that order
|	Means "or"	TITLE | AUTHOR	TITLE or AUTHOR
?	Means "optional", but no more than one is allowed	ISBN?	ISBN does not have to be present, but if it is, there can be no more than one.
*	Means 0 or more Elements.	(TITLE | AUTHOR)*	Any number of TITLE or AUTHOR elements can be present
+	Means 1 or more Elements	AUTHOR+	At least one or more AUTHOR elements must be present
( )	Used to group elements	<!ELEMENT BOOK (AUTHOR | TITLE, YEAR-PUBLISHED, ISBN)>	An AUTHOR or a TITLE element must be present and must precede the YEAR-PUBLISHED and ISBN elements.

Appendix A *DTD attribute types*

Attribute type	Description
CDATA	Can contain any kind of character data.
ID	Must have unique values within the element. In the example below, TYPEID is of the ID type, and so requires unique values within the range of BOOK elements: <BOOK TYPEID="ch1">See Spot Run</BOOK> <BOOK TYPEID="ch2">Jack and Jill</BOOK>
IDREF	The value of an ID type attribute of an element in the document.
IDREFS	Multiple ID's of elements separated by whitespace.
(enumerated)	Attributes can have a specified list of acceptable values.
ENTITY	The name of an entity declared in the DTD.
ENTITIES	The attribute is optional.
NMTOKEN	The attribute is fixed (the syntax is of the type "#FIXED *Value*").
NOTATION	The name of a notation declared in the DTD.
NMTOKENS	Multiple XML names separated by whitespace.

Appendix A *DTD default attribute values*

Attribute value	Description
#REQUIRED	The attribute is required.
#IMPLIED	The attribute is optional.
#FIXED	The attribute is fixed (the syntax is of the type "#FIXED *Value*").

Appendix B – sources

- DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More
  SG24-6079-00
- XML for DB2 Information Integration
  SG24-6994-00
- www.w3c.org
- DB2 for z/OS v8 and Beyond – Curt Cotner
  ftp://ftp.software.ibm.com/software/data/db2zos/VNEXT.pdf

Thank you!!!