SWG BetaWorks

DB2 9

Technical Education Series

“XML Part 1 (Foundation and DBA)”

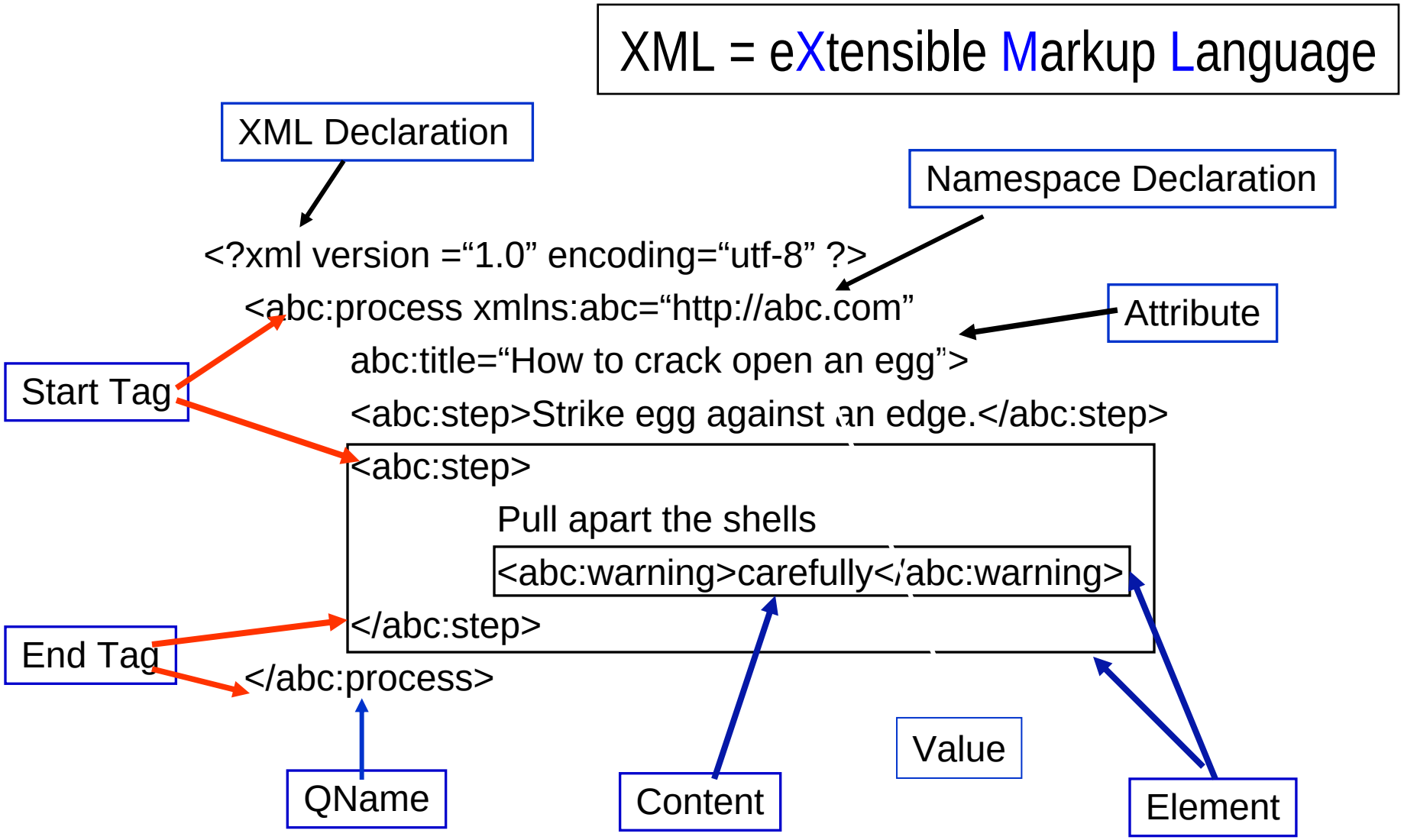
© 2007 IBM Corporation

XML Support in DB2 9 for z/OS

- **Part 1 (Foundation and DBA)**
  - **What is XML?**
  - **XML support in DB2 9**
  - **Storage Infrastructure**
  - **XML Schema Support**
  - **Utilities**
- Part 2 (Application Development)
  - XML in DB2 Application Development
  - Inserting,Updating, Deleting XML
  - Querying XML
    - XPath
  - SQL/XML
    - Publishing XML
  - Programming Access

© 2007 IBM Corporation

XML document



© 2007 IBM Corporation

Well-formed XML Documents

- **An XML document is well-formed, if:**

	Not well-formed 	Well-formed 
Has exactly one root element	bla <c>blub</c>	<a> bla <c>blub</c>
Each opening tag is matched by a closing tag	<a>bla	<a>bla
All elements are properly nested	<a>bla	<a>bla
Attribute values must be quoted		
Does not use disallowed characters in tags or values	<a> 3<5 	<a> 3<5
…	**…**	**…**

Note: xml header <?xml version="1.0"?> is NOT required for wellformedness.
See http://www.w3.org/TR/REC-xml for full definition.

© 2007 IBM Corporation

Why XML?

- **Vendor and platform independent**
  - Any platform, vendor, OS, software, language
- **A very flexible data model**
  - For structured data, semi-structured data, schema-less data
- **Easy to extend**
  - Define new tags as needed
- **Self-describing**
  - Any XML parser can "understand" it!
- **Easy to "validate" (i.e., to check compliance with a schema)**
  - Any Schema Validating XML parser can do it!
- **Easy to transform into other formats (HTML, etc.)**
- **Fully Unicode compliant**

© 2007 IBM Corporation

Example: Financial Data (FIXML)

- Buying 1000 Shares of IBM Stock..

Old FIX Protocol

```
8=FIX.4.2^9=251^35=D^49=AFUNDMGR^56=ABROKER^34=2
^52=20030615-01:14:49^11=12345^1=111111^63=0^64=2003
0621^21=3^110=1000^111=50000^55=IBM^48=459200101^22=
1^54=1^60=2003061501:14:4938=5000^40=1^44=15.75^15=USD
^59=0^10=127
```

New FIXML Protocol

- extensible
- lower appl development & maintenance cost

```
<FIXML>
  <NewOrdSingle ClOrdID="123456"
    Side="2"
    TransactTm="2003-06-15T01:14:49-05:00"
    OrderType="2"
    Price="93.25"
    Acct="26522154">
    <Header Sent="2001-06-21T01:31:28-05:00"
      PosDup="N"
      PosRsnd="N"
      SeqNum="521">
      <Sender ID="AFUNDMGR"/>
      <Target ID="ABROKER"/>
    </Header>
    <Instrument Symbol="IBM"
        ID="459200101"
        IDSrc="1"/>
    <OrderQuantity Qty="1000" Cur="USD"/>
  </NewOrdSingle>
</FIXML>
```


© 2007 IBM Corporation

Why use XML with Databases?

- **Managing Large volumes of XML data is a DB problem**
  - Efficient Search & Retrieval of XML
  - Persistency, Recovery, Transitions, ACID
  - Performance, Scalability
  - …all the same reasons as for relational data!
- **Integration**
  - Integrate new XML data with existing relational data
  - Publish (relational) data as XML
  - Database support for web applications, SOA, web services (SOAP)

© 2007 IBM Corporation

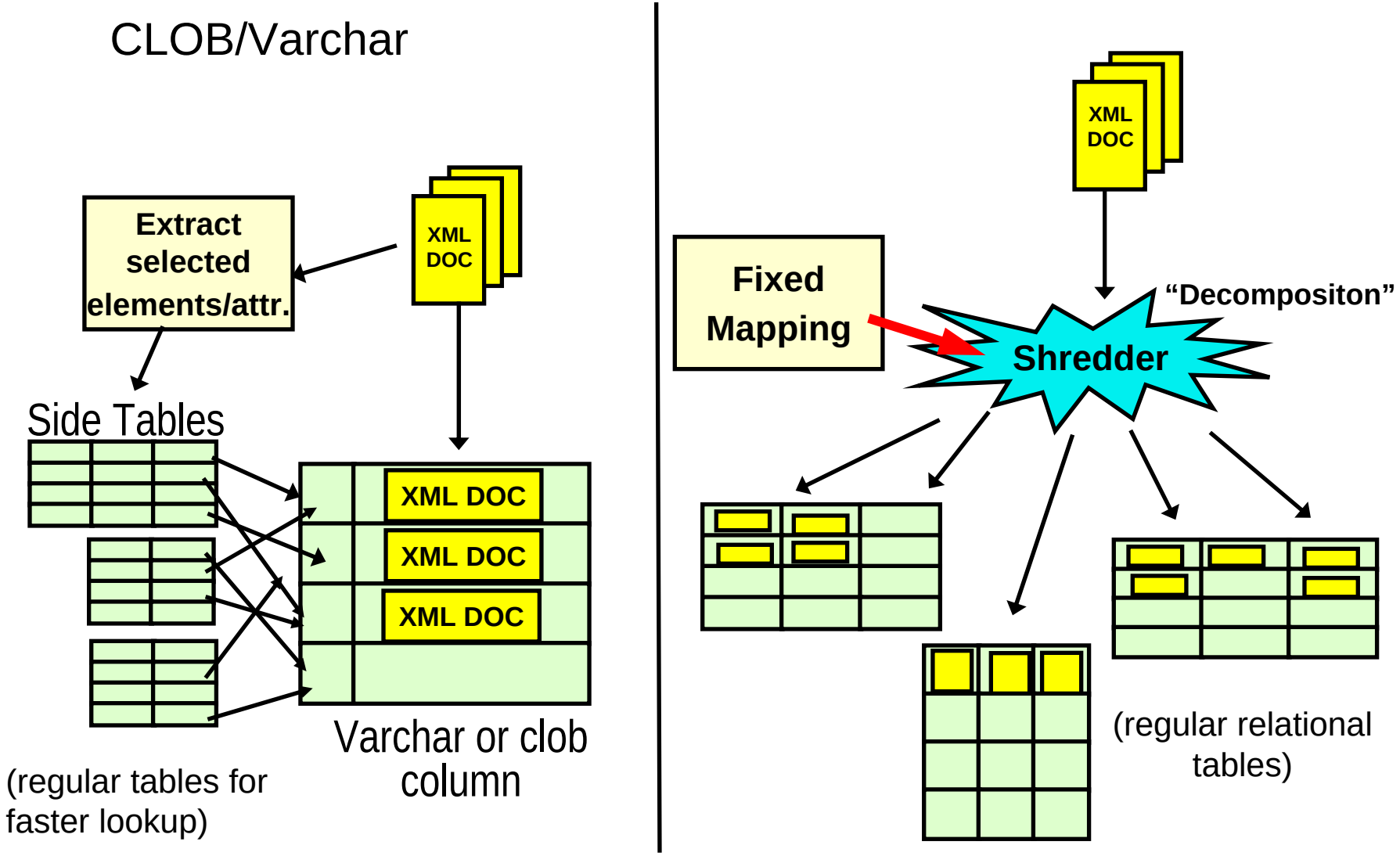
XML Databases

- **XML-enabled Databases**
  - Core data model is not XML but relational
  - Mapping between XML data model and DB's data model is required, or XML is stored as text in LOB
  - e.g. DB2 XML Extender

- **Native XML Databases (DB2 9)**
  - Use the hierarchical XML data model to store and process XML internally
  - No mapping, no storage as text
  - Storage format = processing format

© 2007 IBM Corporation

XML-Enabled Databases: Two Main Options



© 2007 IBM Corporation

Problems of XML-enabled Databases

- CLOB storage:
  - Query evaluation & sub-document level access requires costly XML Parsing – ***Too slow*** **!**
- Shredding:
  - Mapping from XML to relational often too complex
  - Often requires dozens or hundreds of tables
  - Complex multi-way joins to reconstruct documents
  - XML schema changes break the mapping
    - No schema flexibility !
    - For example: Change element from single to multi occurrence requires normalization of relational schema & data

© 2007 IBM Corporation

What is a native XML database?

- **Store XML most optimally**
  - … for querying (i.e. XPath)
  - … for flexibility (that is what XML is all about)
  - … in UTF-8

- **This means**
  - ~~Not storing as CLOB~~
  - ~~Not storing as object-relational~~
  - ~~Not shredding in rows and columns~~

Storing XML document in parsed hierarchical format

© 2007 IBM Corporation

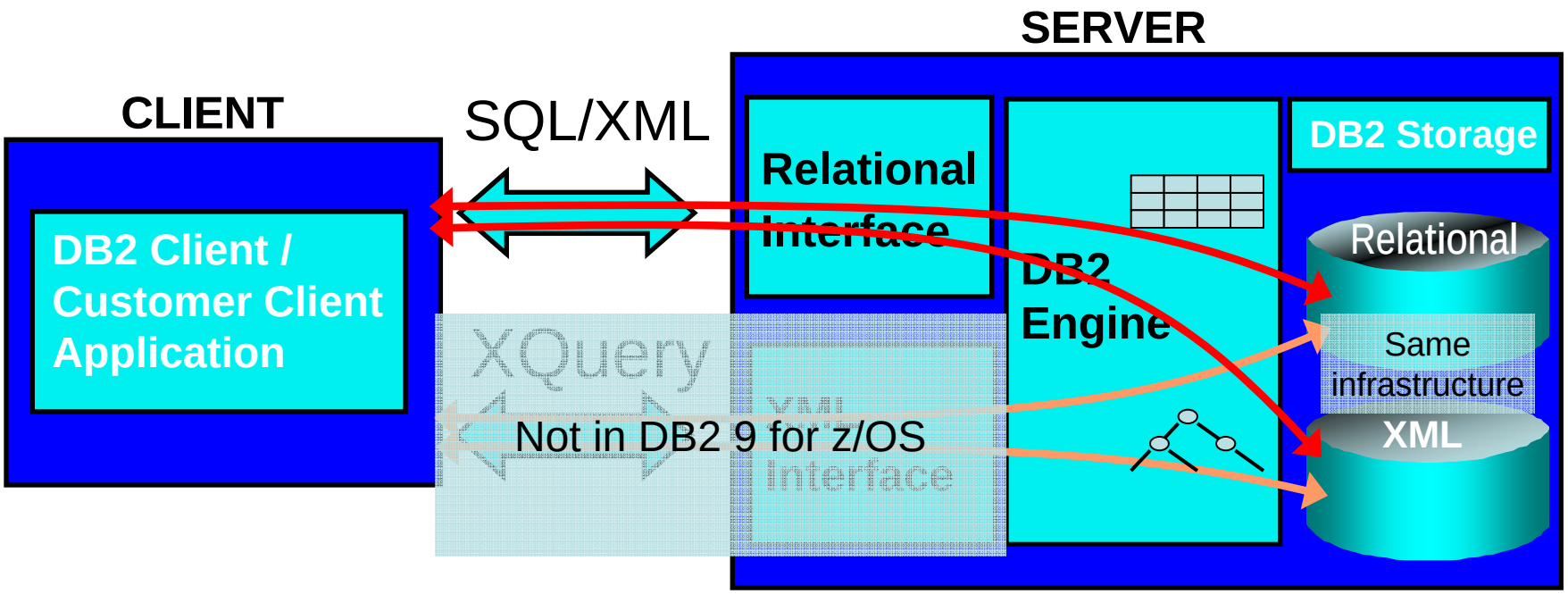
Native XML in DB2 (pureXML®)

- **Standards compliant**
  - XML, XPath, SQL/XML, XML Schema …
- **100% integrated in DB2**
  - leveraging performance, scalability, reliability, availability …
- **100% integrated with SQL**
  - XML is a new SQL type
  - Access relational and XML data in same SQL statement (SQL with XML extensions, SQL/XML)
- **100% integrated with application APIs:**
  - COBOL, Assembler, PL/I
  - Java (JDBC or SQLJ), C or C++ (in embedded SQL or DB2 ODBC applications)

© 2007 IBM Corporation

Native XML & Relational Integration

- **XML Capabilities in all DB2 components**
- **Applications combine XML & relational data**



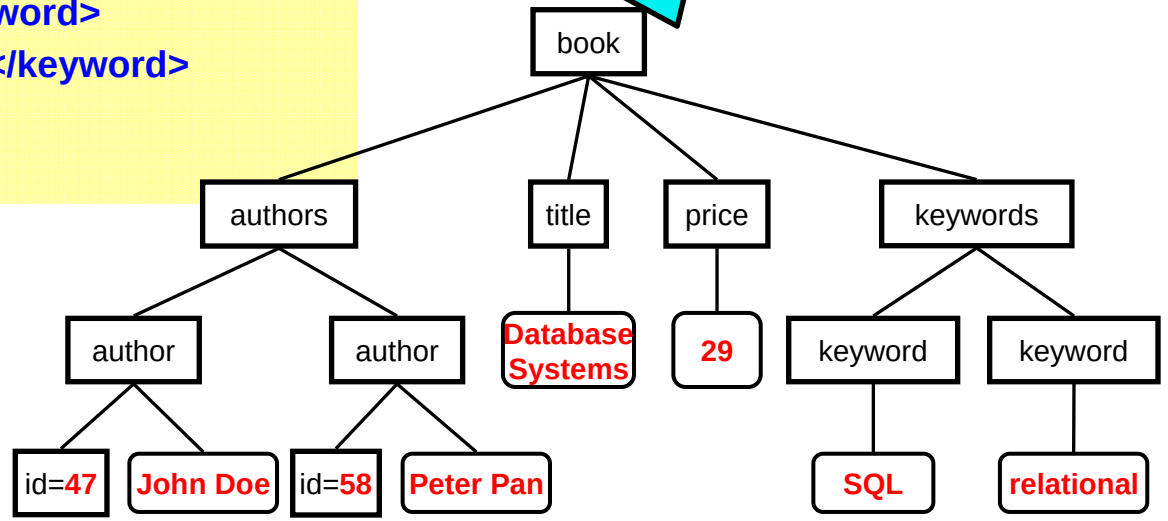
© 2007 IBM Corporation

XML Representation in Tree Format

```
<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>
```

XML Parsing

Serialization

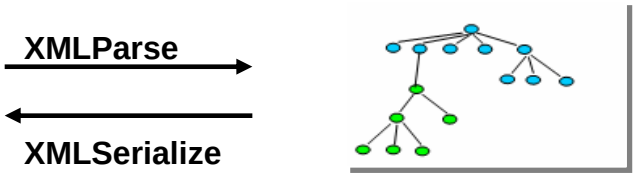


© 2007 IBM Corporation

XML Data Type

- **New data type: XML**
- **Native XML data type**
  - No defined length
  - XML document is parsed and stored natively in XQuery Data Model (XDM) – like a tree format
  - Transformation of XML document to/from internal XDM format is done implicitly by DB2 or explicitly by applications using
    - XMLPARSE()
    - XMLSERIALIZE()

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-
   <shipTo country="US">
    <name>Alice Smith</name>
```



© 2007 IBM Corporation

DDL Examples for XML column creation

```
CREATE TABLE PurchaseOrders (
        ponumber varchar(10) not null,
        podate date not null,
        status char(1),
        XMLPO xml);
```

- Hidden DocID column
- One DocID index
- Internal XML table for each XML column
- NodeID index

```
CREATE VIEW ValidPurchaseOrders as
        SELECT ponumber, podate, XMLPO
           FROM PurchaseOrders
           WHERE status = 'A';

ALTER TABLE PurchaseOrders
         ADD revisedXMLpo xml;
```


© 2007 IBM Corporation

Manipulating XML Data

```
EXEC SQL BEGIN DECLARE SECTION;
      SQL TYPE IS XML AS CLOB(1M) xmlPo;
EXEC SQL END DECLARE SECTION;

INSERT INTO PurchaseOrders VALUES ('200300001',
             CURRENT DATE, 'A', :xmlPo);

UPDATE PurchaseOrders SET XMLpo = :XMLpo_backup
       WHERE ponumber = '12345';

DELETE FROM PurchaseOrders WHERE ponumber = '12345';

SELECT XMLPO INTO :xmlPo
            FROM PurchaseOrders
    WHERE ponumber = '200300001';

LOAD DATA INDDN(SYSREC)…into TABLE PurchaseOrders  (…
```

Host var of XML type

String literal is OK

Whole document replacement

© 2007 IBM Corporation

XML Storage Infrastructure

- XML Data stored in an XML column must be well-formed and is converted into UTF-8 format regardless of document encoding
- Several XML Implicit objects will be created automatically when an XML column is defined in a table
- Those XML implicit objects are:
  - XML Table, XML Table Space
  - XML indicator columns and DOCID column in base table,
  - XML generated internal Indexes
- Each defined XML column in base table will have its associated XML table, XML table space
  - XML table space is any regular table space
- User-defined XML index for better access performance

© 2007 IBM Corporation

DB2 XML Implicit Objects

The following XML Implicit objects will be created when an XML column is defined

- Columns in Base Table
  - XML Indicator column(s), plus
  - one DOCID column
- XML table space – (Xyyynnnn)
- XML table – (Xyyyyyyyyyyyyyyyyynnn) with 3 columns in it
  - DB2_GENERATED_DOCID_FOR_XML - BIGINIT
  - MIN_NODEID – VARBINARY
  - XMLDATA – VARBINARY
  - XML table is an internal table – not to be accessed directly by application
- XML Implicit Indexes are:
  - DocID index on base table
  - NodeID index on XML table – DOCID + NODEID

© 2007 IBM Corporation

XML Table Space and XML Table

- **XML table space – (Xyyynnnn)**
  - yyy is the first three bytes of the base table name
  - Padded with # sign if base table name less than 3 bytes, nnnn will start with 0000
  - Attributes inherited from the base table
- **XML table – (Xyyyyyyyyyyyyyyyyyynnn)**
  - yyyy…yyyy is the truncated18 bytes base tables name or whole table name if shorter
  - nnn starts with 001 (second XML table) and is incremented by 1
  - Generated table has 3 columns
    - DB2_GENERATED_DOCID_FOR_XML with type of BIGINT
    - MIN_NODEID with type of VARBINARY
    - XMLDATA with type of VARBINARY

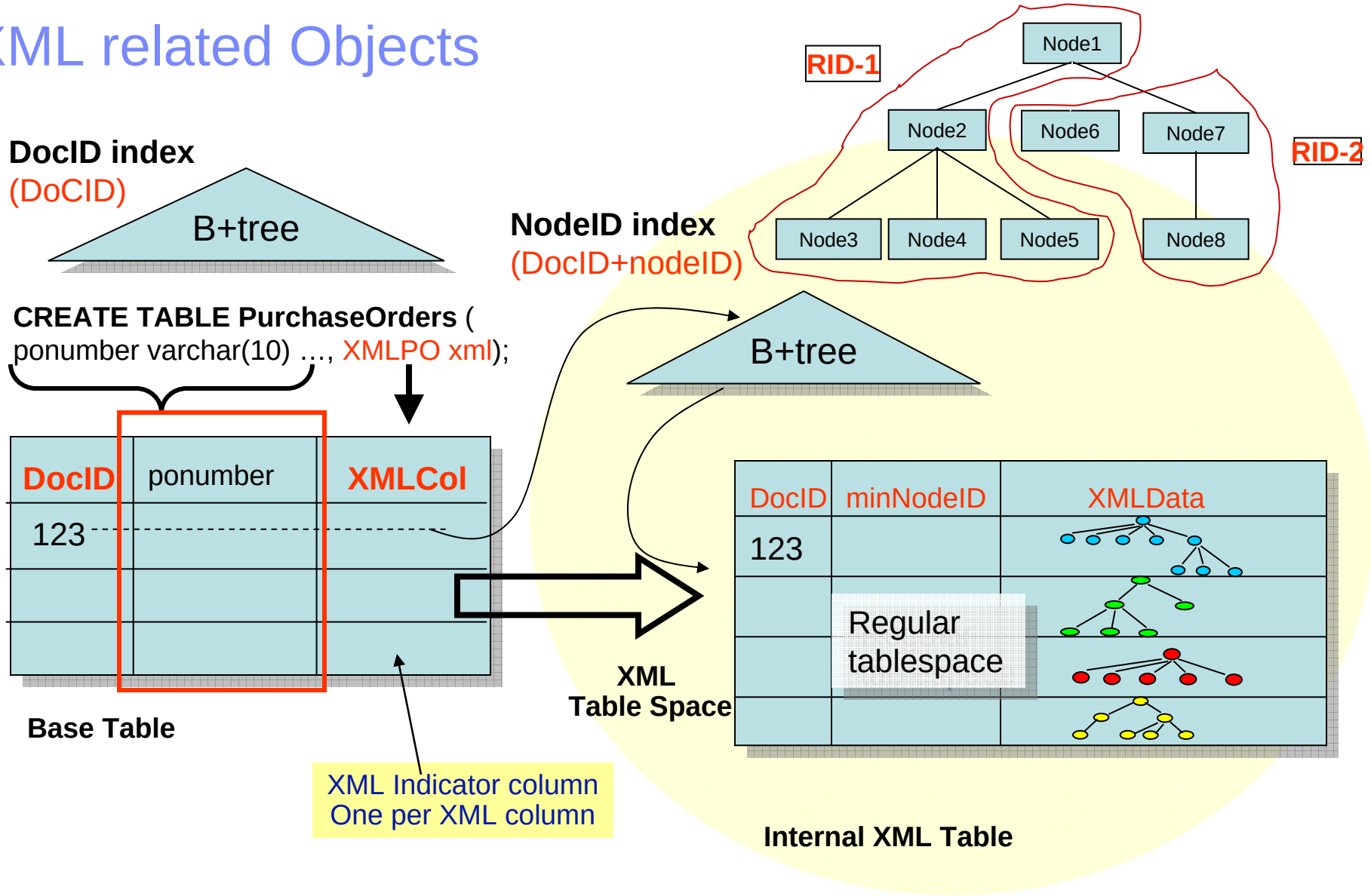
© 2007 IBM Corporation

XML generated Indexes

- **Index on Base Table – DOCID index (I_DOCIDyyyyyyyyyyyyyyyyy)**
  - Key(DB2_GENERATED_DOCID_FOR_XML)
  - yyyyyyyyyyyyyyyyy is the truncated 18 bytes of the base table name
  - provides the association from the base table row to the XML data for an XML column of that row
  - Map DocID to base table RID (regular index)
- **Index on XMLTable – NODEID index (I_NODEID_Xyyyyyyyyyyyyyyyyyynnn)**
  - Key (DB2_GENERATED_DOCID_FOR_XML, NODEID(XMLDATA)) , RID
  - yyyyyy..yyyyyyy is the truncated 18 bytes of the base table name
  - nnn starts at 001 for the second XML column and is incremented by 1
  - NODEID(XMLDATA) provides a nodeid range value for nodeids for a particular RID
  - Use to map NODEID to XML table RID
  - multiple index entries in the INDEX that point to the same RID

© 2007 IBM Corporation

XML related Objects



© 2007 IBM Corporation

Storing XML Trees – Tree Packing

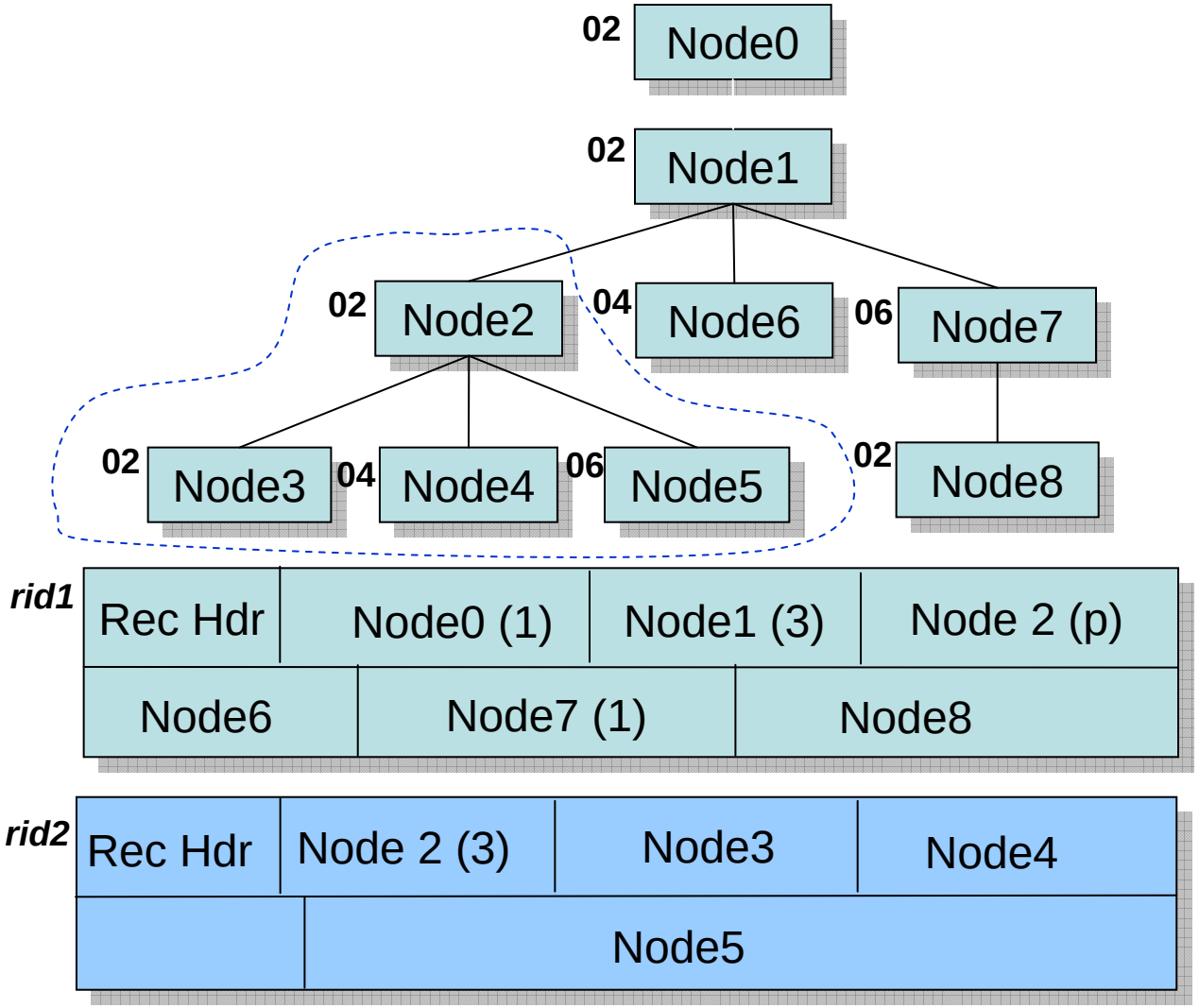
Each node contains local node id, length and optional number of children.

Proxy nodes are used as placeholder for subtrees in a separate record.

It supports traversal using *firstChild*, *nextSibling*, or *nextNode*.

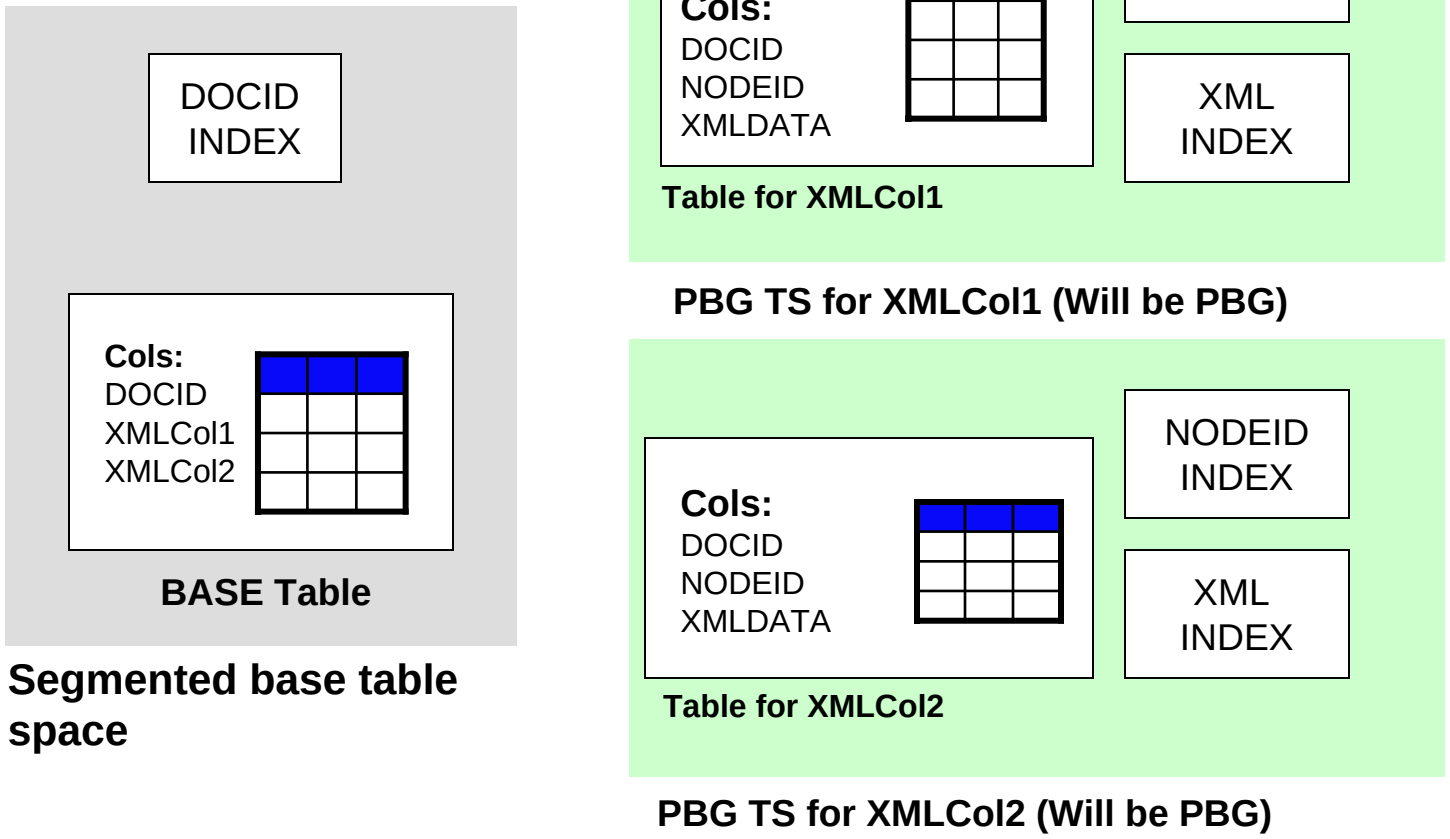
RecHdr contains context path information for the record – absolute ID, path, in-scope namespaces

All names use stringIDs.



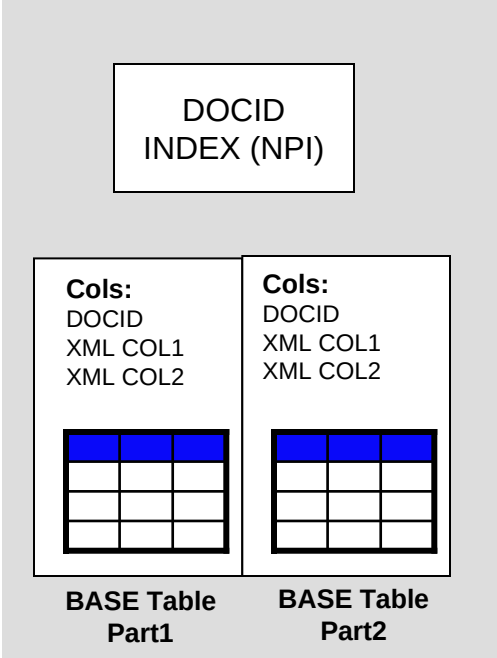
© 2007 IBM Corporation

XML Objects for a Base Table in Segmented Table space

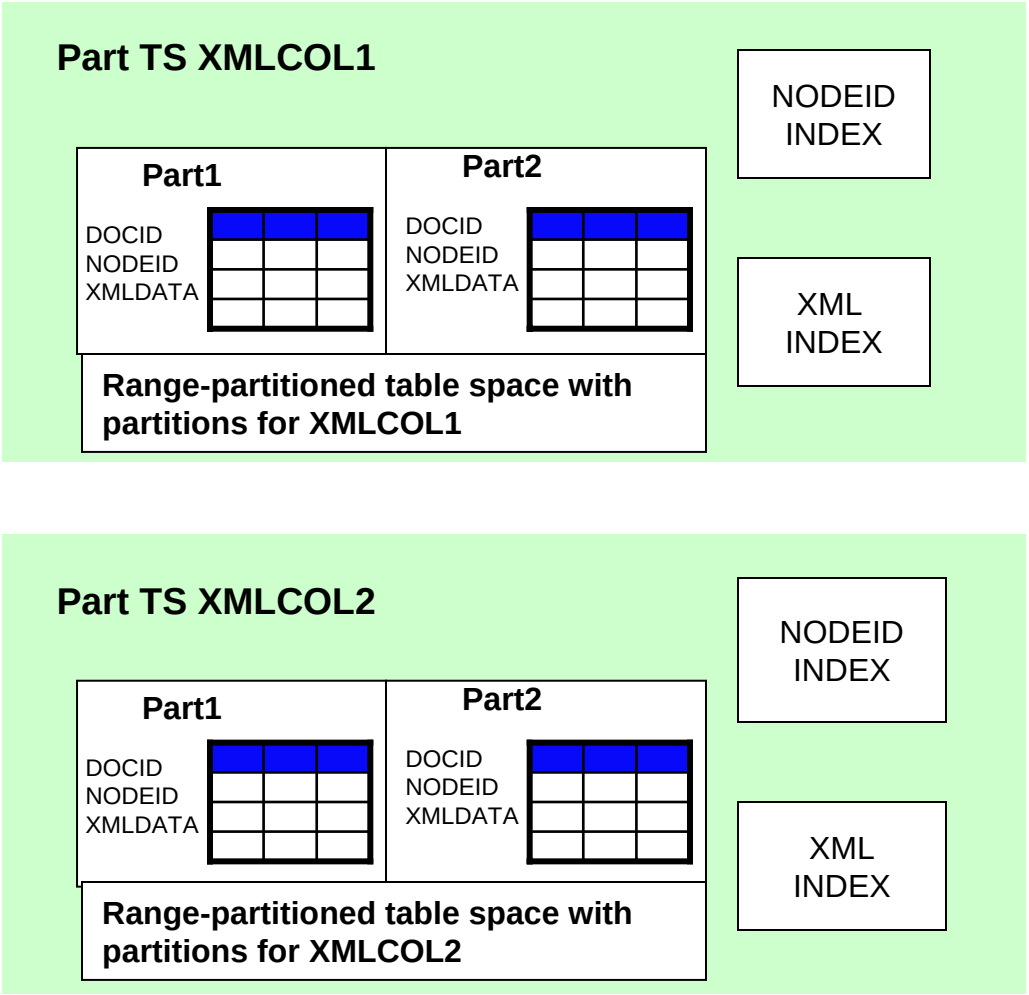


© 2007 IBM Corporation

XML Objects for a Base Table in Partitioned Table space



Partitioned Base TS
2 Parts, Table has 2 XML Columns



© 2007 IBM Corporation

Use of XML Schema in DB2 9

- Defines structure, content, data types of XML documents
  - Validate an XML document before storing to DB2
    - Validation is optional in DB2 9
  - Decompose (a.k.a. Shredding) XML document into relational data
    - XML schema annotations provide information about the columns and tables to be used to receive the XML data

© 2007 IBM Corporation

Use of XML Schema in DB2 9 (Cont)

- XML schemas need to be registered with the database before they can be used
- Supplied Stored procedures to register/remove XML schemas
- Upon registration , XML schemas are stored in XML Schema Repository (XSR)
- XSR is a set of table objects created during installation or migration
  - in Database DSNXR, table space SYSXSR
- XML schema name is in the form of “qualifier.name”. The qualifier is “SYSXSR”
  - No object usage privileges for XML schema, i.e. public

© 2007 IBM Corporation

XML Schema Registration

```
Call SYSPROC.XSR_REGISTER(
   'SYSXSR','ORDERSCHEMA','http://www.n1.com/report.xsd',:hv,NULL)

Call SYSPROC.XSR_ADDSCHEMADOC(
   'SYSXSR','ORDERSCHEMA','http://www.n1.com/ipo.xsd', :hv, NULL)

Call SYSPROC.XSR_ADDSCHEMADOC(
   'SYSXSR','ORDERSCHEMA','http://www.n1.com/address.xsd', :hv, NULL)

Call SYSPROC.XSR_COMPLETE(
   'SYSXSR','ORDERSCHEMA',NULL, 0)

Call SYSPROC.XSR_REMOVE('SYSXSR','ORDERSCHEMA')
```

If there is only 1 document in a XML schema , then it is not necessary to call **SYSPROC.XSR_ADDSCHEMADOC**

© 2007 IBM Corporation

XML Schema Document

report.xsd

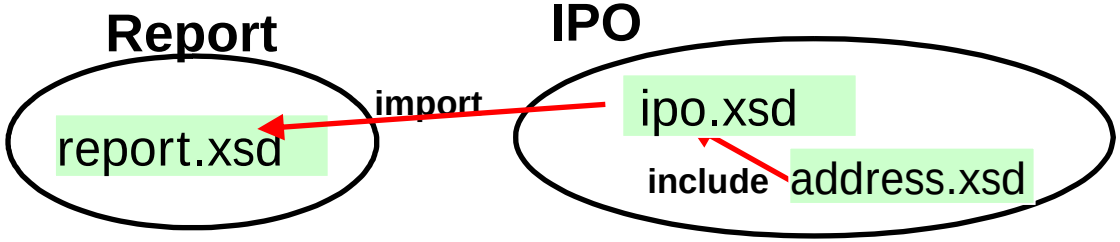
```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                                targetNamespace="Report">
<xsd:import namespace="IPO"
schemaLocation="http://www.n1.com/ipo.xsd"/>
…
</xsd:schema>
```

ipo.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="IPO">
<xsd:include schemaLocation="http://www.n1.com/address.xsd"/>
…
```

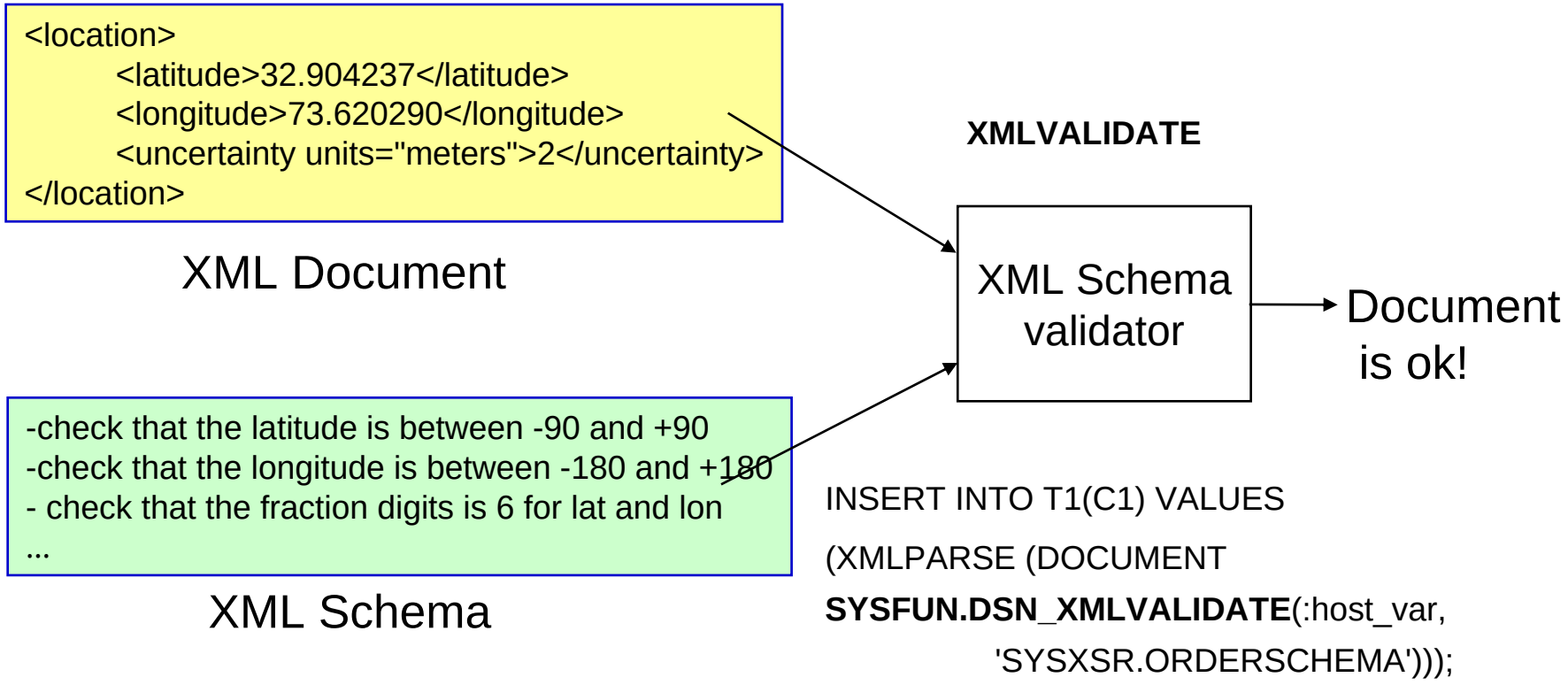
address.xsd

```
…
```



© 2007 IBM Corporation

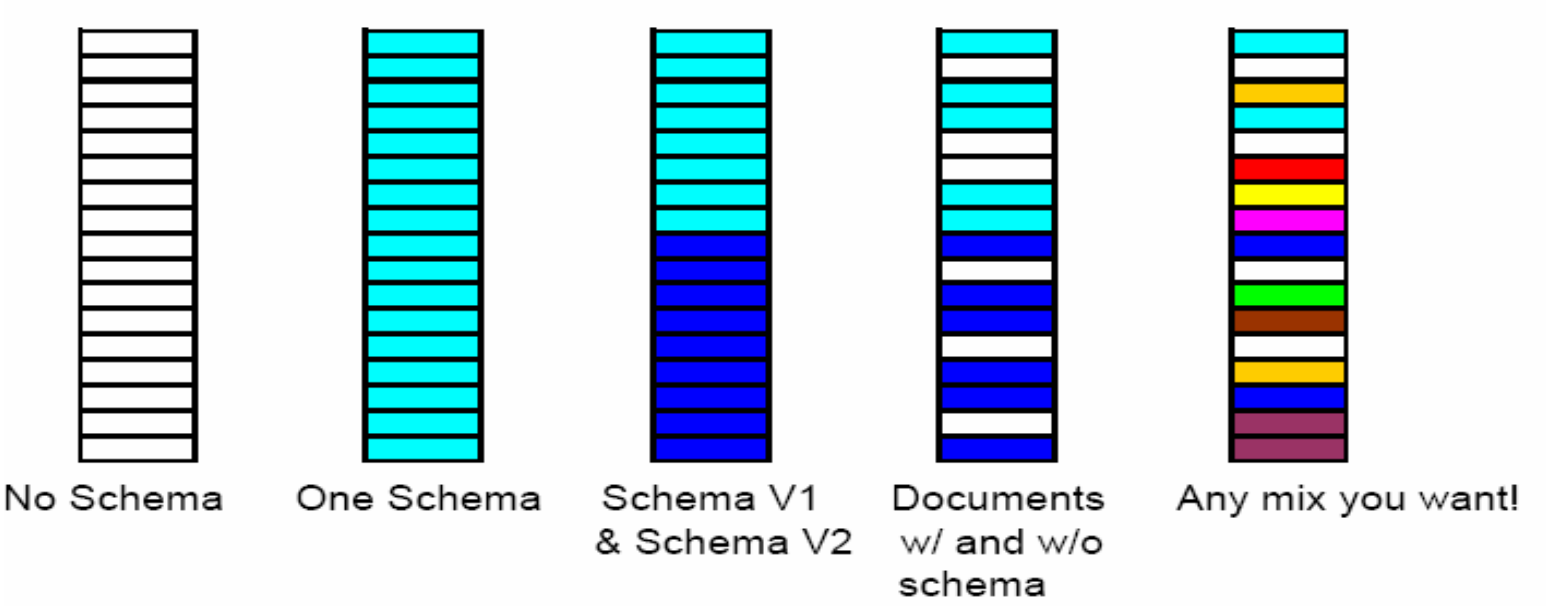
Validating your XML data using XML Schema



© 2007 IBM Corporation

XML Schema Flexibility

Mix of documents in an XML column → Many Options:



© 2007 IBM Corporation

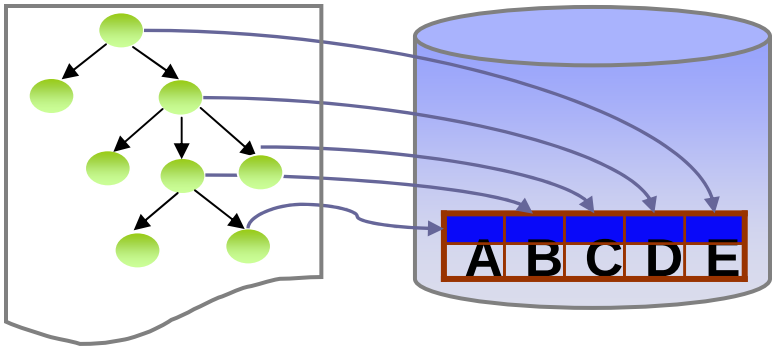
Decomposition Stored Procedure

- SYSPROC.XDBDECOMPXML
  - extract data items from a serialized XML value
  - populate data items into columns of relational tables with information from an annotated XML schema (shredding)
- Registered XML schema has to be enabled for decomposition during registration completion (XSR_COMPLETE)
  - Isusedfordecomposition = 1

Annotated XML Schema Example:

```
<xsd:element name="phone"
type="xsd:string"
        db2-xdb:rowSet="employee_tab"
        db2-xdb:column="phone_col"/>
```

Annotations



© 2007 IBM Corporation

Utilities

DB2 has utilities to support for the XML data type and the related database objects

- CHECK DATA
- CHECK INDEX
- COPY INDEX
- COPY TABLESPACE
- COPYTOCOPY
- LISTDEF
- LOAD
- MERGECOPY
- QUIESCE TABLESPACESET
- REAL TIME STATISTICS
- REBUILD INDEX
- RECOVER INDEX
- RECOVER TABLESPACE
- REORG INDEX
- REORG TABLESPACE
- REPORT TABLESPACESET
- UNLOAD
- Basic RUNSTATS

© 2007 IBM Corporation

REPORT TABLESPACESET - Output

Contains both a regular column and two XML columns. **(New XML text is shown in red)**

TABLESPACE SET REPORT:

TABLESPACE :	DSN00031.MYTABLE
TABLE:	PIC.MYTABLE
INDEXSPACE:	DSN00031.MYTABLEA
INDEX :	PIC.MYTABLEA_#_93Z
INDEXSPACE:	DSN00031.IRDOCIDM
INDEX :	PIC.I_DOCIDMYTABLE

DOCID INDEX

XML TABLESPACE SET REPORT:
TABLESPACE : DSN00031.MYTABLE

BASE TABLE		: PIC.MYTABLE
COLUMN		: COL2
	XML TABLESPACE	: DSN00031.XMYT0000
	XML TABLE	: PIC.XMYTABLE
	XML NODEID INDEXSPACE	: DSN00031.IRNODEID
	XML NODEID INDEX	: PIC.I_NODEIDXMYTABLE
	XML INDEXSPACE	: DSN00031.COL2INDE
	XML INDEX	: PIC.COL2INDEX
COLUMN		:COL3
	XML TABLESPACE	: DSN00031.XMYT0001
	XML TABLE	: PIC.XMYTABLE00
	XML NODEID INDEXSPACE	: DSN00031.IRNO1XBY
	XML NODEID INDEX	: PIC.I_NODEIDXMYTABLE 000

XML User Index

© 2007 IBM Corporation

Performance and Scalability

- XML storage leverages mature optimized storage infrastructure.
- Next generation parsers: XMLSS and XLXP.
- Most efficient XPath streaming algorithm
- Support partitioned table spaces and data sharing.
- XML table spaces have the same performance considerations as existing PARTITIONED and SEGMENTED table spaces do today.
- NodeID index has the same considerations as the existing ROWID auxiliary index for LOB data.
- REORG utility should be used to maintain order and free space.

© 2007 IBM Corporation

Operation and Recovery

- To recover base table space, take image copies of all related objects
  - Use REPORT TABLESPACESET to obtain a list of related objects
  - Use QUIESCE TABLESPACESET to quiesce all objects in the related set
- Use SQL SELECT to query the SYSIBM.SYSXMLRELS table for relationships between base table spaces and XML table spaces
  - COPYTOCOPY may be used to replicate image copies of XML objects.
  - MERGECOPY may be used to merge incremental copies of XML table spaces.
- Point in Timer recovery
  - RECOVER TOCOPY, TORBA, TOLOGPOINT
  - All related objects, including XML objects must be recovered to a consistent point in time
- CHECK utilities to validate base table spaces with XML columns, XML indexes and related XML table spaces.

© 2007 IBM Corporation

"Thank You for listening"

If you have any questions on this DB2 9 for z/OS session, then please send them to the BetaWorks team at:

Ian_Cook@uk.ibm.com
FLETCHPL@uk.ibm.com

© 2007 IBM Corporation