IBM Software Group

Introduction to pureXML in DB2 9 for z/OS

Guogen Zhang, gzhang@us.ibm.com

May 30, 2007

DB2 Information Management Software

e business on demand software

Agenda

- Why XML and XML Databases
- Comparing pureXML with existing approaches
- pureXML features
  - XML data type, DDL, DML
  - XML Query Languages and API
  - Indexing and access methods
  - Schema support
  - Utilities
- Performance
- Tools
- Summary

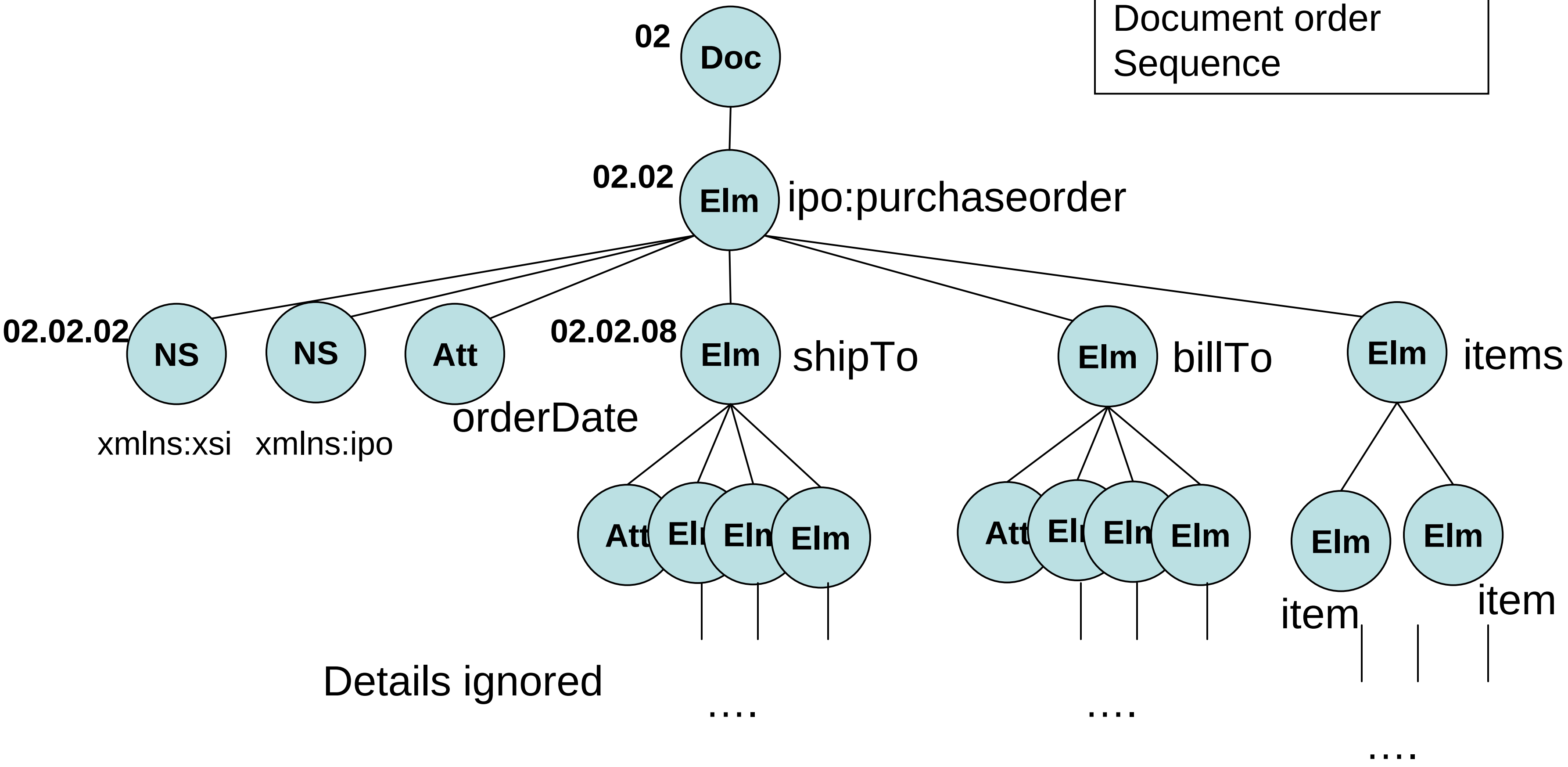
An XML Purchase Order

```
<?xml version="1.0" encoding="UTF-8"?>
<ipo:purchaseOrder
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:ipo="http://www.example.com/IPO" orderDate="1999-12-01">
   <shipTo exportCode="1" xsi:type="ipo:UKAddress">
     <name>Helen Zoe</name>
     <street>47 Eden Street</street>
     <city>Cambridge</city>
     <postcode>CB1 1JR</postcode>
   </shipTo>
   <billTo xsi:type="ipo:USAddress
     <name>Robert Smith</name>
     <street>8 Oak Avenue</street>
     <city>Old Town</city>
     <state>PA</state>
     <zip>95819</zip>
   </billTo>
```

```
    <items>
      <item partNum="833-AA">
        <productName>Lapis necklace</productName>
        <quantity>1</quantity>
        <USPrice>99.95</USPrice>
        <comment>Want this for the
holidays!</comment>
        <shipDate>1999-12-05</shipDate>
      </item>
      <item partNum="926-AA">
        <productName>Baby Monitor</productName>
        <quantity>1</quantity>
        <USPrice>39.98</USPrice>
        <shipDate>1999-12-21</shipDate>
      </item>
    </items>
</ipo:purchaseOrder>
```

XML Data Model (XDM)

Seven kinds of nodes
Node identity
Document order
Sequence



Details ignored

XML Characteristics

- XML can represent flexible structured data in text
  - Nesting
  - Repeating
  - Self-describing
- XML is a universal language to represent e-Business data and transactions.
- Platform-independent, and Unicode compliant
- Easy to understand and easy to process (with the right tools)

Why Hybrid XML Databases?

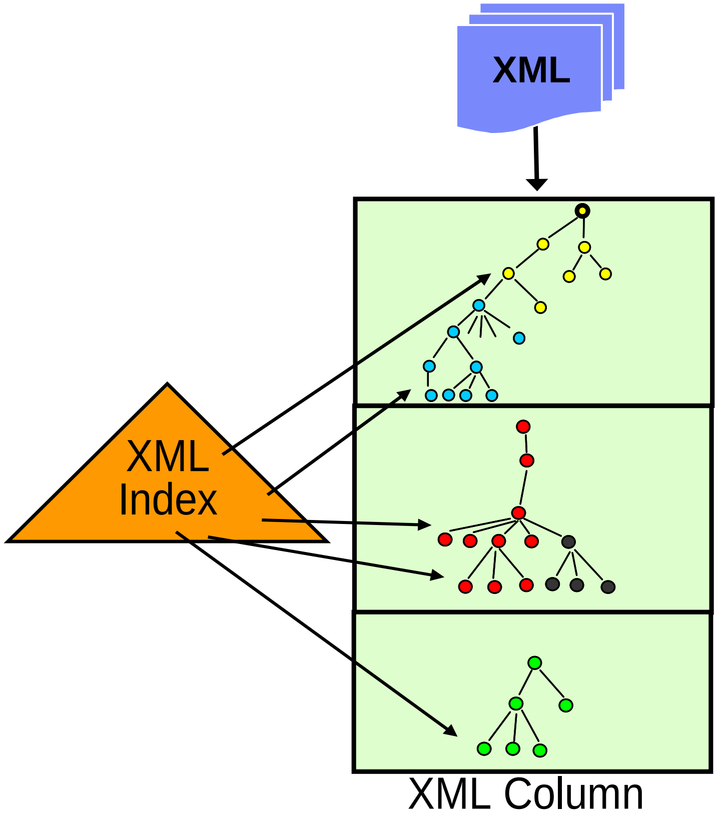
- Businesses need to manage XML data w/ ACID properties, auditing and regulatory compliance, together with relational data.
- XML can be used as a powerful data model, with powerful declarative query language.
- Managing large volumes of XML data is a DB problem
  - …all the same reasons as for relational data!
- Integration
  - Integrate new XML data with existing relational data
  - Publish (relational) data as XML
  - Database support for web applications, SOA, web services (SOAP)

pureXML in DB2 9

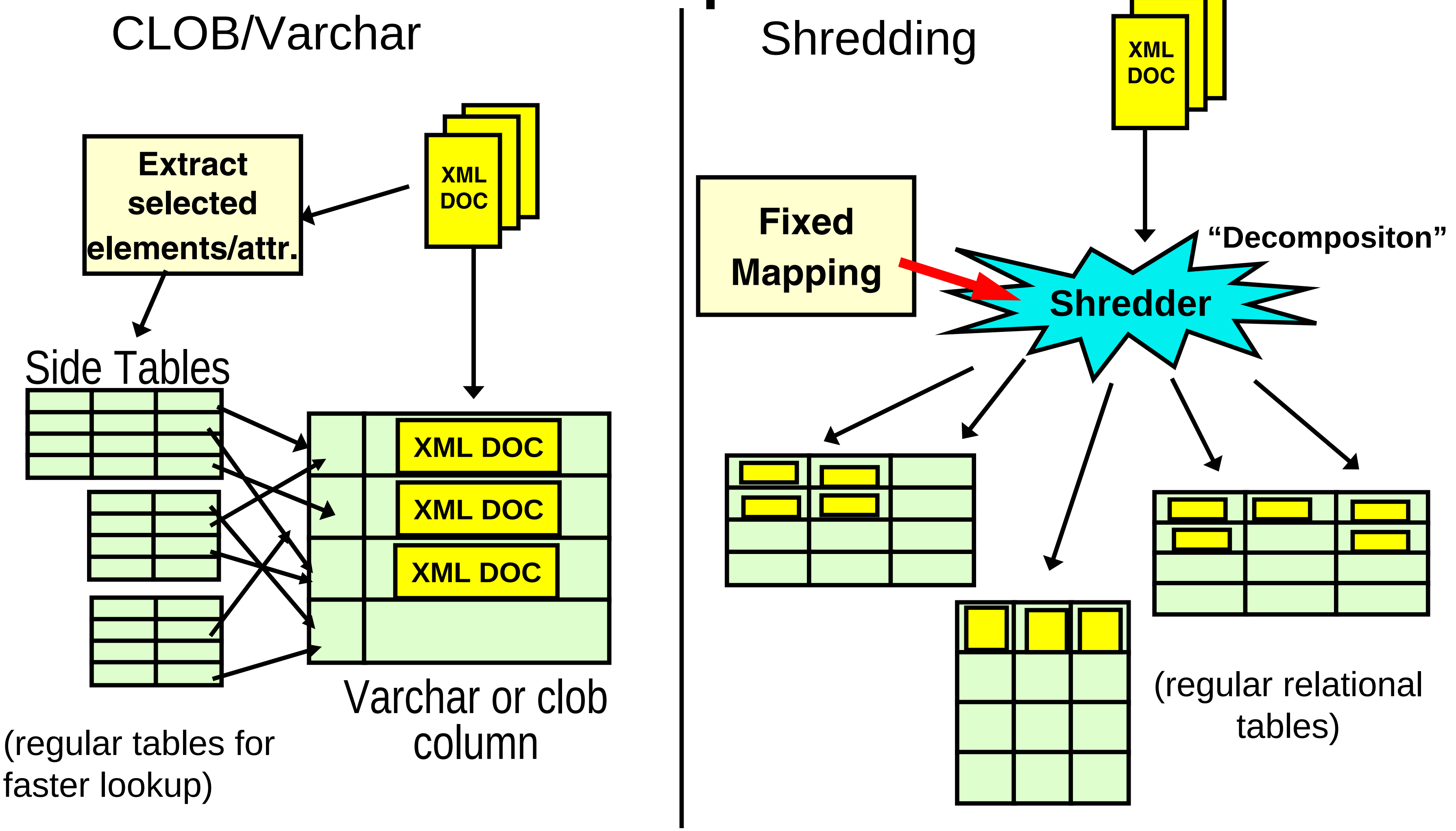
- SQL XML data type and native storage
- Designed specifically for XML
  - Supports XML hierarchical structure storage
  - Native operations and languages: XPath, SQL/XML, (XQuery in the future)
- Not transforming into relational
- Not using objects or nested tables
- Not using LOBs
- Integrated with relational engine, with all the utilities and tools support

What You Can Do with pureXML

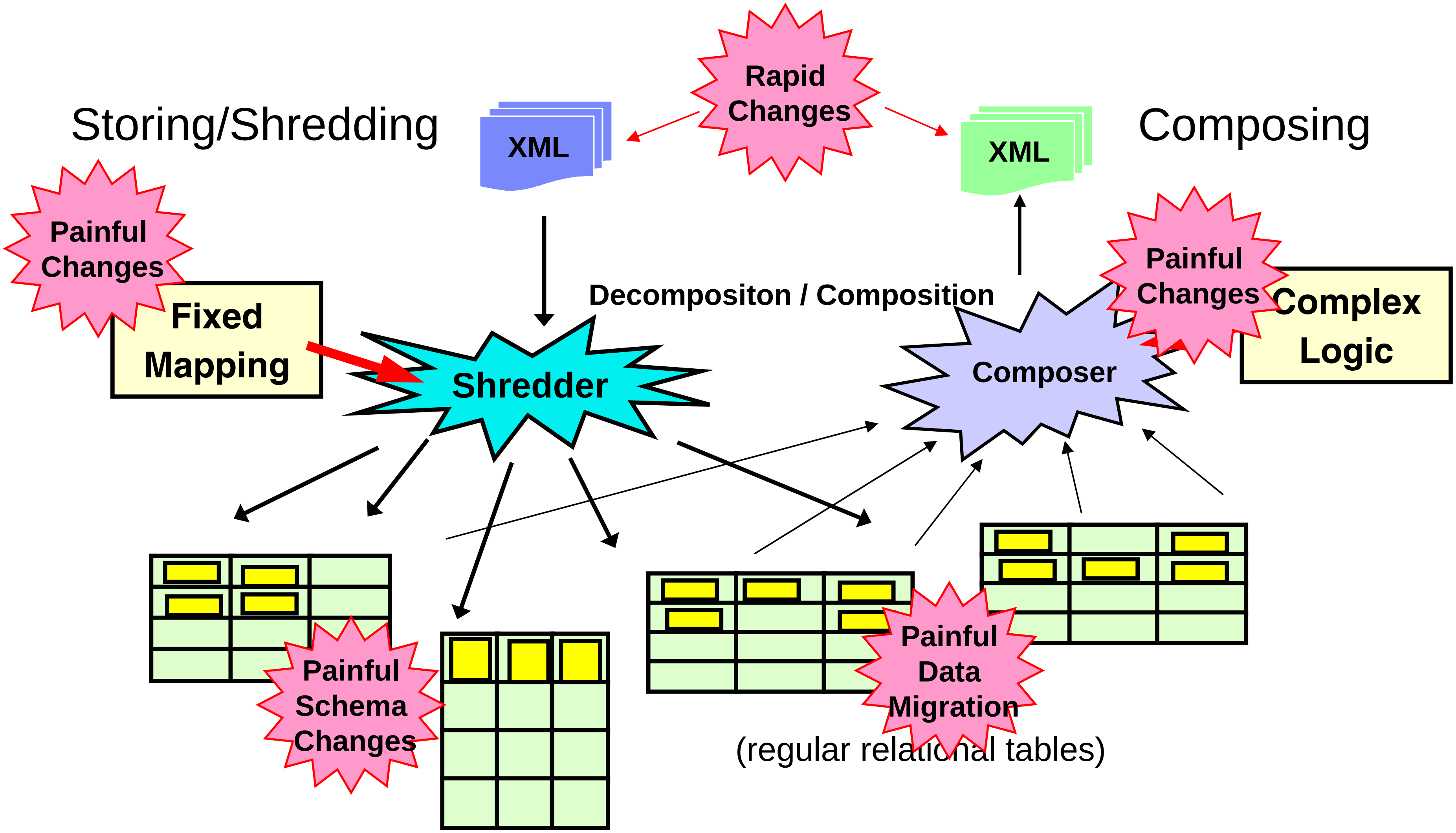
- Create tables with XML columns
- Insert XML data, optionally validated against schemas
- Create indexes on XML data
- Efficiently search XML data
- Extract XML data
- Decompose XML data into relational data
- Construct XML documents from relational and XML data
- All the utilities and tools support for XML



XML-Enabled Databases: Two Main Options

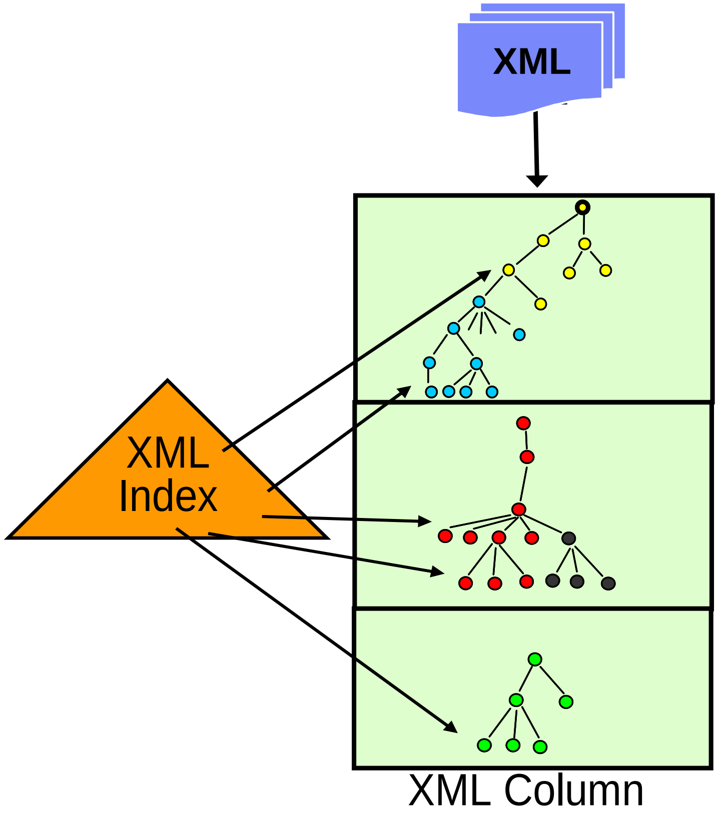


XML Data Processing before pureXML



DB2 pureXML Advantages

DB2's hierarchical storage:
XML type as XML



- Directly store XML, no decomp/comp, normalize/de-normalize
- Eliminates database schema evolution bottleneck
- Declarative language, reduce complexity, dramatically improve application development productivity
- Native processing, high performance
- Unparalleled reliability, availability, scalability

Up to 10 times

Usage Scenarios

- Directly processing XML
  - UNIFI, ACORD, FIXML, FpML, MIMSO, XBRL,
  - DJXDM, HR-XML, HL7, ARTS, HIPAA, NewsML, XForms
  - Insurance policy, contract, purchase order, emails etc
- Versatile schemas
- Sparse attribute values (null v.s. absence)
- Object persistence (single column v.s. many tables)
- Migration from legacy data model (network, hierarchical, relational)
- Generating web pages: XHTML
- Provide/consume Web Services (SOAP), support SOA
- …

Example: Tax Forms

- Application
  - Processing & validating tax returns, payments, refunds
  - Corporate Tax, Personal Income Tax (PIT), Sales Tax

- Objectives
  - Move Tax processing off legacy systems
  - ➢ **Move to a more flexible, automated, extensible framework Reduce cost & labor for implementing tax form changes**
  - Increase performance. Improve straight-through processing from filing to refund/payment

- Typical current environment
  - Processing using manual and/or legacy systems

- This is an example of usage for Online Forms processing in general

Tax Forms

- Usually hundreds-thousands of different tax forms

  → **Schema Diversity**

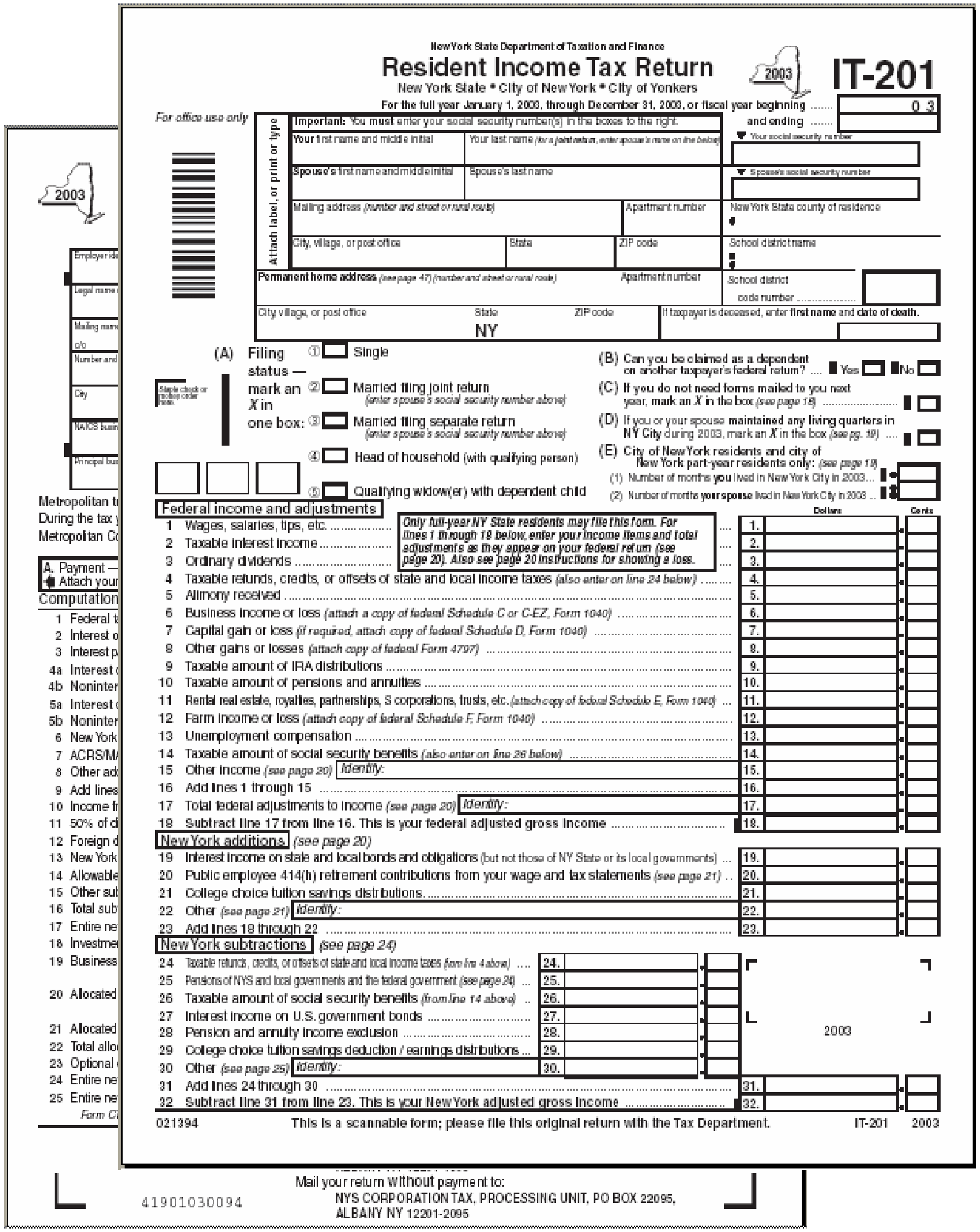
- Typically not every field in a form is used

  → **Sparse Data**

- Many forms change every year

  → **Schema Evolution**

→ **A case for XML !**



2003

Employer ide
Legal name
Mailing name
c/o
Number and
City
NAICS busin
Principal bus

Metropolitan t
During the tax y
Metropolitan C

A. Payment —
Attach your
Computation

1 Federal t
2 Interest o
3 Interest p
4a Interest
4b Noninter
5a Interest
5b Noninter
6 New York
7 ACRS/M
8 Other ad
9 Add lines
10 Income fr
11 50% of d
12 Foreign d
13 New York
14 Allowable
15 Other sub
16 Total sub
17 Entire ne
18 Investme
19 Business
20 Allocated
21 Allocated
22 Total allo
23 Optional
24 Entire ne
25 Entire ne
Form C

New York State Department of Taxation and Finance

Resident Income Tax Return

New York State * City of New York * City of Yonkers

2003

IT-201

For the full year January 1, 2003, through December 31, 2003, or fiscal year beginning 0 3
and ending ....

For office use only

Attach label, or print or type

Important: You must enter your social security number(s) in the boxes to the right.

Your first name and middle initial | Your last name *(for a joint return, enter spouse's name on line below)* | ▼ Your social security number

Spouse's first name and middle initial | Spouse's last name | ▼ Spouse's social security number

Mailing address *(number and street or rural route)* | Apartment number | New York State county of residence

City, village, or post office | State | ZIP code | School district name

Permanent home address *(see page 47) (number and street or rural route)* | Apartment number | School district code number ....

City, village, or post office | State NY | ZIP code | If taxpayer is deceased, enter first name and date of death.

Staple check or money order here.

(A) Filing status — mark an *X* in one box:
① □ Single
② □ Married filing joint return *(enter spouse's social security number above)*
③ □ Married filing separate return *(enter spouse's social security number above)*
④ □ Head of household (with qualifying person)
⑤ □ Qualifying widow(er) with dependent child

(B) Can you be claimed as a dependent on another taxpayer's federal return? .... Yes □ No □

(C) If you do not need forms mailed to you next year, mark an *X* in the box *(see page 18)* .... □

(D) If you or your spouse maintained any living quarters in NY City during 2003, mark an *X* in the box *(see pg. 19)* .... □

(E) City of New York residents and city of New York part-year residents only: *(see page 19)*
(1) Number of months you lived in New York City in 2003...
(2) Number of months your spouse lived in New York City in 2003 ..

Federal income and adjustments

Only full-year NY State residents may file this form. For lines 1 through 18 below, enter your income items and total adjustments as they appear on your federal return (see page 20). Also see page 20 instructions for showing a loss.

		Dollars	Cents
1 Wages, salaries, tips, etc.	1.		
2 Taxable interest income	2.		
3 Ordinary dividends	3.		
4 Taxable refunds, credits, or offsets of state and local income taxes *(also enter on line 24 below)*	4.		
5 Alimony received	5.		
6 Business income or loss *(attach a copy of federal Schedule C or C-EZ, Form 1040)*	6.		
7 Capital gain or loss *(if required, attach copy of federal Schedule D, Form 1040)*	7.		
8 Other gains or losses *(attach copy of federal Form 4797)*	8.		
9 Taxable amount of IRA distributions	9.		
10 Taxable amount of pensions and annuities	10.		
11 Rental real estate, royalties, partnerships, S corporations, trusts, etc. *(attach copy of federal Schedule E, Form 1040)* ...	11.		
12 Farm income or loss *(attach copy of federal Schedule F, Form 1040)*	12.		
13 Unemployment compensation	13.		
14 Taxable amount of social security benefits *(also enter on line 26 below)*	14.		
15 Other income *(see page 20)* Identify:	15.		
16 Add lines 1 through 15	16.		
17 Total federal adjustments to income *(see page 20)* Identify:	17.		
18 Subtract line 17 from line 16. This is your federal adjusted gross income	18.		

New York additions *(see page 20)*

		Dollars	Cents
19 Interest income on state and local bonds and obligations (but not those of NY State or its local governments) ...	19.		
20 Public employee 414(h) retirement contributions from your wage and tax statements *(see page 21)* ..	20.		
21 College choice tuition savings distributions........	21.		
22 Other *(see page 21)* Identify:	22.		
23 Add lines 18 through 22	23.		

New York subtractions *(see page 24)*

24 Taxable refunds, credits, or offsets of state and local income taxes *(from line 4 above)*	24.		
25 Pensions of NYS and local governments and the federal government *(see page 24)* ...	25.		
26 Taxable amount of social security benefits *(from line 14 above)* ..	26.		
27 Interest income on U.S. government bonds	27.		
28 Pension and annuity income exclusion	28.		
29 College choice tuition savings deduction / earnings distributions ...	29.		
30 Other *(see page 25)* Identify:	30.		
31 Add lines 24 through 30	31.		
32 Subtract line 31 from line 23. This is your New York adjusted gross income	32.		

2003

021394 This is a scannable form; please file this original return with the Tax Department. IT-201 2003

41901030094

Mail your return without payment to:
NYS CORPORATION TAX, PROCESSING UNIT, PO BOX 22095,
ALBANY NY 12201-2095

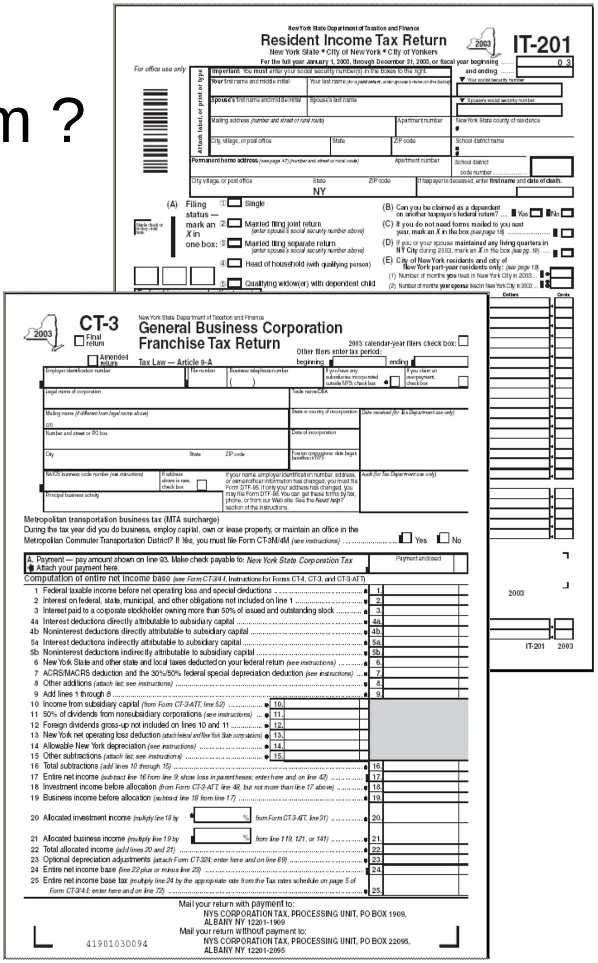
Typical Current Usage: Relational Database

- Solution 1: Each form has a different set of fields (schema)

→ Thousands of Tables … i.e. one per form ?

- Considered not feasible
  - Too many tables to maintain
  - Relational schema would deteriorate over time
  - Not sufficiently flexible and extensible

- Solution 2: Single table whose rows can store *any* form
  - 100s of generic columns … Ouch!



New York State Department of Taxation and Finance

Resident Income Tax Return 2003 **IT-201**

New York State • City of New York • City of Yonkers

For the full year January 1, 2003, through December 31, 2003, or fiscal year beginning 0 3
and ending ......

For office use only

Attach label, or print or type

Important: You **must** enter your social security number(s) in the boxes to the right.

Your first name and middle initial | Your last name *(for a joint return, enter spouse's name on line below)* | Your social security number

Spouse's first name and middle initial | Spouse's last name | Spouse's social security number

Mailing address *(number and street or rural route)* | Apartment number | New York State county of residence

City, village, or post office | State | ZIP code | School district name

Permanent home address *(see page 47) (number and street or rural route)* | Apartment number | School district code number ....

City, village, or post office | State NY | ZIP code | If taxpayer is deceased, enter **first name** and **date of death.**

Staple check or money order here.

(A) Filing status — mark an *X* in one box:
① □ Single
② □ Married filing joint return *(enter spouse's social security number above)*
③ □ Married filing separate return *(enter spouse's social security number above)*
④ □ Head of household (with qualifying person)
⑤ □ Qualifying widow(er) with dependent child

(B) Can you be claimed as a dependent on another taxpayer's federal return? .... Yes □ No □
(C) If you do not need forms mailed to you next year, mark an *X* in the box *(see page 18)* .... □
(D) If you or your spouse **maintained any living quarters in NY City** during 2003, mark an *X* in the box *(see pg. 19)* .... □
(E) City of New York residents and city of New York part-year residents only: *(see page 19)*
(1) Number of months **you** lived in New York City in 2003...
(2) Number of months **your spouse** lived in New York City in 2003 ...

Dollars | Cents

2003

IT-201 2003

CT-3 2003 □ Final return □ Amended return

New York State Department of Taxation and Finance

General Business Corporation Franchise Tax Return

Tax Law — Article 9-A

2003 calendar-year filers check box: □
Other filers enter tax period: beginning ending

Employer identification number | File number | Business telephone number ( ) | If you have any subsidiaries incorporated outside NYS, check box □ | If you claim an overpayment, check box □

Legal name of corporation | Trade name/DBA

Mailing name *(if different from legal name above)* c/o | State or country of incorporation | *Date received (for Tax Department use only)*

Number and street or PO box | Date of incorporation

City | State | ZIP code | Foreign corporations: date began business in NYS

NAICS business code number *(see instructions)* | If address above is new, check box □ | If your name, employer identification number, address, or owner/officer information has changed, you must file Form DTF-95. If only your address has changed, you may file Form DTF-96. You can get these forms by fax, phone, or from our Web site. See the *Need help?* section of the instructions. | *Audit (for Tax Department use only)*

Principal business activity

Metropolitan transportation business tax (MTA surcharge)
During the tax year did you do business, employ capital, own or lease property, or maintain an office in the Metropolitan Commuter Transportation District? If Yes, you must file Form CT-3M/4M *(see instructions)* .... □ Yes □ No

A. Payment — pay amount shown on line 93. Make check payable to: *New York State Corporation Tax*
Attach your payment here. | Payment enclosed

Computation of entire net income base *(see Form CT-3/4-I, Instructions for Forms CT-4, CT-3, and CT-3-ATT)*

1 Federal taxable income before net operating loss and special deductions .... 1.
2 Interest on federal, state, municipal, and other obligations not included on line 1 .... 2.
3 Interest paid to a corporate stockholder owning more than 50% of issued and outstanding stock .... 3.
4a Interest deductions directly attributable to subsidiary capital .... 4a.
4b Noninterest deductions directly attributable to subsidiary capital .... 4b.
5a Interest deductions indirectly attributable to subsidiary capital .... 5a.
5b Noninterest deductions indirectly attributable to subsidiary capital .... 5b.
6 New York State and other state and local taxes deducted on your federal return *(see instructions)* .... 6.
7 ACRS/MACRS deduction and the 30%/50% federal special depreciation deduction *(see instructions)* ... 7.
8 Other additions *(attach list; see instructions)* .... 8.
9 Add lines 1 through 8 .... 9.
10 Income from subsidiary capital *(from Form CT-3-ATT, line 52)* .... 10.
11 50% of dividends from nonsubsidiary corporations *(see instructions)* ... 11.
12 Foreign dividends gross-up not included on lines 10 and 11 .... 12.
13 New York net operating loss deduction *(attach federal and New York State computations)* 13.
14 Allowable New York depreciation *(see instructions)* .... 14.
15 Other subtractions *(attach list; see instructions)* .... 15.
16 Total subtractions *(add lines 10 through 15)* .... 16.
17 Entire net income *(subtract line 16 from line 9; show loss in parentheses; enter here and on line 42)* .... 17.
18 Investment income before allocation *(from Form CT-3-ATT, line 48, but not more than line 17 above)* .... 18.
19 Business income before allocation *(subtract line 18 from line 17)* .... 19.
20 Allocated investment income *(multiply line 18 by* % *from Form CT-3-ATT, line 31)* .... 20.
21 Allocated business income *(multiply line 19 by* % *from line 119, 121, or 141)* .... 21.
22 Total allocated income *(add lines 20 and 21)* .... 22.
23 Optional depreciation adjustments *(attach Form CT-324; enter here and on line 69)* .... 23.
24 Entire net income base *(line 22 plus or minus line 23)* .... 24.
25 Entire net income base tax *(multiply line 24 by the appropriate rate from the Tax rates schedule on page 5 of Form CT-3/4-I; enter here and on line 72)* .... 25.

Mail your return with payment to:
NYS CORPORATION TAX, PROCESSING UNIT, PO BOX 1909,
ALBANY NY 12201-1909
Mail your return without payment to:
NYS CORPORATION TAX, PROCESSING UNIT, PO BOX 22095,
ALBANY NY 12201-2095

41901030094

Generic columns → XML

Current relational storage, inefficient, anonymous columns, requires complex mappings in the application

col1	col2	col3	col4	col5	…	col1000
134	NULL	11/23/05	NULL	NULL		NULL
NULL	276	NULL	NULL	Yes	…	NULL
12	NULL	NULL	99.99	NULL	…	NULL
NULL	NULL	NULL	123.23	NULL	…	No

New XML format:

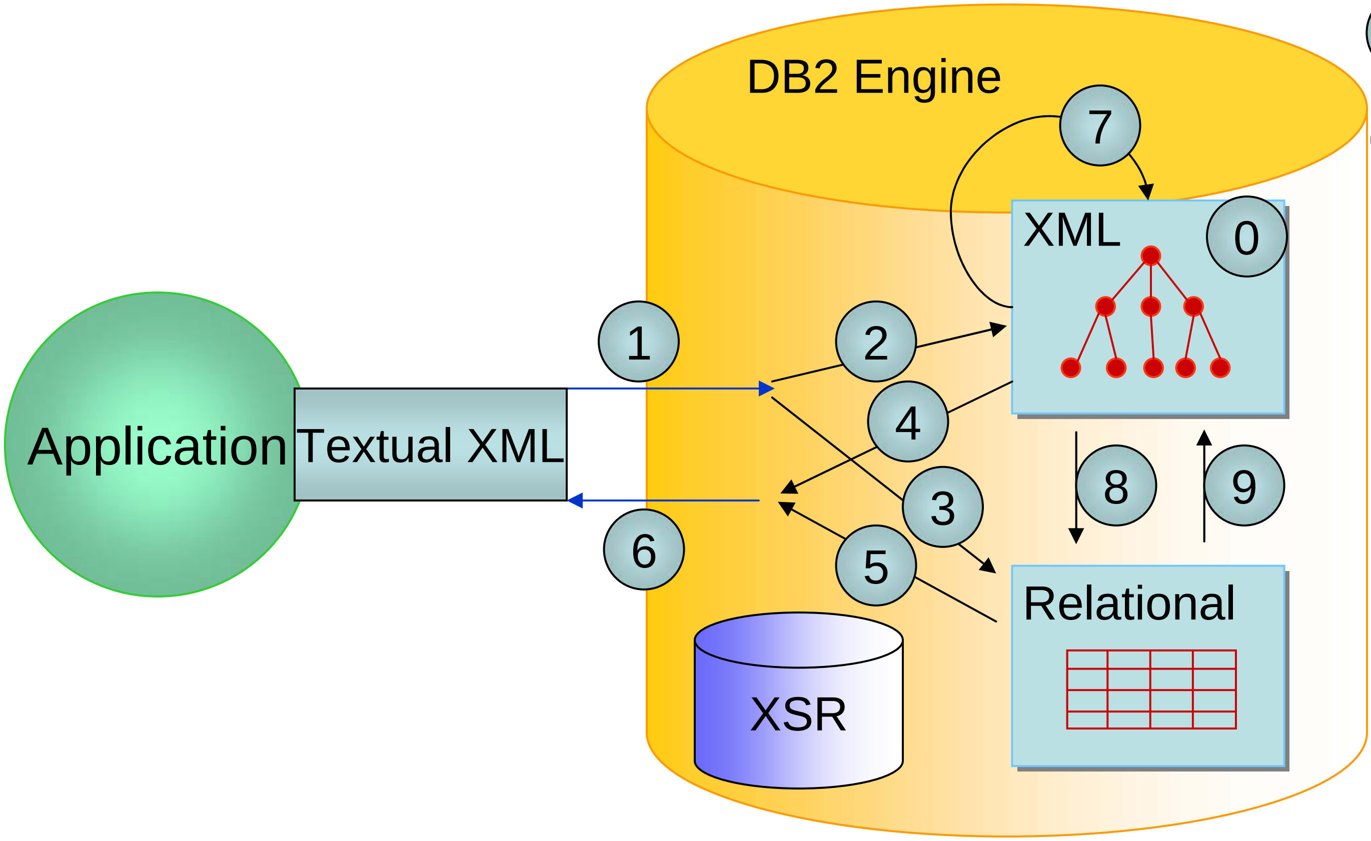
```
<form>
  <wages>134</wages>
  <date>11/23/05</date>
</form>
```

XML: Avoids sparsity. Proper data labeling. 2 columns, not 1000. Transformable. Extensible. Simplifies mapping.

Business Value of pureXML

- ***Lower Development Costs***
  - – Reduced system and development complexity
  - – Improved developer productivity
- ***Greater Business Agility***
  - – Easily accommodate changes to data and schemas
  - – Update applications rapidly and reduce maintenance costs
- ***Improved Business Insight***
  - – Access to information in otherwise unexploited documents
  - – Unprecedented application performance

Summary of SQL/XML Features



0 XML Storage

1 Bind in XML

2 Store as XML

3 Shred into Rel

4 Retrieve XML

5 Publish XML

6 Bind out XML

7 XML to XML

8 XML to Rel

9 Rel to XML

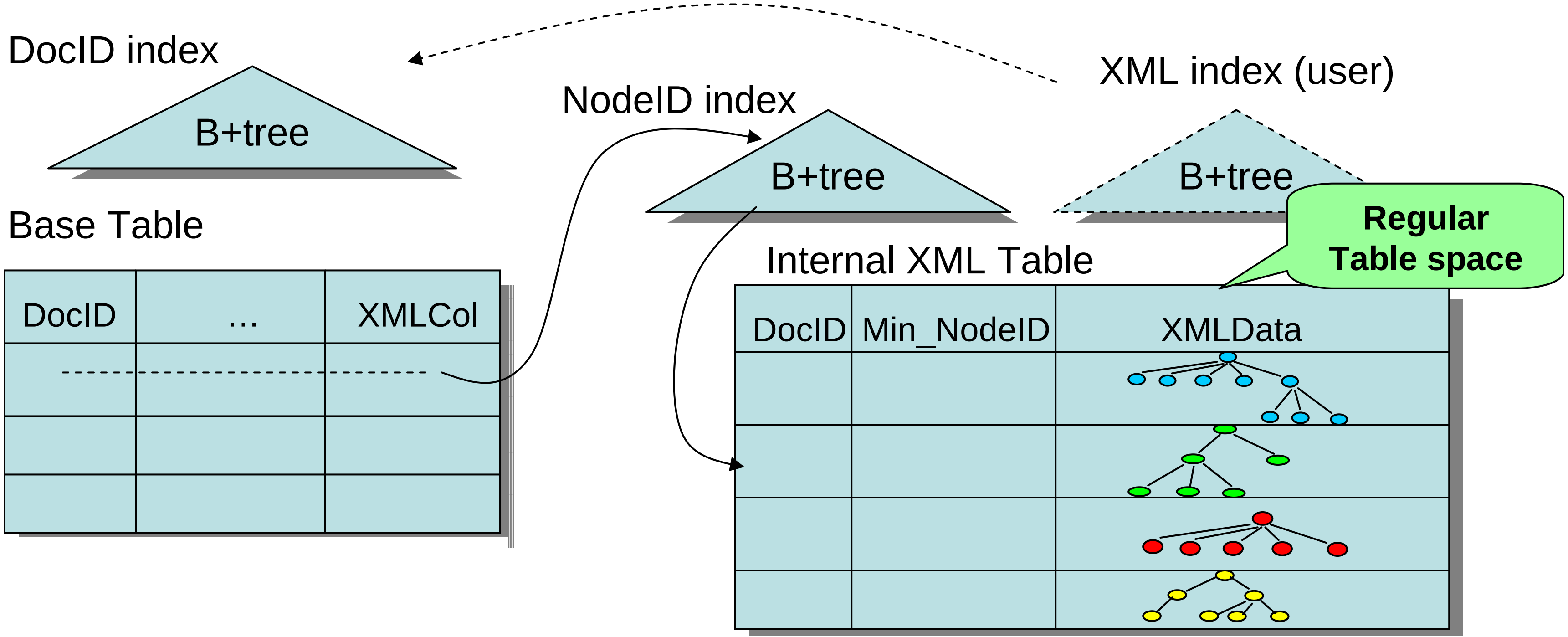
XML Type and DDL

```
CREATE TABLE PurchaseOrders (
  ponumber          varchar(10) not null,
  podate            date not null,
  status            char(1),
  XMLpo             xml);
    [or: IN MYDB.MYTS; ]
    [or: IN DATABASE MYDB; ]
    [or: IN MYTS; ]
```

- Hidden DocID column
- One DocID index
- Internal XML table (16K BP) for each XML column
- NodeID index
- No associated schema
- No length limit

```
CREATE TABLE PO LIKE PurchaseOrders;

CREATE VIEW ValidPurchaseOrders as
  SELECT ponumber, podate, XMLpo
      FROM PurchaseOrders
      WHERE status = ‘A’;

ALTER TABLE PurchaseOrders
    ADD revisedXMLpo xml;
```

XML Storage on Mature Infrastructure



A table with an XML column has a DocID column, used to link from the base table to the XML table.
A DocID index is used for getting to base table rows from XPath value indexes.

Each XMLData column is a VARBINARY, containing a subtree or a sequence of subtrees, with context path.
Rows in XML table are freely movable, linked with a NodeID index.

Storing XML Trees - Tree Packing

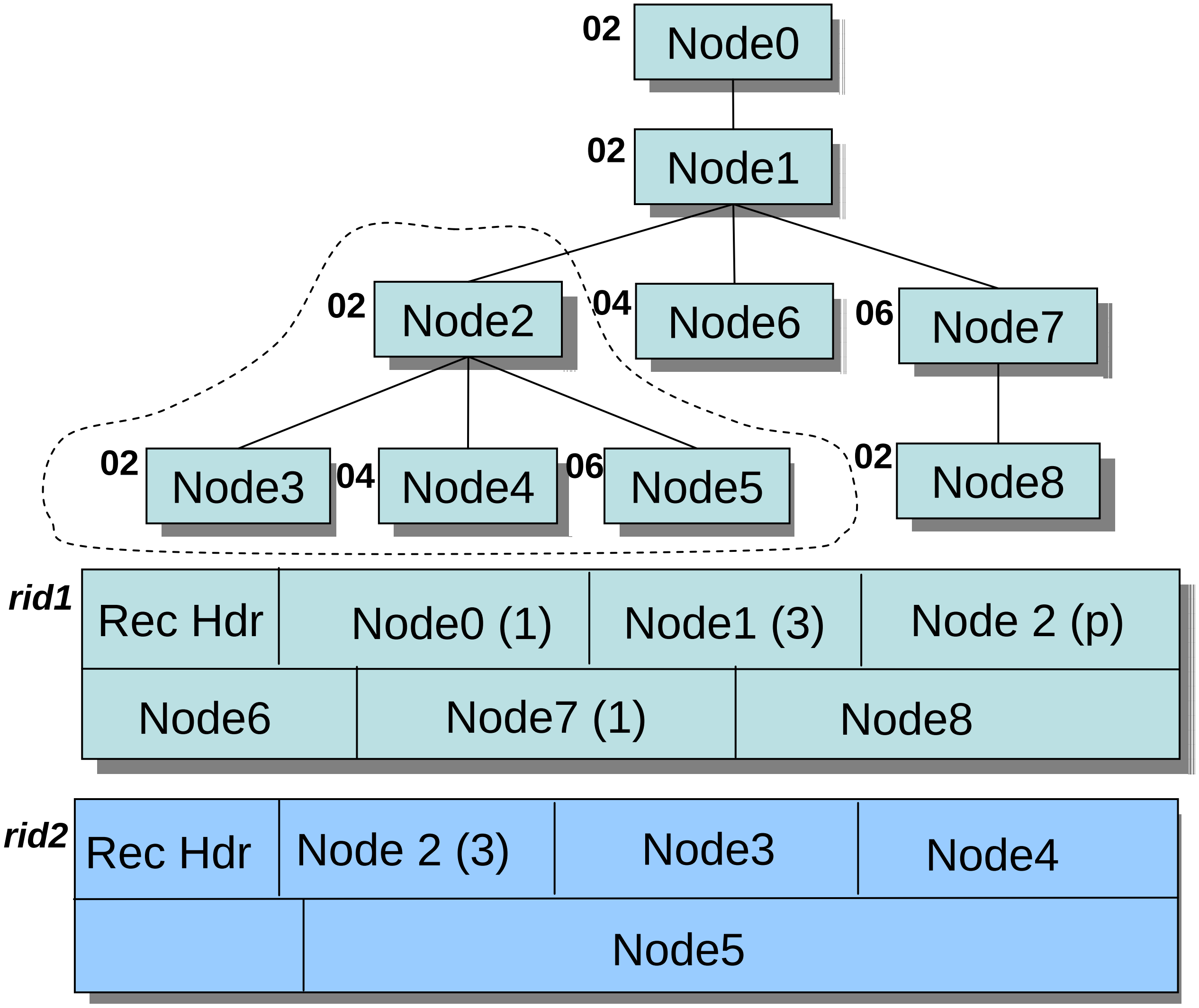
Each node contains local node id, length and optional number of children.

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.



rid1

Rec Hdr | Node0 (1) | Node1 (3) | Node 2 (p)

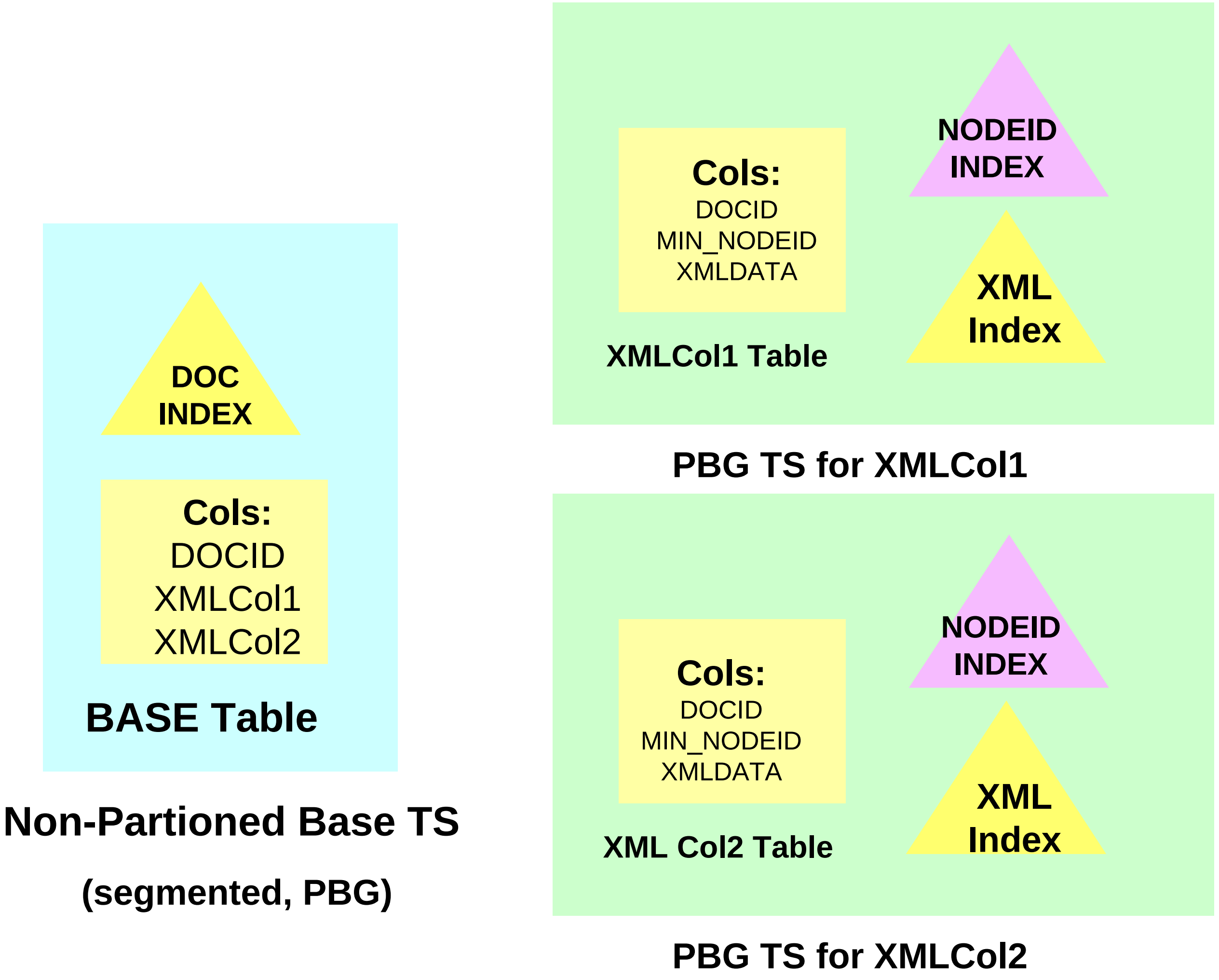
Node6 | Node7 (1) | Node8

rid2

Rec Hdr | Node 2 (3) | Node3 | Node4

Node5

XML objects for non-partitioned base table



DOC INDEX

Cols:
DOCID
XMLCol1
XMLCol2

BASE Table

Non-Partioned Base TS

(segmented, PBG)

Cols:
DOCID
MIN_NODEID
XMLDATA

XMLCol1 Table

NODEID INDEX

XML Index

PBG TS for XMLCol1

Cols:
DOCID
MIN_NODEID
XMLDATA

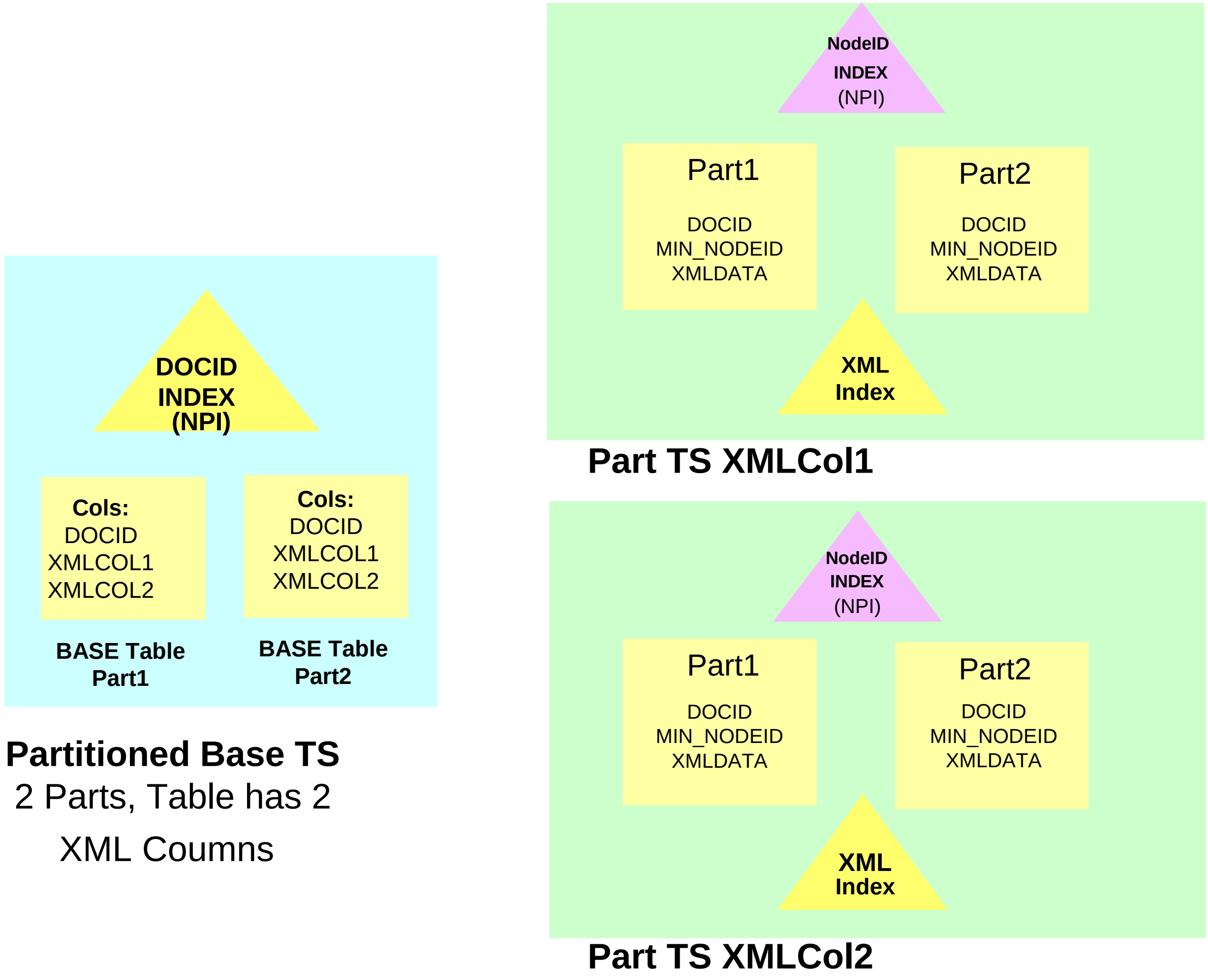
XML Col2 Table

NODEID INDEX

XML Index

PBG TS for XMLCol2

XML objects for partitioned base table



DOCID INDEX (NPI)

Cols: DOCID XMLCOL1 XMLCOL2

BASE Table Part1

Cols: DOCID XMLCOL1 XMLCOL2

BASE Table Part2

Partitioned Base TS
2 Parts, Table has 2
XML Coumns

NodeID INDEX (NPI)

Part1
DOCID
MIN_NODEID
XMLDATA

Part2
DOCID
MIN_NODEID
XMLDATA

XML Index

Part TS XMLCol1

NodeID INDEX (NPI)

Part1
DOCID
MIN_NODEID
XMLDATA

Part2
DOCID
MIN_NODEID
XMLDATA

XML Index

Part TS XMLCol2

Other DDLs Support

- ALTER TABLESPACES
  - Only some of the attributes on the implicit XML table space can be altered
  - LOG attribute updated on the base table spaces will be propagated to the XML tablespace.
- ALTER INDEX
  - Allowed on the implicit NodeID index
  - Not all attributes can be altered
- DROP TABLE
  - Not allowed on the implicit table
  - DROP TABLE of a table with XML columns will drop all the associated implicit objects for all the XML columns
- DROP INDEX
  - Not allowed on the implicit index
- DROP TABLESPACES
  - Not allowed on the implicit table space created

Manipulating XML Data

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

Host var of XML type

```
INSERT INTO PurchaseOrders VALUES (‘200300001’,
               CURRENT DATE, ‘A’, :xmlPo);
```

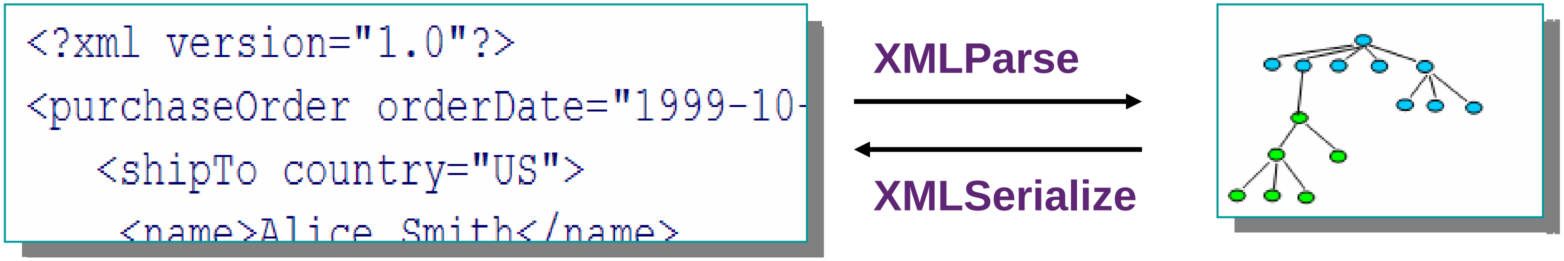
String literal is OK

```
INSERT INTO PurchaseOrders VALUES (‘200300001’,
               CURRENT DATE, ‘A’, CAST(? AS XML));

INSERT INTO PurchaseOrders VALUES(‘200300003’, CURRENT DATE,
    ‘A’, XMLPARSE(DOCUMENT :vchar PRESERVE WHITESPACE) );

INSERT into PurchaseOrders VALUES( '200300004', CURRENT DATE, 'A',
    DSN_XMLValidate(:lobPo, ’SYSXSR.myPOSchema’));

UPDATE PurchaseOrders SET XMLpo = :XMLpo_revised
         WHERE ponumber = ‘12345’;
```

Whole document replacement

```
DELETE FROM PurchaseOrders WHERE ponumber = ‘12345’;
```

XMLParse and XMLSerialize



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

XMLParse

XMLSerialize

- XMLParse
  - Allows strip whitespace or preserve whitespace
  - Implicit XMLParse applies for bind-in XML hostvar or inserting hostvar or string literal.
- XMLSerialize
  - With or without XML declaration
  - Implicit XMLSerialize applies for bind-out XML type

XMLParse: Textual to XDM

- XML host vars: Implicit XMLPARSE only
  - STRIP WHITESPACE always
- Parameter marker: CAST(? As XML), similar to XML host var.
- Non-XML host vars or string literals: Implicit XMLPARSE if inserting into or setting XML columns (DB2 detects from context)
- Explicit XMLPARSE: non-XML host vars or expressions only.
  - Can specify PRESERVE WHITESPACE option
  - Cannot use on an XML host var or column.

Retrieving XML Data

- Simple select:

```
SELECT XMLpo INTO :xmlPo
FROM PurchaseOrders
WHERE ponumber = ‘200300001’;
```

- Select with condition:

```
SELECT XMLPO
FROM PurchaseOrders
WHERE XMLEXISTS(‘//items/item[desc = “Shoe”]’   PASSING
XMLpo);
```

- Extract from a document:

```
SELECT XMLQUERY(‘//items/item/quantity’   PASSING XMLpo)
FROM PurchaseOrders WHERE …;
```

Use Square Brackets [ ] in XMLESISTS

- Use square brackets [ ] to avoid Boolean expressions in XMLEXISTS, which will always return true (for non-empty Boolean value true or false).

```
SELECT XMLPO
FROM PurchaseOrders
WHERE XMLEXISTS('/purchaseorder/items/item/desc = "Shoe"'
                PASSING XMLPO);
```

- Use this to avoid surprise:

```
SELECT XMLPO
FROM PurchaseOrders
WHERE XMLEXISTS('/purchaseorder/items/item[desc = "Shoe"]'
                PASSING XMLPO);
```

XMLSERIALIZE: XML to Textual

- To XML or non-XML host vars: Implicit XMLSERIALIZE
  - INCLUDING XMLDECLARATION always (except for Java)
    <?xml version="1.0" encoding=“xxxx"?>
  - Encoding is consistent with character host var CCSID.
  - Binary host var: always UTF-8
- Explicit XMLSERIALIZE: result is LOB:
  - CLOB, DBCLOB: encoding conversion based on CCSID
  - BLOB: always UTF-8
  - Default: EXCLUDING XMLDECLARATION
  - Can specify INCLUDING XMLDECLARATION option
  - Does not work to XML host vars

Encoding for XML

- Textual XML Data Encoding
  - Internally-encoded (within XML data itself)
    - Encoding Declaration option or **B**yte **O**rder **M**ark within XML data
    - Default: UTF-8
    - Applies to binary variables (DB2 detects encoding)
  - Externally-encoded
    - Character host variables CCSID (override internal encoding)
- DB2 uses UTF-8 to handle XML data. Character data in XML always stored as UTF-8 in XML column.
- During insert, DB2 converts XML data to UTF-8 from the corresponding CCSID.
- During select, DB2 converts XML in UTF-8 to host var encoding locally, or sends serialized XML data in UTF-8 format to remote requester.

Join Queries

- PO’s containing a product with name from PRODUCT.NAME

```
SELECT PRODUCT.NAME, XMLPO
FROM PurchaseOrders, Product
WHERE XMLEXISTS(‘//items/item[desc = $n]’   PASSING XMLpo,
    Product.name as “n”);
```

- PO’s containing a product with price > :price

```
SELECT XMLPO
FROM PurchaseOrders, Catalog
WHERE XMLEXISTS(‘//items/item[desc=$x]’ PASSING XMLpo,
    XMLQUERY(‘/category/product[price > $y]/name’ PASSING
    XCategory, :price as “y”) as “x”) AND
   XMLEXISTS(‘/category/product[price > $y]’ PASSING XCategory,
    :price as “y”)
```

XMLTable – Processing XML with SQL Power

```
SELECT TX.*
FROM PurchaseOrders PO,
     XMLTable (‘/purchaseorder/items/item’
        PASSING PO.XMLpo
        COLUMNS
         “Part #”          CHAR(6)         PATH ‘@partnum’,
         “Product Name”    CHAR(20)        PATH ‘productName’,
         “Quantity”        INTEGER         PATH ‘quantity’,
         “US Price”        DECIMAL(9,2)    PATH ‘USPrice’,
         “Ship Date”       DATE            PATH ‘shipDate’,
         “Comment”         CHAR(80)        PATH ‘comment’
          WITH ORDINALITY "Seqno") AS TX
WHERE PO.ponumber = ‘200300001’;
```

XMLTable function will be delivered after V9 GA

Leveraging the Power of SQL

```
CREATE VIEW ORDER_VIEW AS
SELECT PO.POID, X.*
FROM PurchaseOrders PO,
      XMLTABLE( '//item' PASSING PO.XMLPO
        COLUMNS  "orderDate"        DATE PATH '../../@orderDate',
                 "shipTo City"      VARCHAR(20) PATH '../../shipTo/city',
                 "shipTo State"     CHAR(2) PATH '../../shipTo/state',
                 "Part #"           CHAR(6) PATH '@partnum',
                 "Product Name"     CHAR(20) PATH 'productName',
                 "Quantity"         INTEGER PATH 'quantity',
                 "US Price"         DECIMAL(9,2) PATH 'USPrice',
                 "Ship Date"        DATE PATH 'shipDate',
                 "Comment"          VARCHAR(60) PATH 'comment' ) AS X;
```

```
SELECT "Product Name", "shipTo State",
         SUM("US Price" * "Quantity") AS TOTAL_SALE
FROM ORDER_VIEW
GROUP BY "Product Name", "shipTo State";
```

```
SELECT "shipTo City", "shipTo State",
         RANK() OVER(ORDER BY SUM("Quantity")) AS SALES_RANK
FROM ORDER_VIEW
WHERE "Product Name" = 'Baby Monitor'
GROUP BY "shipTo State", "shipTo City"
ORDER BY SALES_RANK;
```

SQL/XML Constructors

- Construct XML from relational data
  - XMLElement, XMLAttributes, XMLNamespaces, XMLForest, XMLConcat, XMLAGG (V8)
  - Support Binary data types and null handling options(V9)
  - XMLText, XMLPI, XMLComment, XMLDocument (V9)
- Construct new document by extracting parts from an existing document (XMLQuery, XMLTABLE) and other data.

Constructor Example

```
SELECT XMLDOCUMENT(
             XMLELEMENT(NAME “hr:Department",
              XMLNAMESPACES(‘http://example.com/hr’ as “hr”),
              XMLATTRIBUTES (e.dept AS "name" ),
              XMLCOMMENT(‘names in alphabetical order’),
              XMLAGG(XMLELEMENT(NAME “hr:emp", e.lname)
                              ORDER BY e.lname  )
                            ) ) AS "dept_list”
FROM employees e
GROUP BY dept;
```

Can construct XHTML for web pages

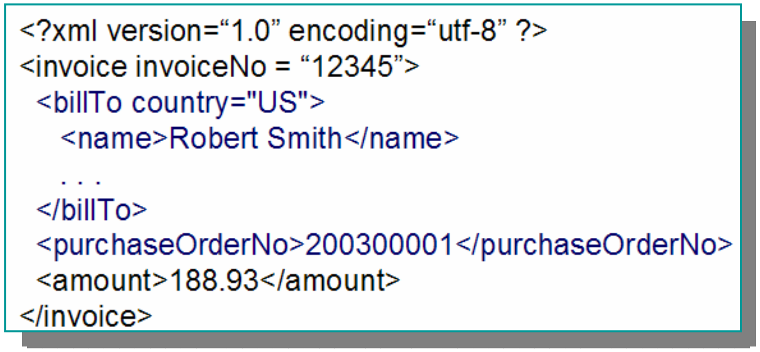
```
<?xml version=“1.0” encoding=“UTF-8”>
<hr:Department xmlns:hr=“http://example.com/hr”
   name="Shipping">
  <!-- names in alphabetical order -->
  <hr:emp>Lee</hr:emp>
  <hr:emp>Martin</hr:emp>
  <hr:emp>Oppenheimer</hr:emp>
</hr:Department>
```

Use SQL/XML to Achieve XQuery Functionality

- Use XMLEXISTS with XPath to find documents.
- Use XMLQuery and XMLTable with XPath to extract parts of documents.
- XPath cannot be used to construct new document.
- SQL/XML has complete constructor functions to make up missing functionality in XPath.
- Use SQL/XML constructor functions and XMLQuery (and XMLTable) to construct new documents from existing documents.

Example: Construct Invoice from Purchase Order

```
SELECT XMLDocument(
    XMLElement(NAME "invoice",
      XMLAttributes( '12345' as "invoiceNo"),
        XMLQuery ('/purchaseOrder/billTo' PASSING xmlpo),
        XMLElement(NAME “purchaseOrderNo”,
             PO.ponumber)
        XMLElement(NAME "amount",
         XMLQuery
          ('fn:sum(/purchaseOrder/items/item/xs:decimal(USPrice))'
            PASSING xmlpo) )
        )     )
FROM PurchaseOrders PO
WHERE PO.ponumber = ‘200300001’;
```



```
<?xml version=“1.0” encoding=“utf-8” ?>
<invoice invoiceNo = “12345”>
  <billTo country="US">
    <name>Robert Smith</name>
    . . .
  </billTo>
  <purchaseOrderNo>200300001</purchaseOrderNo>
  <amount>188.93</amount>
</invoice>
```

XPath Support

- Used in XMLEXISTS, XMLQUERY, XMLTABLE and XML indexing
- XPath 1.0 + - (subset of XPath 2.0/XQuery 1.0)
  - XPath 1.0 constructs in XPath 2.0 semantics
  - + more data types: xs:boolean, xs:integer, xs:decimal, xs:double, xs:string
  - + namespace declaration from XQuery prolog
  - - Axes: only forward axes (child, attribute, descendant, descendant-or-self, self, ., //, @) & parent axis (..) are supported.
- All stored XML data are untyped in V9.
  - Explicit type casting may be needed in some cases

Examples of XPath - Typing

- No cast is needed: “Find all the products in the Catalog with RegPrice > 100”
  XMLQUERY(‘/Catalog/Categories/Product[RegPrice > 100]’ PASSING XCatalog)

- Cast is needed: “Find all the products on sale in the Catalog”
  XMLQUERY(‘/Catalog/Categories/Product[RegPrice > xs:double(SalePrice)]’ PASSING XCatalog)

- No cast is needed: “Find all the products with more than 10% discount in the Catalog”
  XMLQUERY(‘/Catalog/Categories/Product[RegPrice * 0.9 > SalePrice]’ PASSING XCatalog)

Examples of XPath - Cardinality

- No cardinality problem: “Find all the products in the Catalog with RegPrice > $price”
  XMLQUERY(‘/Catalog/Categories/Product[RegPrice > $price]’ PASSING XCatalog, 200 as “price”)

- To avoid cardinality violation: “Find all the products on sale in the Catalog”
  XMLQUERY(‘/Catalog/Categories/Product[RegPrice > SalePrice/xs:double(.))]’ PASSING XCatalog)

- To avoid cardinality violation: “Find all the products with more than 10% discount in the Catalog”
  XMLQUERY(‘/Catalog/Categories/Product[RegPrice/(. * 0.9) > SalePrice]’ PASSING XCatalog)

Application Interfaces

- XML type is supported in
  - Java (JDBC, SQLJ), ODBC,
  - C/C++, COBOL, PL/I, Assembly
  - .NET
- Applications use:
  - XML as CLOB(n), XML as CLOB_FILE
  - XML as DBCLOB(n), XML as DBCLOB_FILE
  - XML as BLOB(n), XML as BLOB_FILE
  - All character or binary string types are supported
- XMLParse and XMLSerialize apply (implicitly or explicitly)

SQLDA and DCLGEN

- SQL type value 988 for XML data
- SQL type value 989 for nullable XML data
- Output from DCLGEN for an XML column is as follows:
  - PL/I: SQL TYPE IS XML AS CLOB(1M);
  - C/C++: SQL TYPE IS XML AS CLOB(1M);
  - COBOL: SQL TYPE IS XML AS CLOB(1M).

Host Interface Examples

```
CREATE TABLE T1(ID INT, XMLCOL XML);

INSERT INTO T1 VALUES(100, :XMLHV);
INSERT INTO T1 VALUES(150, :VBHV);
INSERT INTO T1 VALUES(200, XMLPARSE(DOCUMENT :VBHV) );
INSERT INTO T1 VALUES(210, XMLPARSE(DOCUMENT :VBHV
                                        PRESERVE WHITESPACE) );

SELECT XMLCOL INTO :XMLHV
FROM T1
WHERE T1.ID = 100;

SELECT XMLCOL INTO :VBHV
FROM T1
WHERE T1.ID = 100;

SELECT XMLSERIALIZE(XMLCOL AS BLOB(100K))
     INTO :BLOBHV
FROM T1
WHERE T1.ID = 200;
```

Java JDBC Example

Use standard interface:

```
PreparedStatement pstmt = connection.prepareStatement("INSERT INTO
      PurchaseOders VALUES(?, ?)");     // second column: XML type
…
InputStream fin = new FileInputStream(file);
pstmt.setBinaryStream( 2, fin, flen );
pstmt.execute();

Statement s = connection.createStatement();
ResultSet rs = s.executeQuery ("select ponumber, xmlpo from purchaseOrders");
while (rs.next()) {
  int po_no = rs.getInt ("ponumber");
  String spo= rs.getString(2);
  System.out.println (spo); // uninterpreted flat xml text
}
```

Or use com.ibm.db2.jcc.DB2Xml interface

FETCH CONTINUE for XML and LOB

- No size associated with XML values
- Hard to allocate large memory
- Shortcomings with LOB Locator
- New FETCH CONTINUE statements: (one of two ways)
  - DECLARE CURSOR1 CURSOR FOR SELECT C2 FROM T1;
  - OPEN CURSOR1;
  - FETCH WITH CONTINUE CURSOR1 into :clobhv;
  - if (sqlcode >= 0) & sqlcode <> 100
  - Loop if truncation occurs until lob/xml complete (total length)
  - FETCH CURRENT CONTINUE CURSOR1 into :clobhv;
  - Consume :clobhv content
  - end loop
- Another way is to use FETCH … INTO DESCRIPTOR :SQLDA

XML Data Exchange in DRDA

- Support for DRDA XML data access
  - DB2 for z/OS V9 (ODBC driver support is provided)
  - DB2 Universal Database for Linux, Unix, and Windows Version 9.1 (including CLI support).
  - DB2 Universal Driver (JDBC, SQLJ) V3.1
- Transport is also in serialized XML format
- Encoding either within XML value itself (DRDA internally-encoded) or in DRDA descriptor (DRDA externally-encoded)
- Sending input data from DRDA Requester to Server
  - BLOB as host variable
    - Data is in DRDA internally encoded string value
  - DBCLOB, CLOB as host variables
    - Data is in the DRDA externally encoded string value , i.e. CCSID in DRDA descriptor
  - Server side converts XML string value to UTF-8
- Sending output data from DRDA Server to Requester
  - Always with UTF-8 DRDA externally encoded
  - Requester side converts XML string value into application host variable CCSID
- Separate send/receive for XML data if remote result set contains XML data

XML Indexes

- XPath value index: index values of elements or attributes inside a document.
- Index entries include:
(key value, DocID, NodeID, RIDx)
- Support string (VARCHAR) or numeric (DECFLOAT) key type

```
CREATE INDEX ON
PurchaseOrders(XMLPO)
Generate Keys Using
XMLPATTERN
'/purchaseOrder/items/item/desc'
as SQL VARCHAR(100);
```

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
   <name>Alice Smith</name>
   . . .
  </shipTo>
  <billTo country="US">
   <name>Robert Smith</name>
   . . .
  </billTo>
  <comment>Hurry, my lawn is going wild!</comment>
  <items>
   <item partNum="872-AA">
    <desc>Lawnmower</desc>
    <quantity>1</quantity>
    <USPrice>148.95</USPrice>
    <comment>Confirm this is electric</comment>
   </item>
   <item partNum="926-AA">
    <desc>Baby Monitor</desc>
    <quantity>1</quantity>
    <USPrice>39.98</USPrice>
    <shipDate>2003-05-21</shipDate>
   </item>
  </items>
</purchaseOrder>
```

This index can be used for predicate:
XMLEXISTS('/purchaseOrder/items/item[desc = "Baby Monitor"]' passing XMLPO)

Something Special for XML Index

- The number of keys for each document (each base row) depends on the document and XMLPattern.
- For a numeric index, if a string from a document cannot be converted into a number, it is ignored.
  - <a><b>X</b><b>5</b></a>, XMLPattern ‘/a/b’ as SQL Decfloat. Only one entry ‘5’ in the index.
- For a string (VARCHAR(n)) index, if a key value is longer than the limit, INSERT or CREATE INDEX will fail.

Be Careful Creating Indexes on Non-leaf Nodes

- Indexing everything is not supported. SQLCODE -20305 if a key value spans multiple rows in the internal XML table.
- Always safe to index leaf nodes (<=1000 bytes).
- Indexing non-lead nodes will result in concatenation of values for the key: /…/name, the key value will be “JohnJoe” with strip whitespace, or
“ John   Joe ” (w/ LFs) with preserve whitespace
  - Query has to use the same string to match: …/customer[name = “JohnJoe”]

```
<name>
  <first>John</first>
  <last>Joe</last>
</name>
```

New Access Methods

Access Methods	Description
DocScan "R" (QuickXScan)	Base algorithm: given a document, scan and evaluate XPath
DocID list access "DX" unique DocID list from an XML index, then access the base table and XML table.	XMLExists('/Catalog/Categories/Product[RegPrice > 100]' passing catalog) with index on '/Catalog/Categories/Product/RegPrice' as SQL DECFLOAT
DocID ANDing/ORing "DX/DI/DU" union or intersect (unique) DocID lists from XML indexes, then access the base table and XML table.	XMLExists('/Catalog/Categories/Product[RegPrice > 100 and Discount > 0.1]' …) With indexes on: '//RegPrice' as SQL DECFLOAT and '//Discount' as SQL DECFLOAT

Another Example for Indexes and Query

```
CREATE TABLE ACORD.REQUEST (
 ID          BIGINT NOT NULL PRIMARY KEY,
 REQUESTXML  XML,
 RESPONSEXML XML
) IN DATABASE DBACORD

CREATE INDEX ACORD.ACORDINDEX1 ON ACORD.REQUEST(REQUESTXML)
GENERATE KEYS USING XMLPATTERN
'declare default element namespace "http://ACORD.org/Standards/Life/2";
 /TXLife/TXLifeRequest/TransRefGUID' as SQL VARCHAR(24)

CREATE INDEX ACORD.ACORDINDEX2 ON ACORD.REQUEST(REQUESTXML)
GENERATE KEYS USING XMLPATTERN
'declare default element namespace "http://ACORD.org/Standards/Life/2";
 /TXLife/TXLifeRequest/OLifE/Holding/Policy/@id' AS SQL VARCHAR(9)
```

Another Example for Indexes and Query (cont’ed)

Find participant information about a policy.

```
SELECT
   XMLQuery('declare default element namespace "http://ACORD.org/Standards/Life/2";
         /TXLife/TXLifeRequest/OLifE/Holding/Policy/Life/Coverage/LifeParticipant'
         PASSING R.REQUESTXML),
   XMLQuery('declare default element namespace "http://ACORD.org/Standards/Life/2";
         /TXLife/TXLifeRequest/OLifE/Party
          [@id =
           /TXLife/TXLifeRequest/OLifE/
             Holding/Policy/Life/Coverage/
              LifeParticipant/@PartyID ] '
         PASSING R.REQUESTXML)
FROM  ACORD.REQUEST R
WHERE XMLExists('declare default element namespace
                  "http://ACORD.org/Standards/Life/2";
         /TXLife/TXLifeRequest[TransRefGUID="2004-1217-141016-000012"]/
          OLifE[Holding/Policy/@id="POLICY12"]'
        PASSING R.REQUESTXML)
```

```
   +---------------------------------------------------------------------------------------+
   |  PLANNO    | ACCESSTYPE | MATCHCOLS | ACCESSCREATOR |     ACCESSNAME      | MIXOPSEQ  |
   +---------------------------------------------------------------------------------------+
 1_|          1 | M          |         0 |               |                     |         0 |
 2_|          1 | DX         |         1 | ACORD         | ACORDINDEX2         |         1 |
 3_|          1 | DX         |         1 | ACORD         | ACORDINDEX1         |         2 |
 4_|          1 | DI         |         0 |               |                     |         3 |
   +---------------------------------------------------------------------------------------+
```

A SEPA Query Example

- Assume table INTERBANKDD(CollectionMsg XML, StatusMsg XML)
  - CollectionMsg: pacs.003.001.01
  - StatusMsg: pacs.002.001.02
- Find information of Direct Debit Transactions that were created on or after 2006-06-28 and have been rejected, and return the reason also.

```
SELECT XMLQuery(‘declare default element namespace
    "urn:iso:std:iso:20022:tech:xsd:pacs.003.001.01";
    /Document/pacs.003.001.01/DrctDbtTxInf' PASSING CollectionMsg),
    XMLQuery(‘declare default element namespace
    "urn:iso:std:iso:20022:tech:xsd:pacs.002.001.02";
    /Document/pacs.002.001.02/TxInfAndSts/StsRsnInf/StsRsn' PASSING
    StatusMsg)
FROM INTERBANKDD
WHERE XMLEXISTS('declare default element namespace
    "urn:iso:std:iso:20022:tech:xsd:pacs.002.001.02";
    /Document/pacs.002.001.02[OrgnlGrpInfAndSts/OrgnlCreDtTm >= "2006-
    06-28"]/TxInfAndSts[TxSts = "RJCT"]' PASSING STATUSMsg)
```

XML Schema Support

- XML Schema adds constraints on XML data.
- Register a schema in XML Schema Repository (XSR)
- External names
  - target namespace: e.g., "http://www.ibm.com/software/catalog"
  - schema location: e.g., "http://www.ibm.com/schemas/software/catalog.xsd"
- SQL identifier - used to reference schemas in SQL
  - unique identifier in DB, e.g., SYSXSR.ORDERSCHEMA
- Where are schemas used?
  - SYSFUN.DSN_XMLValidate in SQL (UDF for XMLValidate)
  - Decomposition

Registering an XML Schema (Stored Procedures)

- XSR_REGISTER (rschema, name, schemalocation, xsd, docproperty)
- XSR_ADDSCHEMADOC (rschema, name, schemalocation, xsd, docproperty)
- XSR_COMPLETE (rschema, name, schemaproperties, isUsedForDecomp)
- XSR_REMOVE(rschema, name)

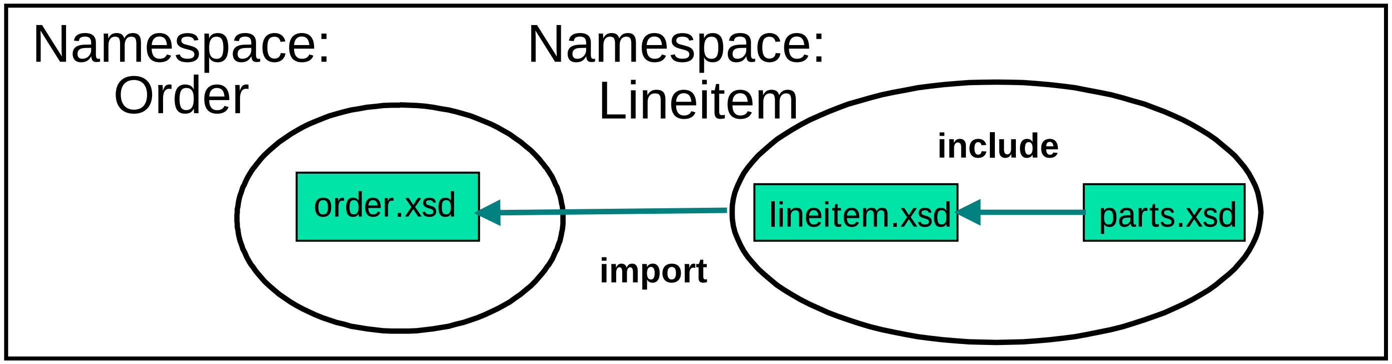
- Parameters:

rschema	– null or ‘SYSXSR’;
identifier	– SQL name (VARCHAR(128));
schemalocation	– VARCHAR(1000);
xsd	– XML schema document (BLOB(30M));
docproperty	– BLOB(5M), may be used by tools;
schemaproperties	– same as docproperties
isUsedForDecomp	– INTEGER, 1 yes, 0 no.

- Java Driver (JCC) provides a set of APIs for schema registration

Example: Registering an XML Schema

Orderschema



- XSR_REGISTER(‘SYSXSR', 'ORDERSCHEMA', 'http://www.n1.com/**order.xsd**', :xsd, :docproperty)
- XSR_ADDSCHEMADOC(‘SYSXSR', 'ORDERSCHEMA', 'http://www.n1.com/**lineitem.xsd**', :xsd, :docproperty)
- XSR_ADDSCHEMADOC(‘SYSXSR', 'ORDERSCHEMA', 'http://www.n1.com/**parts.xsd**', :xsd, :docproperty)
- XSR_COMPLETE (‘SYSXSR', 'ORDERSCHEMA', :schemaproperty, 0)

Using XML Schema

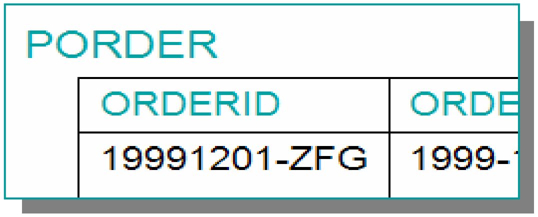
- Schema validation (type annotation not kept)

```
INSERT into PurchaseOrders
VALUES( '200300001', CURRENT DATE, 'A',
SYSFUN.DSN_XMLValidate(:lobPO,’SYSXSR.ORDERSCHEMA’
));
```

- Annotated schema-based decomposition – store using tables. (XDBDECOMPXML stored proc)
  E.g. orderID ->PORDER.ORDERID

```
<attribute name="orderID" type=“xs:string”
      db2-xdb:rowSet =  “PORDER”
      db2-xdb:column= “ORDERID” />
```

annotations



PORDER

ORDERID	ORDE
19991201-ZFG	1999-

Utilities

- Enhanced to handle new XML type, XML tablespaces, and XML indexes
- 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

DBA Considerations

- Database administrators should treat XML database objects as they do in LOB database objects.
- Like LOB objects, XML objects contain data stored outside the base table space.
  - XML table spaces and index spaces must be consistent also with their related base table
- All the utilities either support or tolerate XML, with some specific XML keyword support (next chart).
- In addition, XML has more considerations, such as table space size limit, compression, and indexes.

Get XML Data In with LOAD

- To load XML directly from input record, specify XML as the field type.

  -
    ```
    LOAD DATA INDDN(INFILE)  LOG NO RESUME(NO)
    FORMAT DELIMITED
         INTO TABLE PURCHASEORDERS
    ```

  -
    ```
    LOAD DATA INDDN(INFILE)  LOG NO RESUME(NO)
    …   XMLPO POSITION(20)
        XML PRESERVE WHITESPACE
         INTO TABLE PURCHASEORDERS
    ```
- To load XML from a file, specify CHAR or VARCHAR along with either BLOBF, CLOBF or DBCLOBF.
- Schema validation not supported for LOAD.
- XML compression takes effect after first REORG, not on initial LOAD. Same for FREEPAGE, PCTFREE.

Get XML Data Out with UNLOAD

- To unload XML data directly to output record, specify XML as the field type.
  - non-delimited format: a 2-byte length will precede the value of the XML.
  - For delimited output, no length field is present.
- To unload XML data to a separate file:
  - Specify CHAR(n)/VARCHAR(n) BLOBF, CLOBF or DBCLOBF for file names
  - Use the template control statement to create the XML output file and filename

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 Time 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.
- If there is an availability issue with one object in the related set, availability of the others may be impacted.

Diagnosing Problem related to XML objects

- Identify XML tables and their related objects
  - Run REPORT TABLESPACESET
- Validate that the auxiliary index is consistent with the underlying table spaces
  - Run CHECK INDEX on all indexes, DocID, NodeID and XML value indexes
- Validate the logical connection between the base table and XML table.
  - Run CHECK DATA against the base table space.
- Use Repair to diagnose problem related to base table spaces with XML columns and their DocID index
  - Use REPAIR LOCATE KEY to locate a row using DocID key in the DocID index
- Use Repair to diagnose problem related to XML table spaces and their NodeID index or XML Value Index
  - Use REPAIR LOCATE RID to locate a row using a RID.

Performance Monitoring and Tuning

- Since XML native support is built on top of regular tablespace structure, there are no special changes in DB2 Performance Expert to support XML other than minor points - such as new XML locks.
- XML performance problem can be analyzed through accounting traces and performance traces.
- There is a new LOAD MODULE for XML: DSNNXML
- XML indexes have the same consideration as other indexes.
- The REORG utility should be used to maintain order and free space.
- Run RUNSTATS for statistics to help pick XML indexes.

Table Space Size Consideration

- XML storage is about 1:1 original doc size without compression and with strip whitespace.
- An XML table space always use 16KB pages.
  - For non range-partitioned base table spaces, PBG table space is used for XML.
- Range-partitioned base table spaces: XML partitioning follows base table partitioning.
- The number of rows to fit into a relational partition is limited by the number of documents to fit into an XML partition.
  - For example, 4K doc size, 40GB partition can roughly store 10M documents (or 7.5M to be safe).

Compression

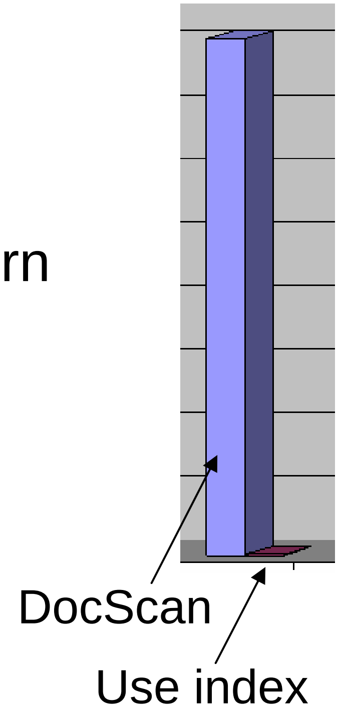
- XML Tablespace compression
  - Inherit from base table Compress YES
- Significant disk storage savings, especially with preserve whitespace option (70%)
- CPU cost similar to relational
  - Significant CPU impact if you select many document (DocScan)
- Initial LOAD will trigger base table compression but not for XML tablspace
- Compression happens at next REORG

XML Indexing

- Each index adds 15-20% CPU time to the basic INSERT cost. Create indexes that are only needed.
- Specify full path for index XML patterns, avoid wild card, or descendant axis
- Rebuild index is recommended over create index
- Code XPath conditions that will match index patterns.

Index Exploitation

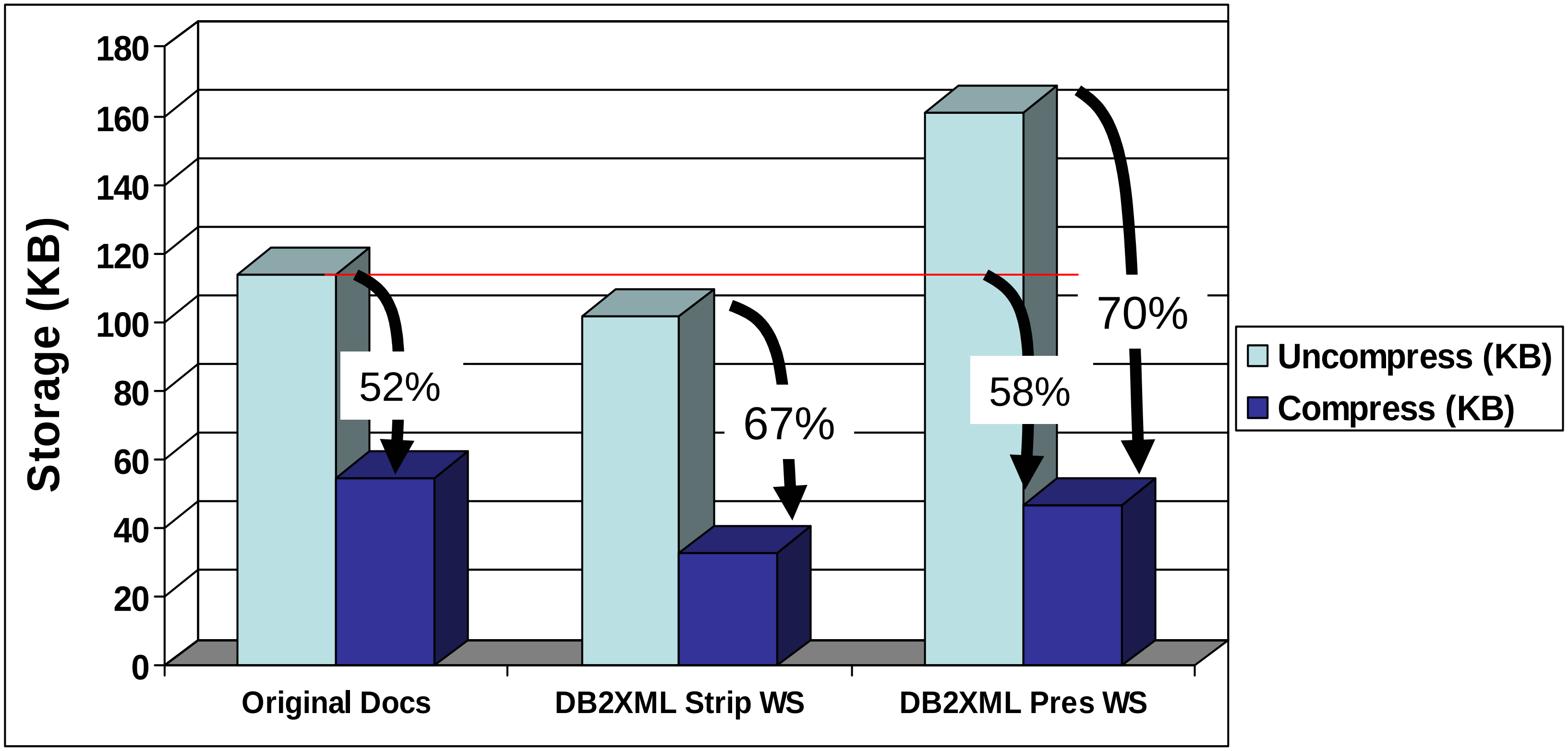
- Indexes are critical for query performance.
- Indexes are not used for the XMLQuery funtion.
- Indexes are used for XMLExists and XMLTable.
- /po/items/item[price > 123.5] can match XMLPattern /po/items/item/price
- /po[billTo/city="London"]/items/item[price > 123.5] matches two indexes:
  /po/billTo/city and
  /po/items/item/price
  (also //city and //price - not recommended)



Performance and Scalability

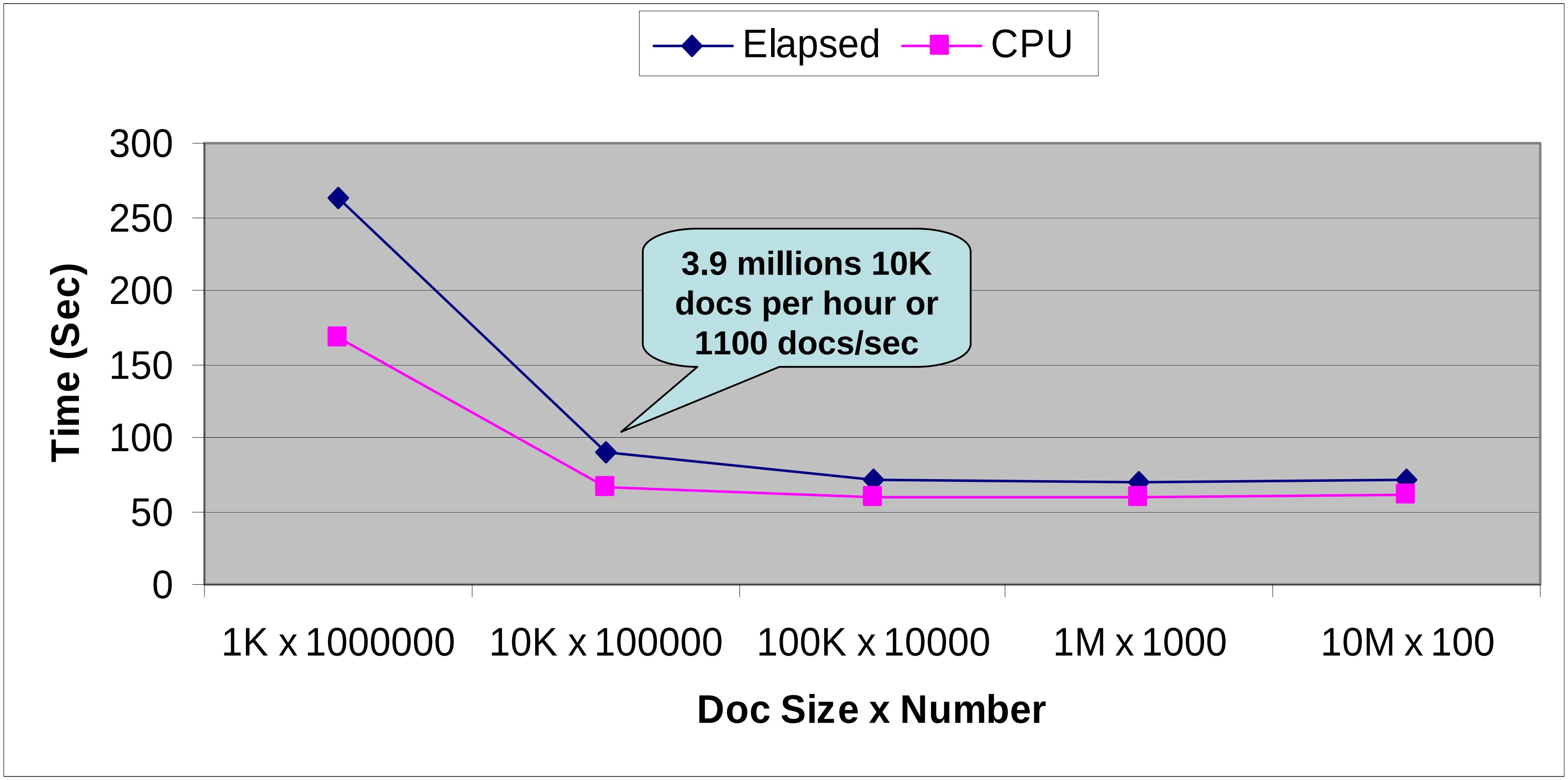
- XML storage is very compact, and it leverages mature optimized storage infrastructure (regular table spaces).
- Next-generation parsers: z/OS XML System Services and XLXP-C.
- Highly efficient XPath streaming algorithm
- Support partitioned table spaces and data sharing.
- Initial sweet spot: a large number of small documents.

Storage for UNIFI Messages



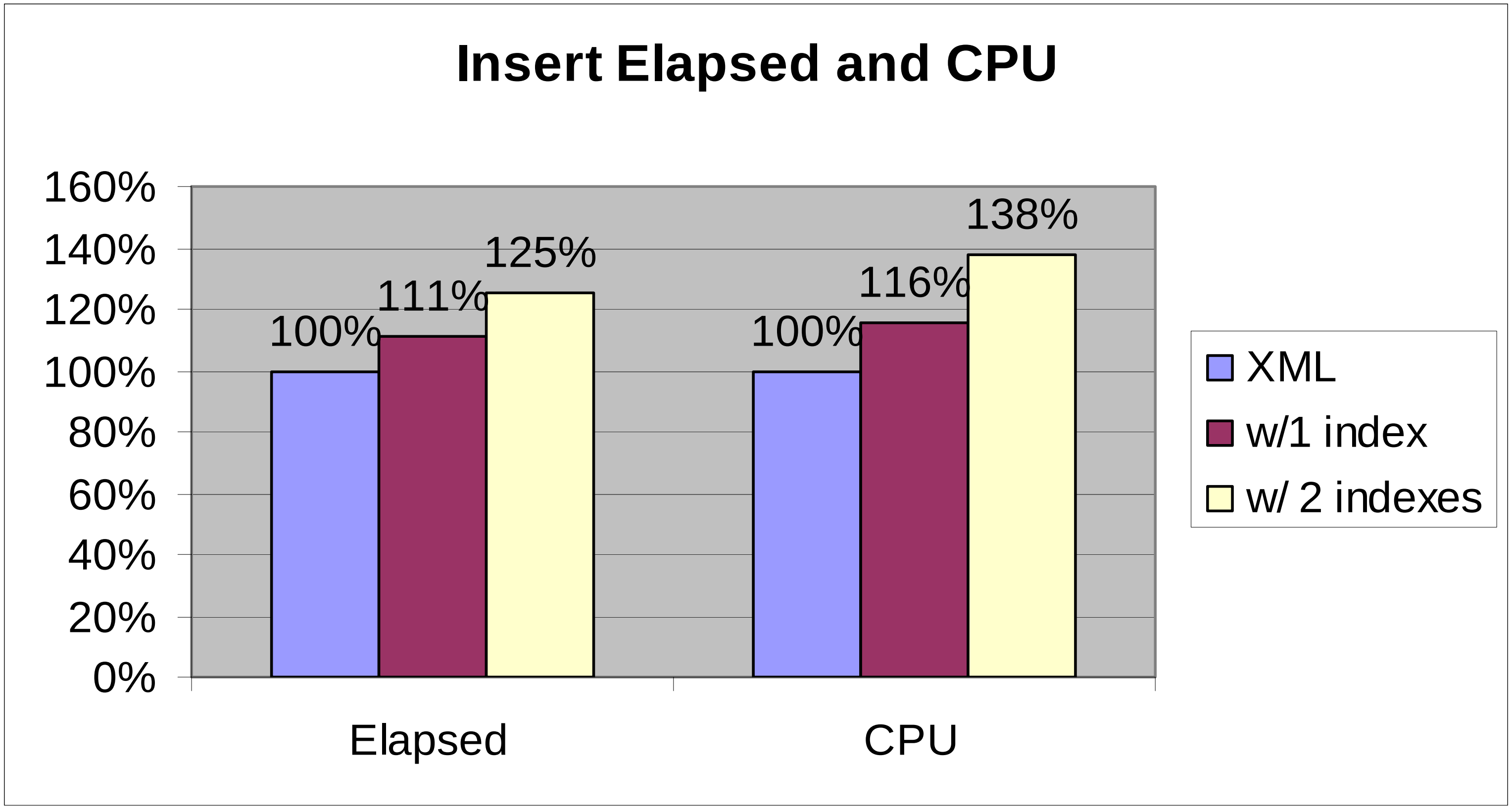
96 sample documents
Strip WS: Strip Whitespaces
Pres WS: Preserve Whitespaces

Insert Performance (Batch)

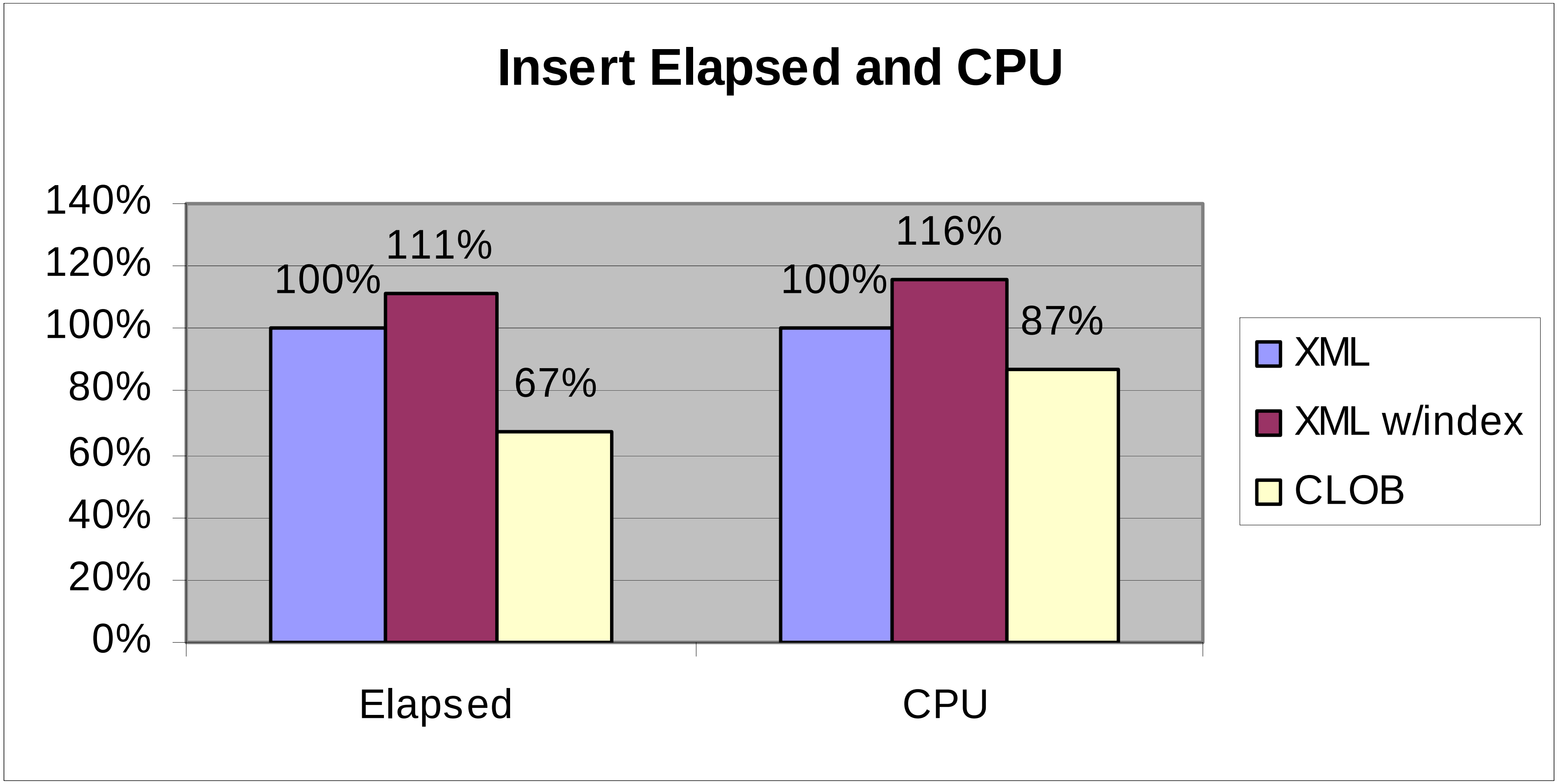


Measurement in March 2007, z9 DS8300, Single thread, Docs in EBCDIC

Insert XML – with indexes

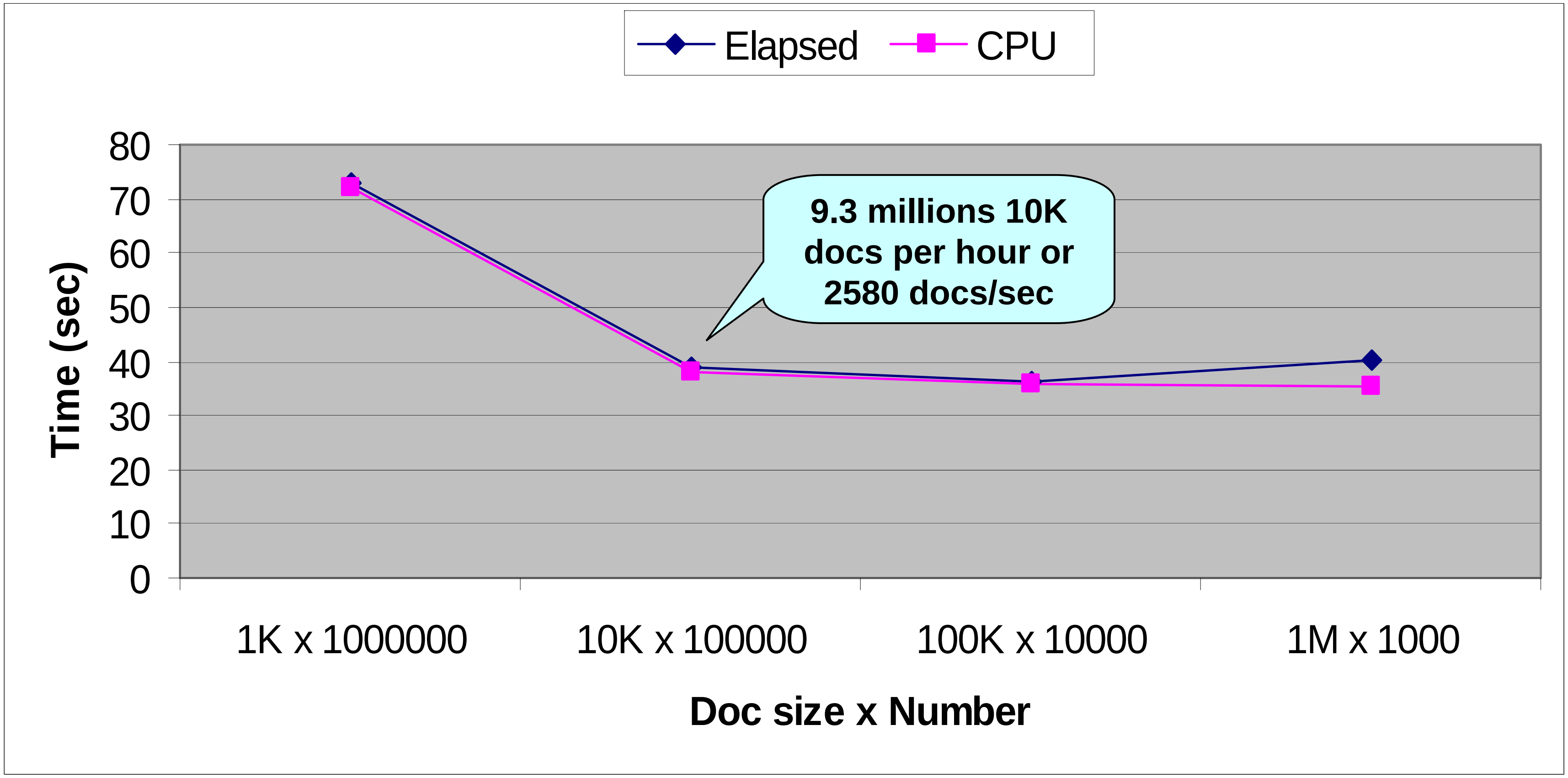


Insert Performance



(average of 1K to 10M document insert performance)

Fetch Performance (Batch)

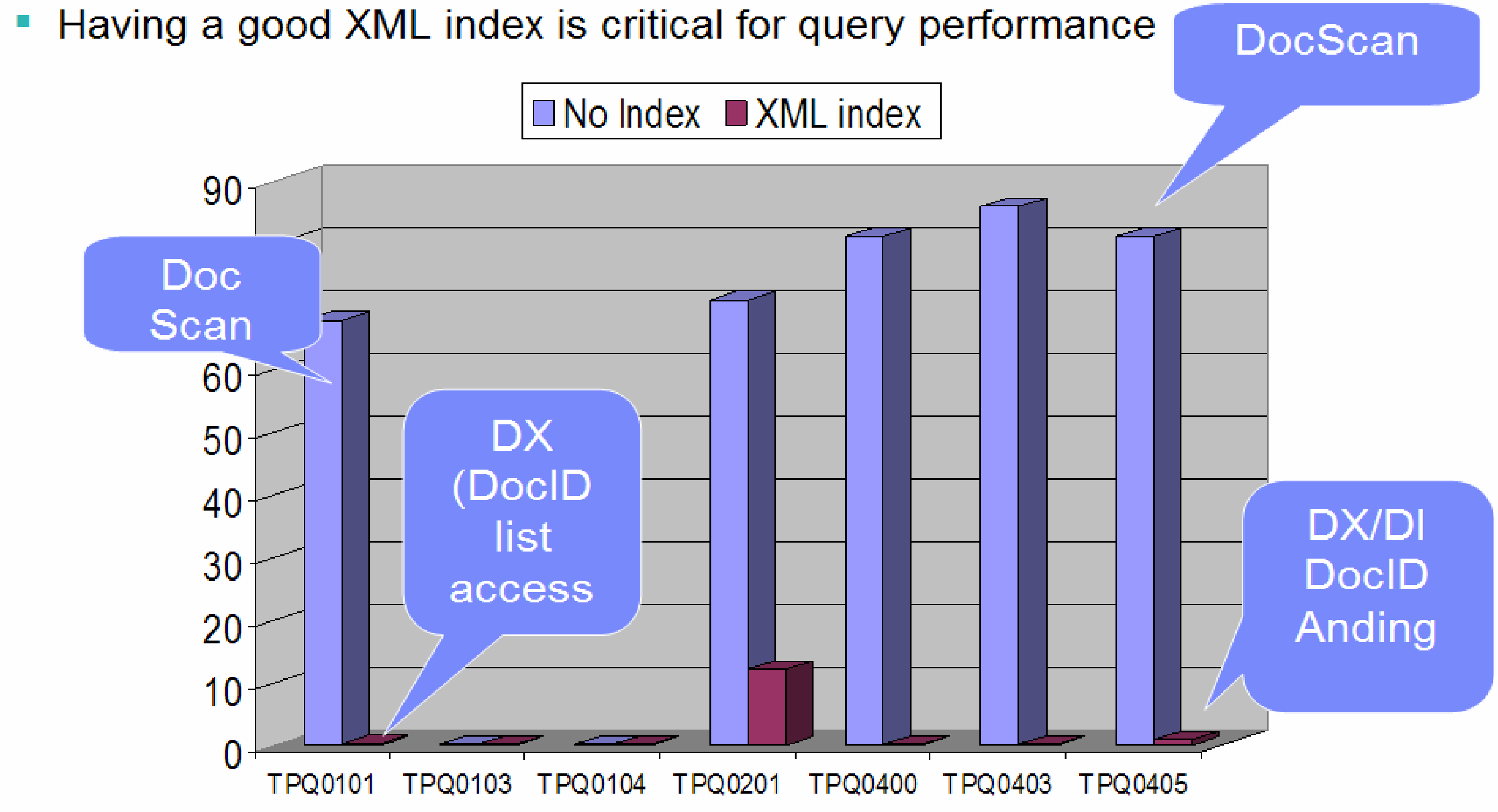


9.3 millions 10K docs per hour or 2580 docs/sec

Measurement in March 2007, z9 DS8300, Single thread, Docs in EBCDIC

XML Index Exploitation

- Having a good XML index is critical for query performance



Insert v.s. Validation v.s. Decomposition

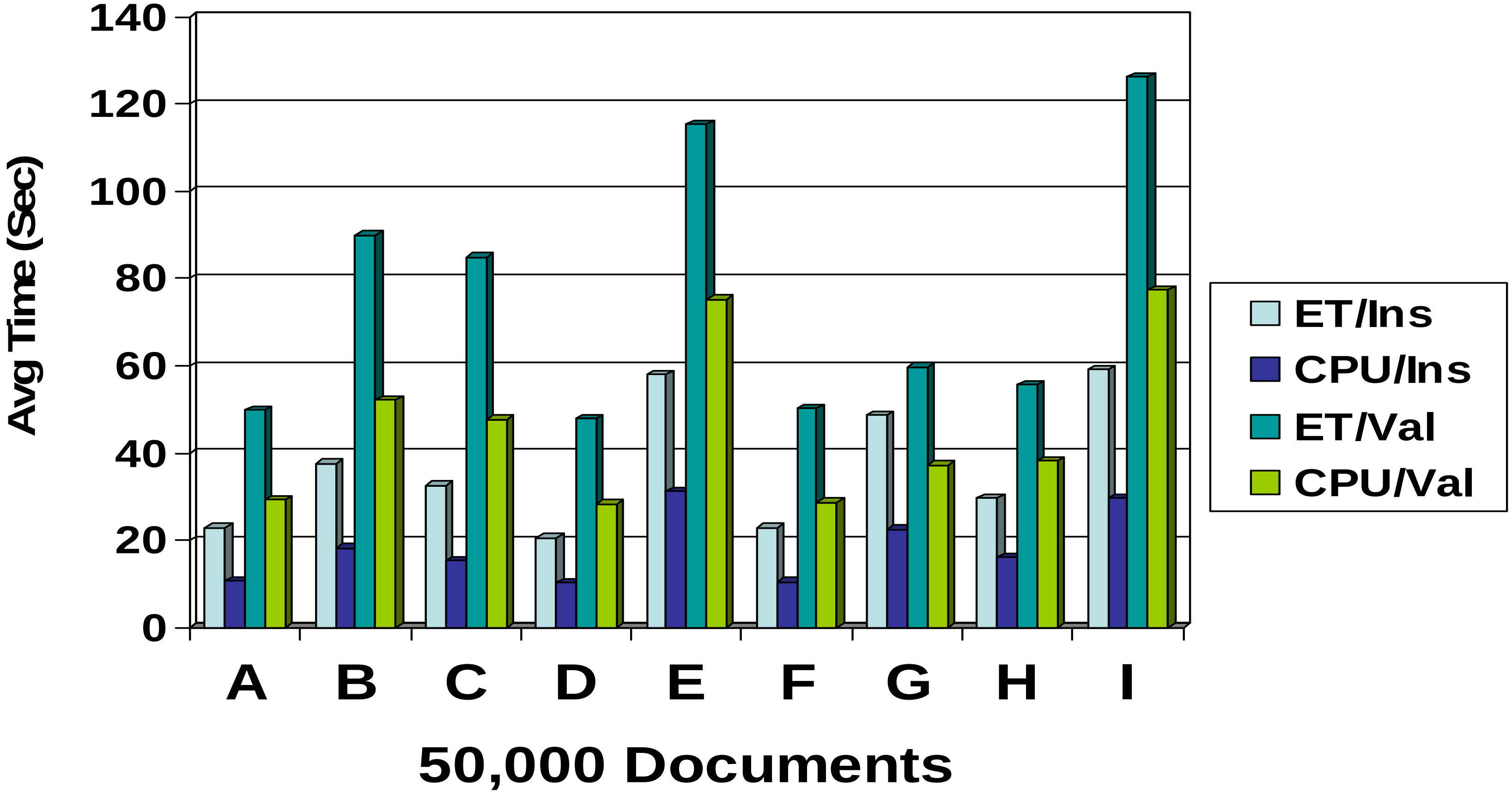
Testcases	Insert elapsed	Valid. elapsed	Insert CPU	Valid. CPU	Decomp elapsed	Decomp CPU
Custacc - 10,000 docs 4-19K	35.90s	43.51s	8.37s	13.98s	43.51s	13.98s
Qcapture – 4k doc x 3000 times	1.89s	3.37s	0.97s	2.21s	10.33s	5.6s
Shred23 – three 2k docs x 1000	0.55s	2.02s	0.26s	0.62s		

z9-109, 3 x 1.7GHz, 12GB CS, ESS M800

SEPA Sample XML Documents

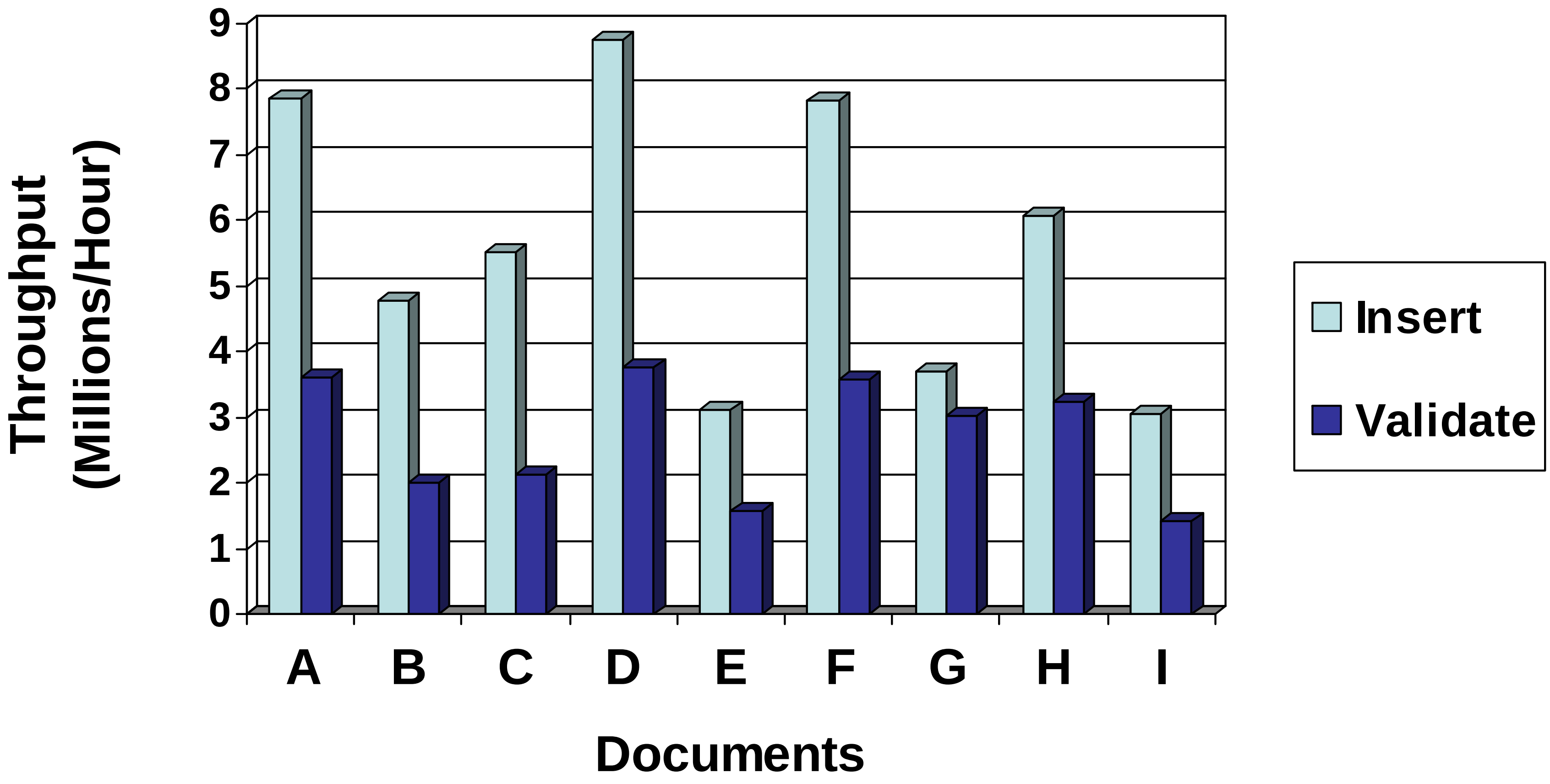
- A: Inter-bank direct debit collection (pacs.003.001.01)
- B: Return or Refund, interbank payment return (pacs.004.001.01)
- C: Reject - Payment status report (pacs.002.001.02)
- D: Interbank reversal (pacs.007.001.01)
- E: Customer Direct Debit Initiation (pain.008.001.01)
- F: Customer to bank payment reversal (pain.007.001.01)
- G: Bank to customer reject (pain.002.001.02)
- H: Customer Credit transfer initiation (pain.001.001.02)
- I: FI to FI Customer credit transfer (pacs.008.001.01)

SEPA/UNIFI XML Sample Insert Performance



ET: Elapsed Time, CPU: CPU Time, Ins: INSERT only, Val: w/ Validation

SEPA/UNIFI XML Sample Insert Throughput



Tools

- Tool choices:
  - Rational Data Architect
  - Rational Application Developer
  - Developer Workbench (DWB)
  - .NET
  - QMF
  - SPUFI
- Schema registration, validation
- Annotation for decomposition
- Mapping relational to XML schema for XML generation

XML Features in V9 - Summary

- First-class XML type, native storage of XQuery Data Model (XDM)
- Complete SQL/XML constructor functions
- XMLPARSE and XMLSERIALIZE
- XML indexes
- Other SQL/XML functions with XPath
  - XMLEXISTS, XMLQUERY, XMLTABLE
- XML Schema repository, Validation UDF, and decomposition
- DRDA (distributed support) and application interfaces
- Utilities and tools

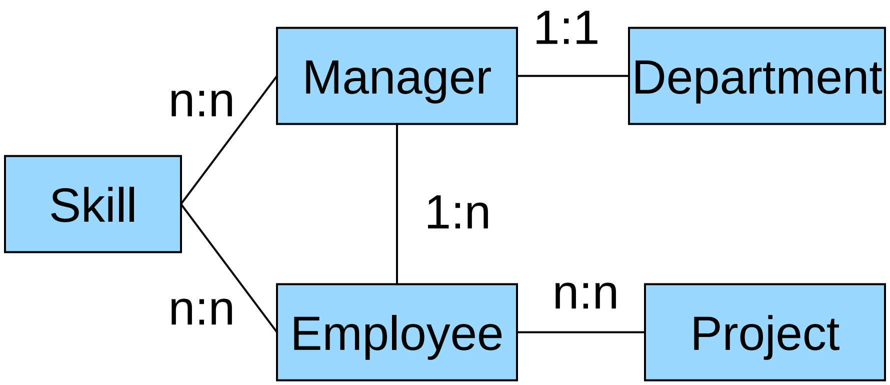
Some Restrictions and Limits

- Update: whole document replacement
- Largest document size: ~2GB
- Type annotation not kept in storage after validation.
- XML indexes:
  - Numeric and string types
  - String index key length: 1000 bytes
  - Keys do not span records
- XMLEXISTS is indexable, but stage 2, always re-evaluated by DocScan after index access.
- Range-partitioned table spaces: XML partitioning follows base table partitioning
- Triggers: XML columns cannot be transition vars
- Stored procedures: no XML type arguments

System Configurations

- Basic XML parsing requires z/OS XMLSS: z/OS 1.8 or z/OS 1.7 with APAR OA16303
- XML schemas requires IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V5 (5655-N98), SDK5.
- Zparms for storage: XMLVALA and XMLVALS
  - Default: 200MB and 10GB.
- Buffer pool for XML tables (default BP16K0), authorization.
  - DEFAULT BUFFER POOL FOR USER XML DATA ===> BP16K0 BP16K0 - BP16K9

XML Normalization Example



1. (D, M, S*, (E, P*, S*)*)*, S*, P* ✓
2. (M, S*, D, (E, S*, P*)*)*, P*, S* ✓
3. (P, (E, S*, M, S*, D)*)*, S* ✗
4. (S, (M, D, (E, P*)*)* ✗
5. …

Representing N:N Relationship

Employee

E1	N1	…
E2	N2	…
E3	N3	…

Project

P1	PN1	…
P2	PN2	…

P-E

P1	E1	…
P1	E2	…
P2	E2	
P2	E3	
P3	E3	…

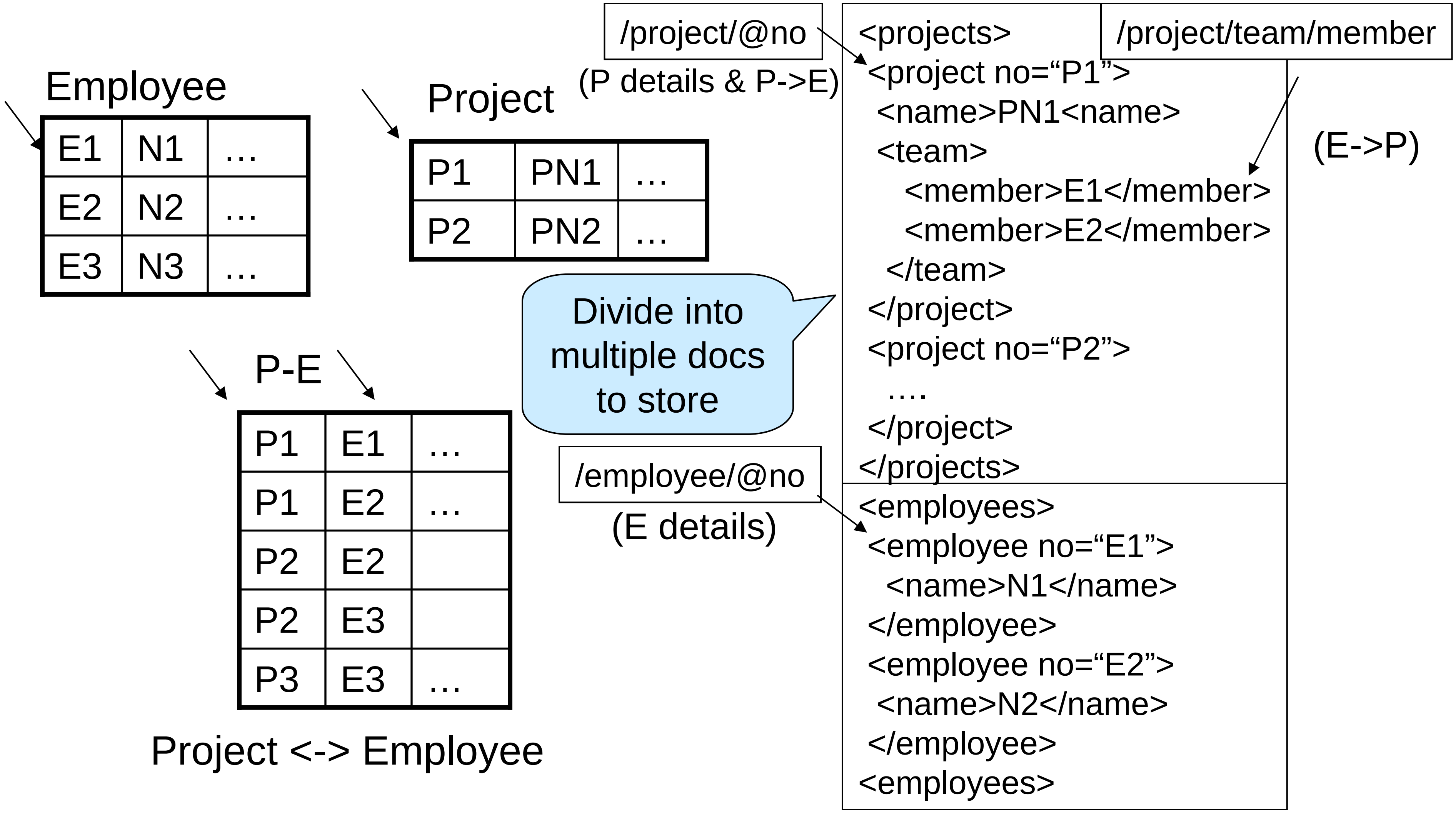
Project <-> Employee

Divide into multiple docs to store

```
<projects>
 <project no=“P1”>
  <name>PN1<name>
  <team>
     <member>E1</member>
     <member>E2</member>
   </team>
 </project>
 <project no=“P2”>
   ….
 </project>
</projects>
```

```
<employees>
 <employee no=“E1”>
   <name>N1</name>
 </employee>
 <employee no=“E2”>
  <name>N2</name>
 </employee>
<employees>
```

Indexing N:N Relationship



Employee

E1	N1	…
E2	N2	…
E3	N3	…

Project

P1	PN1	…
P2	PN2	…

P-E

P1	E1	…
P1	E2	…
P2	E2	
P2	E3	
P3	E3	…

Project <-> Employee

Divide into multiple docs to store

/project/@no

(P details & P->E)

/project/team/member

(E->P)

```
<projects>
 <project no="P1">
  <name>PN1<name>
  <team>
     <member>E1</member>
     <member>E2</member>
   </team>
 </project>
 <project no="P2">
   ….
 </project>
</projects>
```

/employee/@no

(E details)

```
<employees>
 <employee no="E1">
   <name>N1</name>
 </employee>
 <employee no="E2">
  <name>N2</name>
 </employee>
<employees>
```

XML Related Locks

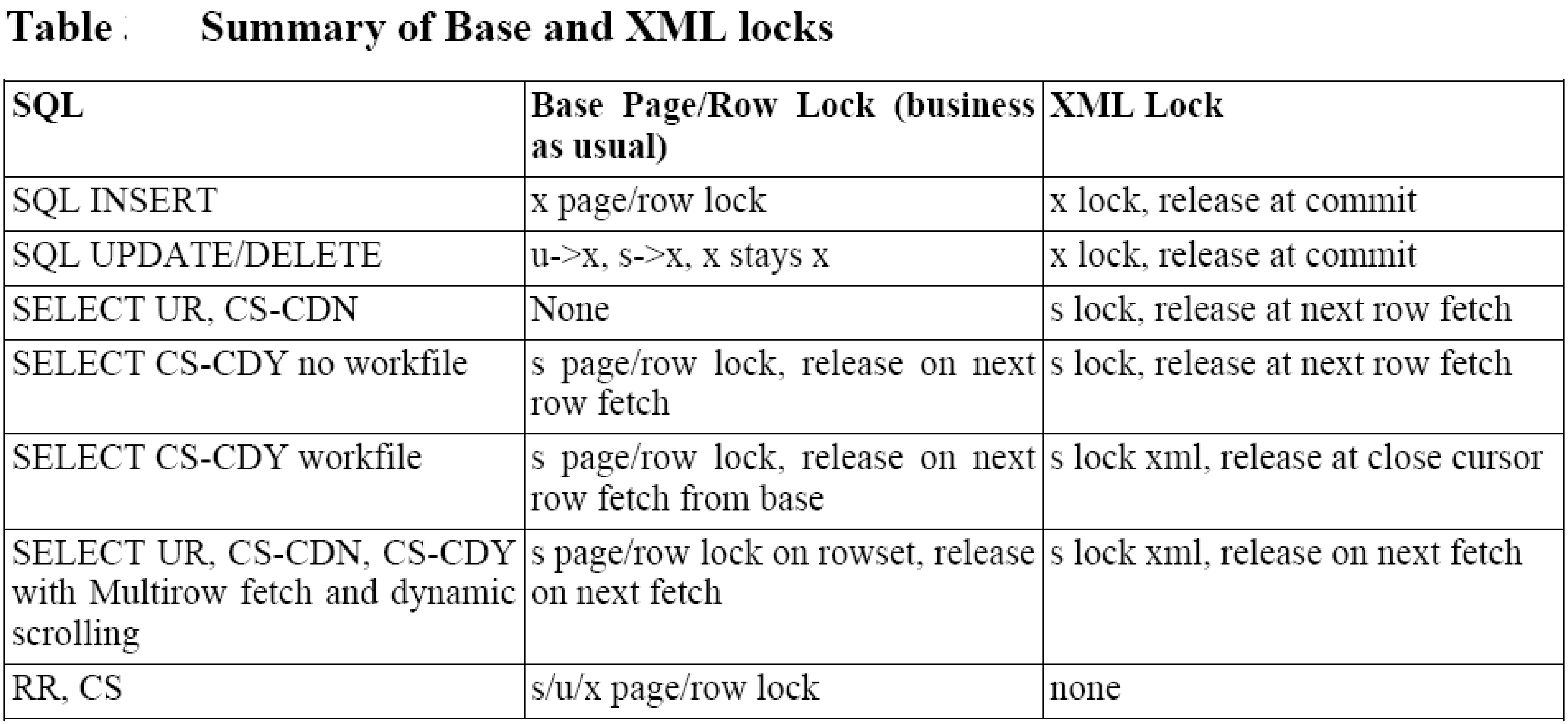


Table : Summary of Base and XML locks

SQL	Base Page/Row Lock (business as usual)	XML Lock
SQL INSERT	x page/row lock	x lock, release at commit
SQL UPDATE/DELETE	u->x, s->x, x stays x	x lock, release at commit
SELECT UR, CS-CDN	None	s lock, release at next row fetch
SELECT CS-CDY no workfile	s page/row lock, release on next row fetch	s lock, release at next row fetch
SELECT CS-CDY workfile	s page/row lock, release on next row fetch from base	s lock xml, release at close cursor
SELECT UR, CS-CDN, CS-CDY with Multirow fetch and dynamic scrolling	s page/row lock on rowset, release on next fetch	s lock xml, release on next fetch
RR, CS	s/u/x page/row lock	none

Summary

- Why XML and XML databases
- Usage scenarios
- DB2 pureXML features
- Performance
- Usage guide