InformationOnDemand2013

November 3 – 7
Mandalay Bay | Las Vegas, NV

#ibmiod

Exciting pureXML Enhancements in IBM DB2 11 for z/OS

Session Number IDZ-1860

Jane Man, IBM

Mengchu Cai, IBM

Rick Chang, IBM

Please note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

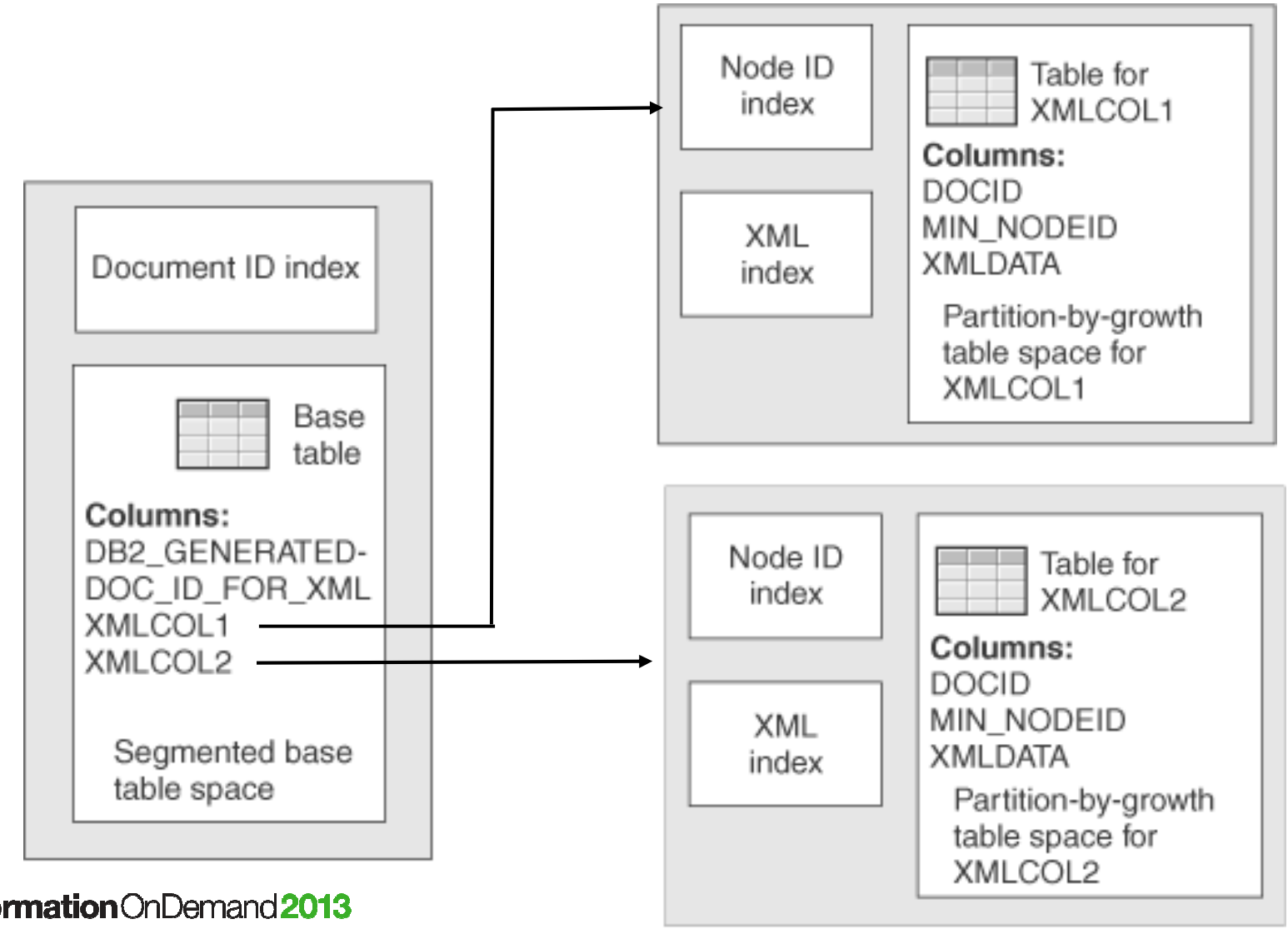
Agenda

1. Overview of pureXML enhancements
   - XML Storage
   - Querying XML
   - Validating XML
   - Updating XML

2. Customer Use Case

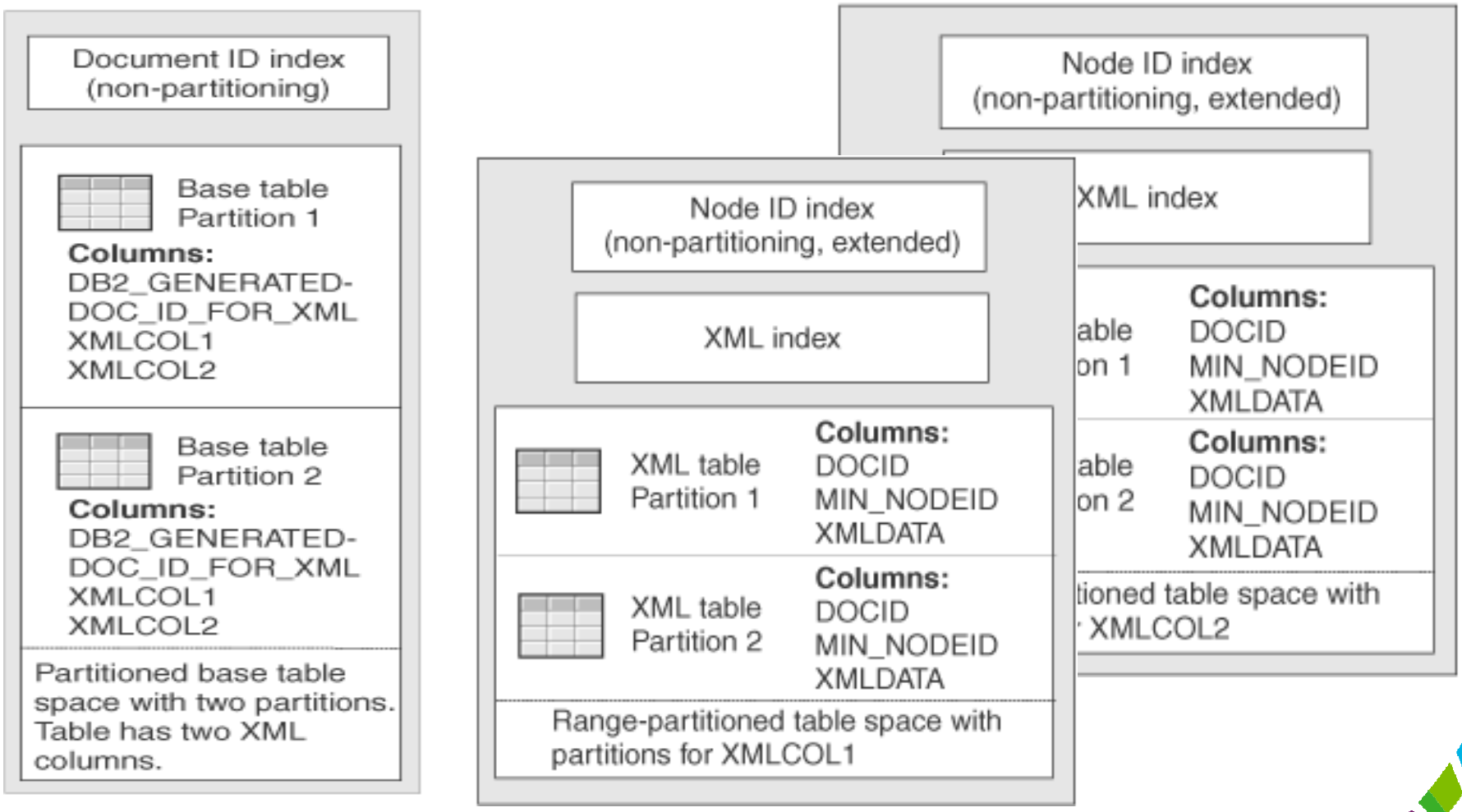
3. Summary

XML Storage for non-partitioned base table



XML storage for range-partitioned base table

1-to-1 correspondence between base table partitions and XML table partitions



XML Insert Scalability Relief from Page Latch Contention

- **Problem:** Since DB2 10 generates DOCID values in sequential order via an implicitly created sequence object and the XML DOCID and NODEID indexes are non-partitioned indexes, concurrent inserts create ***hotspots*** in these NPIs. As the number of threads increased, time spent waiting for page latch increased.
- **Solution:**
  - allow randomization of the DOCIDs, which would eliminate the hotspots in both indexes.
  - To enable it, set zparm XML_RANDOMIZE_DOCID to YES.
  - Only applies to new XML columns.
  - Field QWP1XRDI is added in IFCID 106 to trace the XML_RANDOMIZE_DOCID internal setting.
- Retrofit to DB2 10 via PM31486,PM31487

Randomize XML DOCID

```
DSNTIP8 INSTALL DB2 - PERFORMANCE AND OPTIMIZATION
===> _
Enter data below:
  1 CURRENT DEGREE          ===> 1     1 or ANY
  2 CACHE DYNAMIC SQL       ===> YES   NO or YES
  3 OPTIMIZATION HINTS      ===> NO    Enable optimization hints. NO
   or YES
  4 MAX DEGREE              ===> 0     Maximum degree of parallelism.
   0-254
  5 PARALLELISM EFFICIENCY===> 50        Efficiency of parallelism.
   0-100
  6 IMMEDIATE WRITE         ===> NO        NO, YES
  7 EVALUATE UNCOMMITTED    ===> NO        Evaluate uncommitted data.
   NO or YES
  8 SKIP UNCOMM INSERTS     ===> NO        Skip uncommitted inserts. NO
   or YES
 …
 15 RANDOMIZE XML DOCID     ===> NO    NO or YES
```


Querying XML

InformationOnDemand2013

November 3 – 7
Mandalay Bay | Las Vegas, NV

#ibmiod

Overview of Basic XQuery support

- XQuery features – retrofit to DB2 10
  - XPath expression – shipped in DB2 9
  - FLWOR expression
  - XQuery constructors
  - if-then-else expression
  - XQuery built-in functions and operators
  - XQuery prolog

SQL functions to invoke XQuery

- XMLQUERY
  - A scalar function that returns the result of an XQuery expression as an XML sequence.
  - Example: find the shipping address of each order

```
Select XMLQUERY('/order/shipTo' passing order)
From T1;
```

- XMLEXISTS
  - An SQL predicate that determines whether an XQuery expression returns a non-empty sequence.
  - Example: find those order that ship to IBM SVL.

```
Select order From T1
WHERE XMLEXISTS('/order/shipTo[street="555 Baily Ave" and
        city="San Jose" and state="CA"]' passing order);
```

- XMLTABLE
  - A table function that returns the result of an XQuery expression as a table.

XQuery constructors

- To construct an element with a known name and content, use XML syntax:

```
Select xmlquery(
  '<book isbn="0131873254">
     <title>Introduction to XQuery</title>
   </book>')
from sysibm.sysdummy1;
```

- If the content of an element or attribute must be computed, use a nested expression enclosed in **{ }**.

```
Select xmlquery(
  '<book isbn="{$b/@isbn}">
     <title>{$b/title/text()}</title>
   </book>' passing T1.book as "b")
from T1;
```


XQuery Prolog to control namespace behavior

- Syntax:

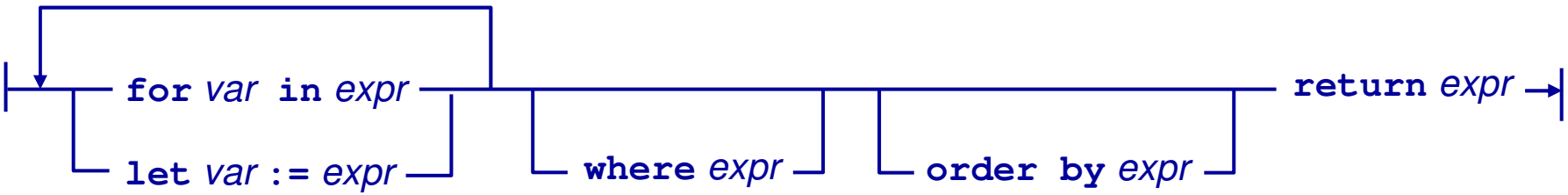
```
>>--declare--copy-namespaces--+--preserve-----+-,--inherit--;-->< 
                              '--no-preserve--'
```

- If copy-namespaces mode specifies preserve, all in-scope-namespaces of the original element are retained in the new copy. If copy-namespaces mode specifies no-preserve, the new copy retains only those in-scope namespaces of the original element that are used in the names of the element and its attributes.
- If copy-namespaces mode specifies inherit, the copied node inherits all the in-scope namespaces of the constructed node, augmented and overridden by the in-scope namespaces of the original element that were preserved by the preceding rule.
- Example

```
SELECT XMLQUERY('declare namespace bar="http://www.bar.com";
  declare copy-namespaces no-preserve,inherit;
  <bar:book>{$b//bar:title}</bar:book>'
  passing xmlparse('<s:Envelope xmlns:s="www.soap.org">
     <s:Body><bar:title xmlns:bar="www.bar.com">…</bar:title>
     </s:Body>
   </s:Envelope>') as "b")
FROM SYSIBM.SYSDUMMY1; <s:Envelope>
Namespace xmlns:s="www.soap.org" will be stripped from the new title element created inside <book>.
Return : <bar:book xmlns:bar="http://www.bar.com"/><bar:title>…
```


FLWOR expression

- A FLWOR expression iterates though a sequence, binds variables, applies a predicate, and constructs a new result.



FOR and LET clauses generate a list of tuples of bound variables.

WHERE clause applies a predicate, eliminating some of the tuples

ORDER BY clause imposes an order on the surviving tuples

RETURN clause is executed for each surviving tuple, generating an ordered list of outputs

An example query

- Find all items in a order that cost more than $100.

```
Select xmlquery(
  'for $i in $O/order/items/item
   let $p:=$i/price, $q:=$i/quantity,
       $amount:=$p * $q
   where $amount >100
   order by $amount
   return <item partno="{$i/partno}">
            {$amount}
          </item>'
passing T1.order as "O")
 from T1 …
```

```
<order orderDate="2011-05-18">
  <shipTo country="US">…</shipTo>
  <billTo country="US"> ..</billTo>
  <items>
    <item partno="872-AA"> …
      <quantity>2</quantity>
      <price>39.99</price> </item>
    <item partno="926-AA">…
      <quantity>2</quantity>
      <price>74.99</price></item>
    <item partno="945-ZG">
      <quantity>1</quantity>
      <price>178.99</price></item>
  </items>
</order>
```

Input

Output

```
<item partno="945-ZG">178.99</item>
<item partno="926-AA">149.98</item>
```


Conditional expression (if-then-else)

- Find out credit card type by its card number.

```
SELECT XMLQUERY(`
  let $y:=fn:replace($x, "[ \-]", "") return
  if (fn:matches($y, "^4[0-9]{12}([0-9]{3})?$"))
  then "Visa"
  else if (fn:matches($y, "^5[1-5][0-9]{14}$"))
     then "Master“
  else if (fn:matches($y,"^3[47][0-9]{13}$"))
     then "AmericanExpress”
  else if (fn:matches($y, "^6(011|5[0-9]{2})[0-
  9]{12}$"))
     then "Discover“
  else  "Unknown“’
PASSING '4123 456 789012' as "x")
FROM sysibm.sysdummy1
```

This returns “Visa".

Castable expression

- castable tests whether a given value is castable into a given target type.
- castable can be used to avoid errors at evaluation time.

```
avg(for $p in $o/order/items/item/price
    return if ($price castable as xs:decimal)
           then xs:decimal($price) else () )
```

- castable can also be used for dynamic typing.
  - Create function add(x XML, y XML) … return XMLQUERY(

```
'if ($x castable as xs:double) then $x+$y
 else if ($x castable as xs:date) then
     xs:date($x)+ xs:yearMonthDuration($y)
 else fn:concat($x, $y)' passing X as "x", Y as "y");
```

  - add(XMLPARSE('<val>2012-01-02</val>'),
    XMLPARSE('<val>P3M</val>')) returns 2012-04-02

XQuery performance improvements in Sequoia

- FLWOR improvements
  - Simple where predicate push down.

```
for $i in /order/items/item
where $i/price > 100
return $i/desc
```

Push the where predicate into xpath as below:

```
for $i in /order/items/item[price > 100]
return $i/desc
```

  - Simple FLWOR to xpath

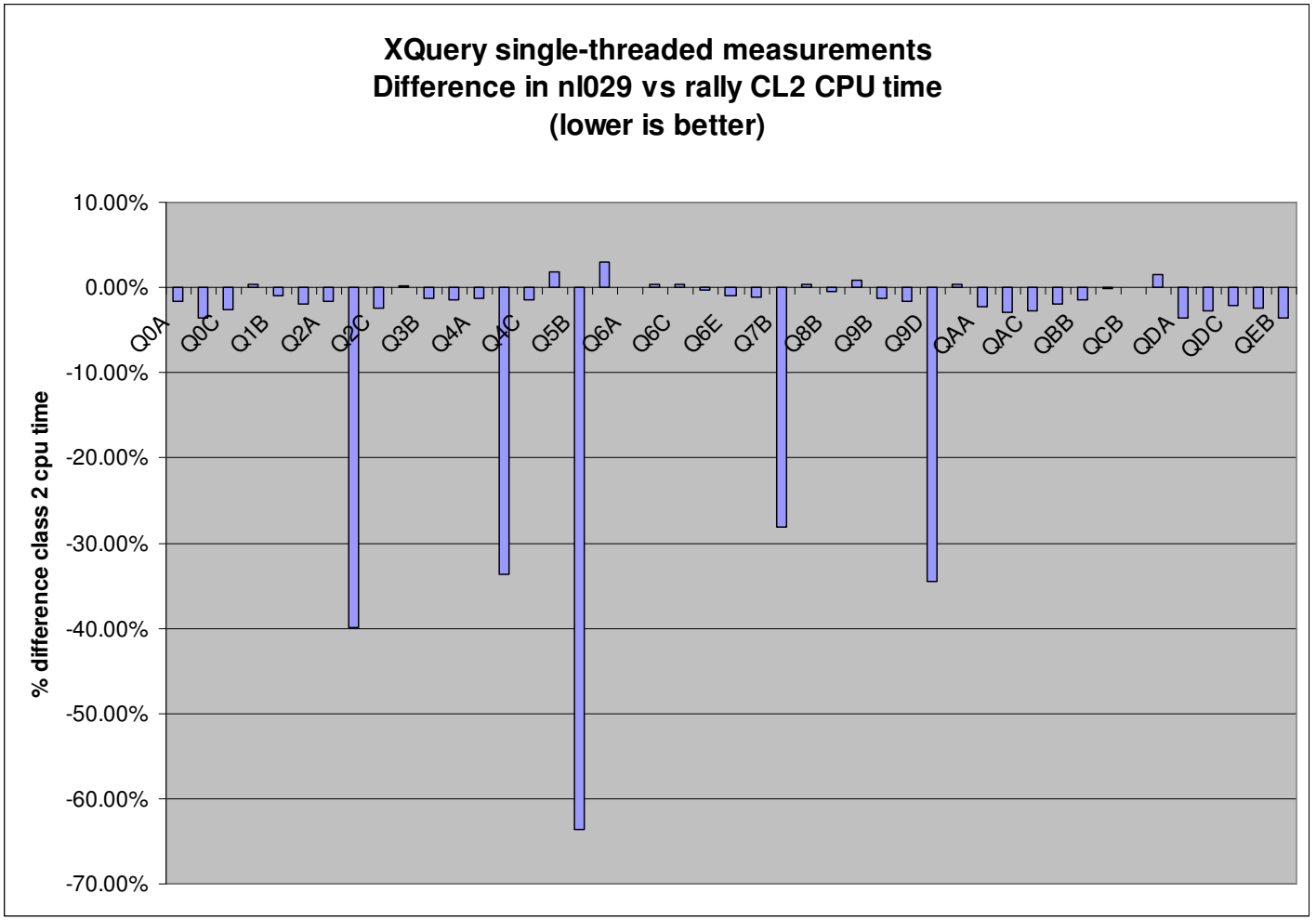
```
for $i in /order/shipTo return $i
```

rewrite into xpath as below:

```
/order/shipTo
```

  - Up to 60% CPU reduction for qualified queries.

Performance for XMLQuery with FLWOR



More XQuery performance improvements

- Defer nested XQuery constructor (lazy evaluation)
  - Defer the construction of <c> node.

```
<Output>
   { for $i in $x/a/b
     return <c>{$i/d}</c>}
</Output>
```

  - Up to 20% CPU reduction, up to 60% storage reduction
- XQuery built-in function and operator improvements
  - For one operator, reduce storage usage by 64KB, 0~3% CPU reduction.
  - Internal test for 300 '+' operators showed 39% CPU reduction, 98% storage reduction.

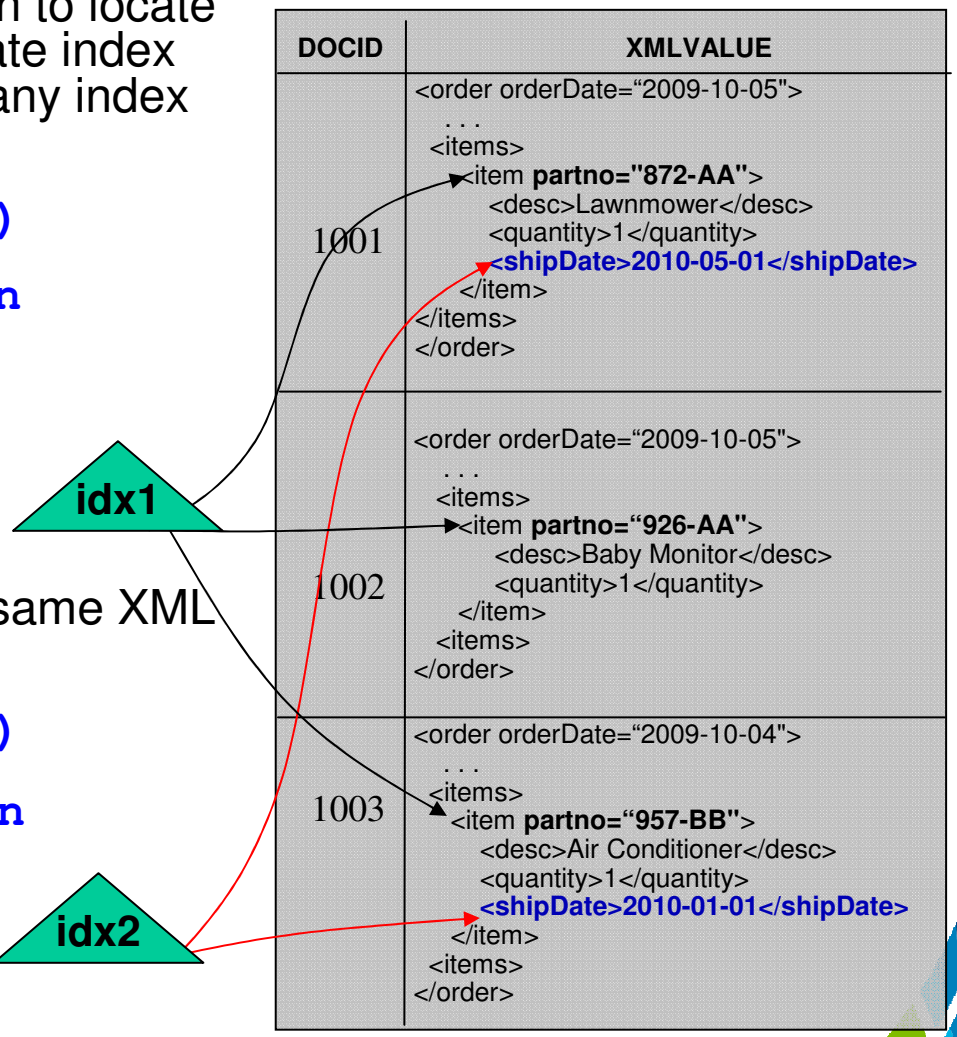
Indexing XML data

- An XML index uses an XPath expression to locate a node in an XML document and generate index key. Each document can have 0,1 or many index keys.

```
CREATE INDEX idx1 ON T1(order)
  GENERATE KEY USING XMLPattern
  '/order/items/item/@partno'
  AS SQL VARCHAR(8);
```

- Multiple indexes can be created on the same XML column.

```
CREATE INDEX idx2 on T1(order)
  GENERATE KEY USING XMLPattern
  '/order/items/item/shipDate'
  AS SQL DATE;
```



XML index matching

- The following query can use idx1.

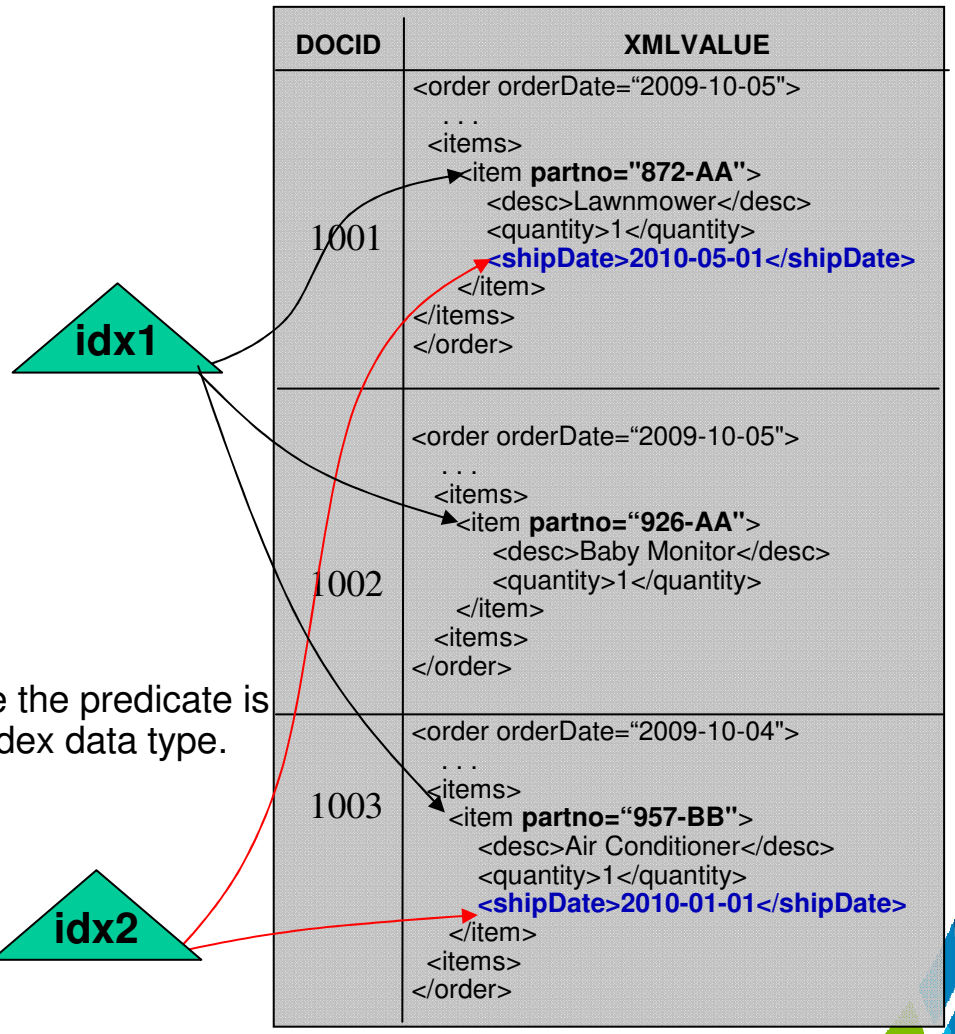
```
SELECT * FROM T1
WHERE XMLEXISTS('/order/items/item
        [@partno="872-AA"]'
       PASSING T1.order);
```

- The following query can use idx2.

```
SELECT * FROM T1
WHERE XMLEXISTS('/order/items/item
   [shipDate=xs:date("2010-05-01")]'
   PASSING T1.order);
```

- But the following query can not use idx2 because the predicate is a string comparison which does not match the index data type.

```
SELECT * FROM T1
WHERE XMLEXISTS('/order/items/item
      [shipDate="2010-05-01"]'
       PASSING T1.order);
```

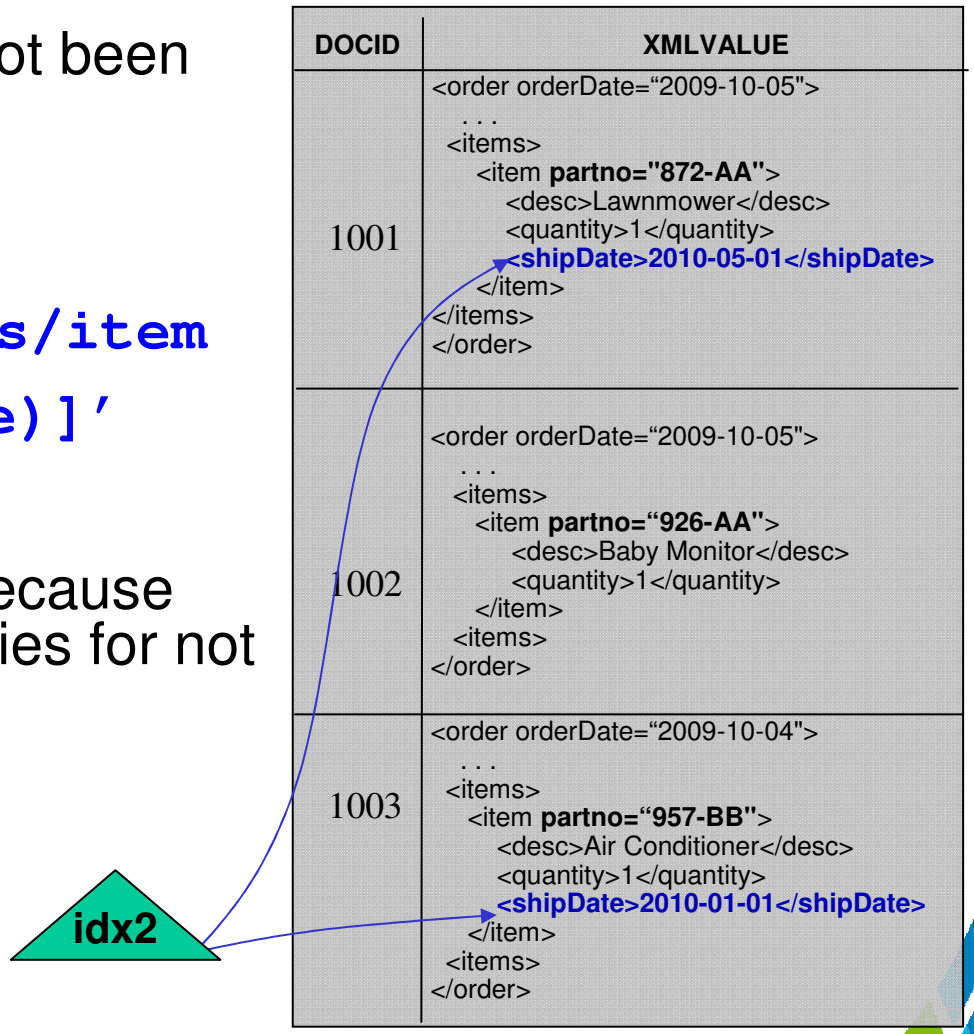


Non-existential predicates

- Find all the orders that have not been shipped yet.

```
SELECT * FROM T1
 WHERE
XMLEXISTS(`/order/items/item
        [fn:not(shipDate)]'
   PASSING T1.order);
```

- This query can not use idx2 because idx2 does not contain key entries for not shipped orders (1002).



XML index for non-existential predicates

- Create an index using fn:exists to indicate the existence of a node.

```
CREATE INDEX idx3 on T1(order)
GENERATE KEY USING XMLPattern
'/order/items/item/fn:exists(shipDate)'
AS SQL VARCHAR(1);
```

The follow queries can use idx3.

```
SELECT * FROM T1
WHERE XMLEXISTS('/order/items/item
      [fn:not(shipDate)]'
  PASSING T1.order);

SELECT * FROM T1
WHERE XMLEXISTS('/order/items/item
      [fn:exists(shipDate)]'
  PASSING T1.order);
```

T

F

T

DOCID	XMLVALUE
1001	<order orderDate="2009-10-05"> . . . <items> <item **partno="872-AA"**> <desc>Lawnmower</desc> <quantity>1</quantity> **<shipDate>2010-05-01</shipDate>** </item> </items> </order>
1002	<order orderDate="2009-10-05"> . . . <items> <item **partno="926-AA"**> <desc>Baby Monitor</desc> <quantity>1</quantity> </item> <items> </order>
1003	<order orderDate="2009-10-04"> . . . <items> <item **partno="957-BB"**> <desc>Air Conditioner</desc> <quantity>1</quantity> **<shipDate>2010-01-01</shipDate>** </item> <items> </order>

XML indexes for case-insensitive search

- Use **fn:upper-case()** in predicate for case-insensitive search.

```
SELECT * FROM T1
WHERE XMLEXISTS(`/order/items/item
        [fn:upper-case(desc)="BABY MONITOR"]'
        PASSING T1.order);
```

  - This query can use idx4 created below.

- Use **fn:upper-case()** to create case-insensitive XML index.

```
CREATE INDEX idx4 ON T1(order)
  GENERATE KEY USING XMLPATTERN
  `/order/items/item/fn:upper-case(desc)'
  AS SQL VARCHAR(128);
```


XMLTABLE

Row generating expression

```
SELECT X.* FROM T1,
  XMLTABLE('//item' passing T1.order
   COLUMNS
     partNo   CHAR(8)       path '@partno',
     quantity INTEGER       path 'quantity',
     price    DECIMAL(6,2)  path 'price',
     shipDate Date          path 'shipDate') X
```

column definition

```
<order orderDate="2011-05-18">
  <shipTo country="US">…</shipTo>
  <billTo country="US"> ..</billTo>
  <items><item partno="872-AA">
     <shipDate>2011-05-23</shipDate>
     <quantity>2</quantity>
     <price>39.99</price></item>
   <item partno="926-AA">…
     <shipDate>2011-05-24</shipDate>
     <quantity>2</quantity>
     <price>74.99</price></item>
   <item partno="945-ZG">
     <quantity>1</quantity>
     <price>178.99</price></item>
  </items>
</order>
```

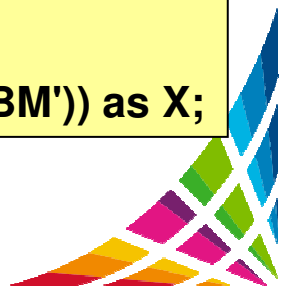
partNo	quantity	price	shipDate
872-AA	2	39.99	2011-05-23
926-AA	2	74.99	2011-05-24
945-ZG	1	178.99	(null)

Encapsulate web service in a parameterized view with XMLTable

```
CREATE FUNCTION STOCKQUOTE(SYMBOL VARCHAR(4))
 RETURNS TABLE (Symbol VARCHAR(4), LAST DECIMAL(6,2), …)
 RETURN SELECT * FROM
  XMLTABLE('//GetQuoteResult'  PASSING XMLPARSE(
        DB2XML.SOAPHTTPNC(…,
   '<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body><GetQuote xmlns="http://ws.cdyne.com/">
         <StockSymbol>' || SYMBOL || '</StockSymbol>
         <LicenseKey>12345</LicenseKey>
      </GetQuote></soap:Body></soap:Envelope>') )
    COLUMNS   “StockSymbol"          VARCHAR(4) ,
       "LastTradeAmount"      DECIMAL(6,2),
       “LastTradeDateTime"  TIMESTAMP(0),
       “Change"                   Decimal(8,2),
       “OpenAmount"           Decimal(8,2),
       “High"            Decimal(8,2),
       “Low"             Decimal(8,2),
       "Volume"       INT,
      “PrevCls”       Decimal(8,2)) XT
```

Invocation:

```
SELECT   X.*
FROM TABLE( STOCKQUOTE('IBM')) as X;
```

XMLTABLE Performance Improvements (1)

- DB2 10 post-GA improvements
  - Remove unreferenced column definitions.
    - 47% CPU reduction on XMLTABLE with 30 columns
  - Merge common column path expression
  - Up to 74% reduction on XPath storage consumption
  - Storage reuse for output XML columns, resolve -904 issue. (PM69176, Q4/2012)
    - **INSERT INTO SEPA_CdtTrf(doc)**

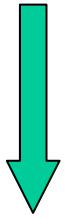
      SELECT X.doc

      FROM SEPA, XMLTABLE (`//CdtTrfTxInf' passing doc);

XMLTABLE Performance Improvements (2)

- DB2 Sequoia improvements
  - Date and timestamp predicates pushdown.

```
SELECT X.* FROM T1,
   XMLTABLE(`//item' passing T1.order
     COLUMN partNO varchar(8) path `@partNo',
         shipDate Date  path `shipDate') X
Where x.shipDate = `2011-05-23';
```



Push SQL predicates into XPath, enable XML index access.

```
SELECT * FROM T1,
  XMLTABLE(`//item[shipDate=$d]' passing T1.order,
                  cast(`2011-05-23' as Date) as "d"
  COLUMN partNO varchar(8) path `@partNo',
         shipDate Date  path `shipDate') X;
```


XMLTABLE Performance Improvements(3)

- Optimize index key range for varchar predicates.

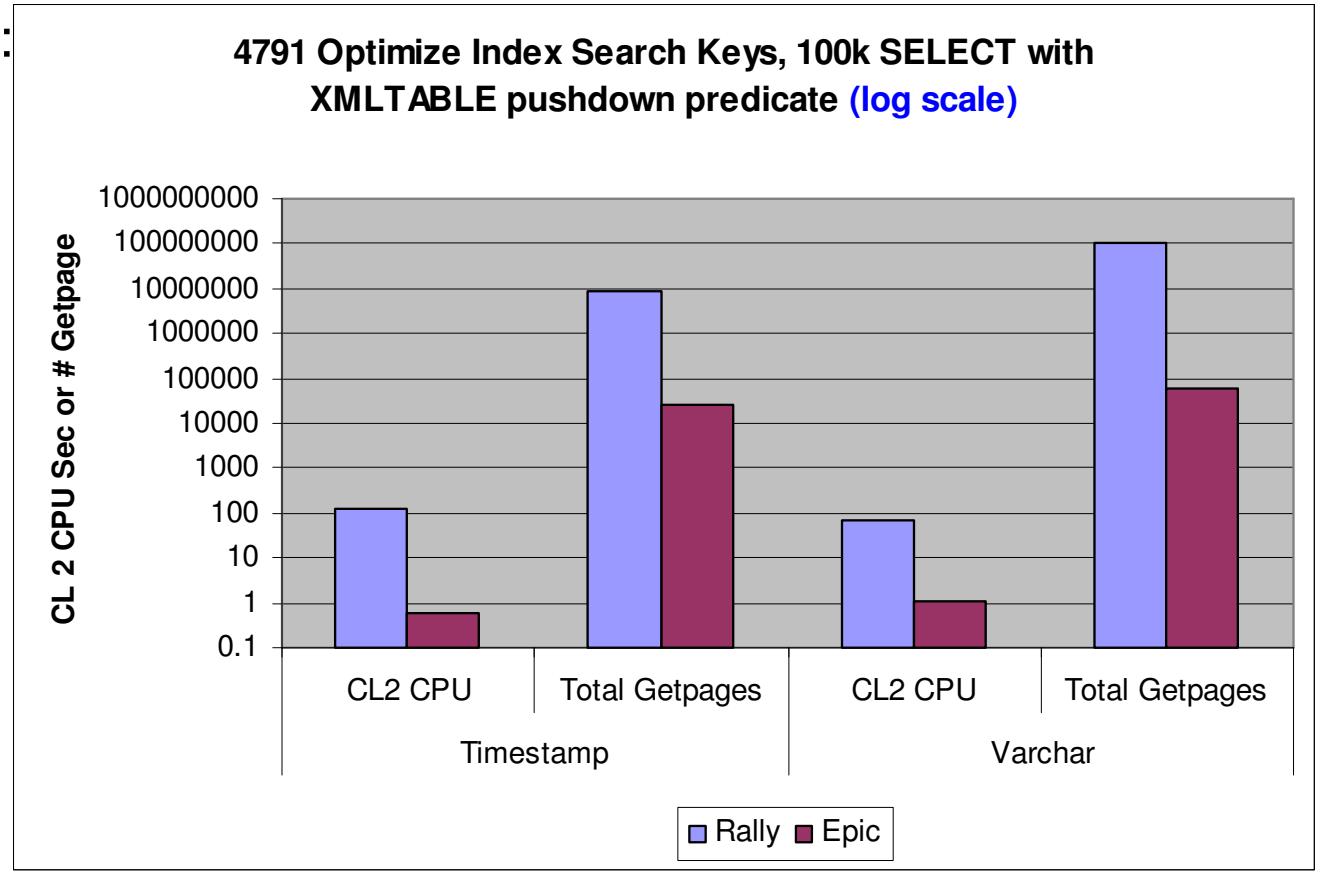
```
SELECT X.* FROM T1,
   XMLTABLE(`//item' passing T1.order
     COLUMN partNO varchar(6) path `@partNo',
         shipDate Date  path `shipDate') X
Where x.partNo = `872-AA';
```

Extra upper bound predicate is added to reduce index key range.

```
SELECT * FROM T1,
  XMLTABLE(`//item[left(@partNo,6)=$d]'
   passing T1.order, `872-AA' as "d"
  COLUMN partNO char(8) path `@partNo',
         shipDate Date  path `shipDate') X
where XMLEXISTS (`//item[@partNo>=$d and
                         @partNo <= $u]' passing
 T1.order, `872-AA' as "d", '872-AA'||X'FF' as "u");
```


Performance of optimizing index key range

- Test Description: single-threaded 100k SELECT with XMLTABLE and pushdown '=' predicate on varchar column.
- Results: Order of magnitudes improvement in CPU time and Getpage counts for affected queries



Validating XML

InformationOnDemand2013

November 3 – 7
Mandalay Bay | Las Vegas, NV

#ibmiod

XML Schema

- XML Schema is a language for *expressing constraints about XML documents.*
  - Defines elements, attributes, types of elements,…
  - An XML schema can consist of multiple schema files.
  - Checking an XML document against a schema is known as validation.
- XML Schema Repository (XSR)
  - Stores registered XML schemas (XSR objects).
  - an *XML schema must be registered in XSR before it is used for validation*.

```
<xsd:schema
targetNamespace="http://tpox.com/custac"
 ....
<xsd:element name="Customer"
  type="custacc:CustomerType"/>
<xsd:complexType name="CustomerType">
 <xsd:sequence>
   <xsd:element name="Name" type="xsd:string"/>
   <xsd:element name="DateOfBirth"
    type="xsd:date"/>
   <xsd:element name="Address" type="xsd:string"
    maxOccurs="unbounded"/>
   <xsd:element name="Phone" type="xsd:string"
       minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType> ....
```

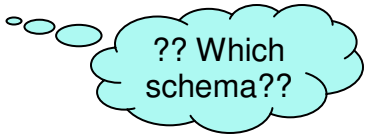
```
<Customer xmlns="http://tpox.com/custac"
xsi:schemaLocation=" http://tpox.com/custac
http://tpox.com/custac/customer1.xsd"...>
   <Name>John Smith</Name>
   <DateOfBirth>1980-04-01</DateOfBirth>
   <Address>555 Bailey Ave. San Jose, CA,
95141
   </Address>
   <Phone>408-444-5555</Phone>
   <Phone>408-222-3333</Phone>
 </Customer>
```

XSRobjID	Schema Name	Target Namespace	Schema Location	Registration Timestamp
1	CUST1	http://tpox.com/custac	http://tpox.com/custac/customer1.xsd	2009-01-01 10:00:00.0000
2	PO2	http://www.example.com/PurchaseOrder_2	http://www.example.com/PurchaseOrder_2_0.xsd	2010-01-01 10:00:00.0000

Validate XML

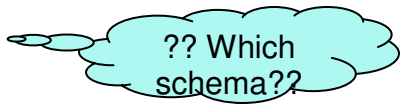
Explicit validation

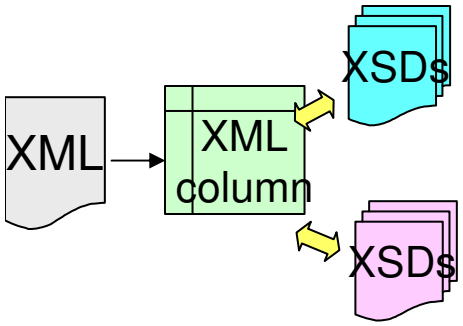
- Create table T1 (C1 XML);
- Insert into T1 values (dsn_xmlvalidate(:hv,'SYSXSR.PO1'));
- Update T1 set C1 = dsn_xmlvalidate(:hv,'SYSXSR.PO1');
- Insert into T1 values (dsn_xmlvalidate(:hv));



Implicit validation

- Create table T1 (C1 XML
  (XMLSCHEMA ID SYSXSR.PO1,
  ID SYSXSR.PO2));
- INSERT, UPDATE, MERGE, LOAD and CHECK DATA utilities will perform validation
- Insert into T1 values (:hv);





Validation is provided by z/OS XML service
100% zIIP redirect-able

Dynamic schema resolution

- Dynamic schema resolution for both implicit and explicit validation

DB2 dynamically determines an XML schema based on namespace, schema location and registration timestamp.

XML schema name	Target Namespace	Schema Location	Registration Timestamp
PO1	http://www.example.com/PO1	http://www.example.com/PO1.xsd	2009-01-01 10:00:00.0000
PO2	http://www.example.com/PO2	http://www.example.com/PO2.xsd	2010-01-01 10:00:00.0000
PO3	NO NAMESPACE	http://www.example.com/PO3.xsd	2010-01-30 10:00:00.0000
PO4	http://www.example.com/PO2	http://www.example.com/PO4.xsd	2010-02-23 08:00:00.000

XML Instance document:

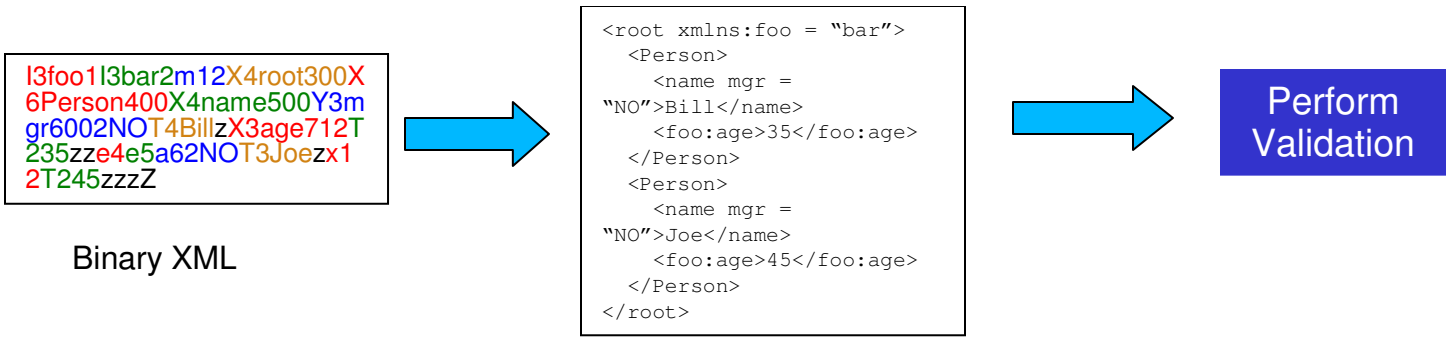
```
INSERT INTO purchase_orders VALUES(2,
'<po:purchaseOrder xmlns:po="http://www.example.com/PO2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.com/PO2
http://www.example.com/PO2.xsd">
...
</po:purchaseOrder>');
```

***Best practices* for schema evolution**

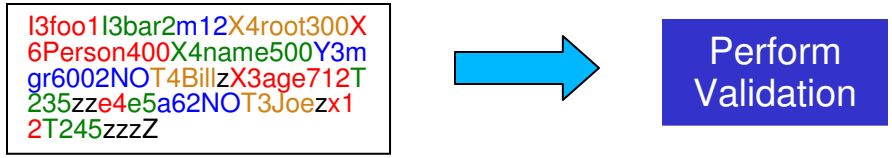
1) specify schema location in each instance document.
2) If not, then make sure that new versions with same target namespace are backward compatible.
3) Otherwise use different namespace.

Validate Binary XML

- **Problem:** To validate binary XML, DB2 10 need to serialize the binary XML into string XML, which defeats the purpose of using binary XML.



- **Solution in DB2 Sequoia**: Validate binary XML directly (both insert and load)
  - Dependency: z/OS R13 PTF UA63422 or R12 PTF UA65591



Performance improvement

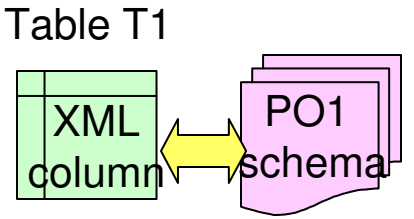
INSERT into XML column with type modifier using binary XML, *30~40% CPU reduction, 15~18% CPU* reduction vs string XML.

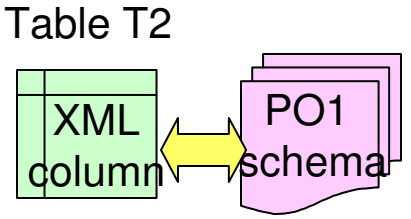
LOAD into XML column with type modifier using binary XML, *41% CPU reduction vs DB2 10, 18% improvement* vs string XML

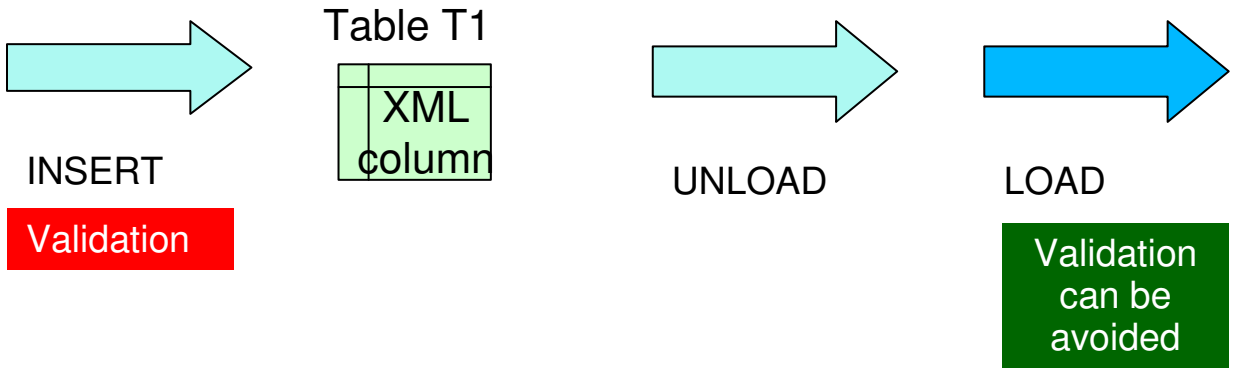
Avoid Revalidation during LOAD

- **Example**

1. CREATE TABLE T1(ID INT,
   XMLCOL XML(XMLSCHEMA ID SYSXSR.PO1));
2. CREATE TABLE T2(ID INT,
   XMLCOL XML(XMLSCHEMA ID SYSXSR.PO1));
3. Populate table T1 – validation
4. UNLOAD from T1 using binary XML. PO1
5. LOAD to T2 using binary XML.







Performance Results

NL029 Avoid Revalidation with Load

Elapsed Time

Class1			
Job	Base	NL029	% delta
SEPA	487.16	343.33	-29.5242
Custacc	122.75	81.14	-33.8982
Medline	109.11	61.4	-43.7265
IRS25MB	80.6	52.95	-34.3052
IRS250MB	160.09	93.17	-41.8015
IRS766MB	156.94	94.28	-39.9261

CPU Time

Class1			
Job	Base	NL029	% delta
SEPA	217.24	66.66	-69.315
Custacc	49.77	19.18	-61.4627
Medline	40.18	12.5	-68.89
IRS25MB	38.01	9.27	-75.6117
IRS250MB	76.72	18.33	-76.1079
IRS766MB	78.11	18.32	-76.5459

Sepa - 10k, 1MB, 10MB, 25MB, 50MB, and 100MB docs loaded into 1 table
Custacc - 200K Rows of 4-10k xml docs
Medline - 200 10MB docs
IRS25MB 25MB doc 75 rows
IRS250MB 250MB doc 15 rows
IRS766MB 766mb doc 5 rows

Updating XML

InformationOnDemand2013

November 3 – 7
Mandalay Bay | Las Vegas, NV

#ibmiod

XMLModify – sub document update

- DB2 10 supports sub document modification.
  - Using XQuery basic updating expressions
    - **Insert** expression
    - **Replace** expression
    - **Delete** expression
  - Basic updating expressions can only be used in the **XMLMODIFY** function
  - *Only changed records in the XML table are updated*. Other parts untouched.
  - Concurrency control by the base table (document level)

Examples

Insert Example

```
UPDATE T1 SET order = XMLMODIFY
    ('insert node $s as last into //item[1]',
      XMLELEMENT(name "shipDate",'2011-05-30') as "s");
```

The keywords node and nodes may be used inerchageably

Replace Example

```
UPDATE T1 SET order = XMLMODIFY
   ('replace value of node //item[1]/price
       with //item[1]/price * 0.9') where …;
```

Delete Example

```
UPDATE T1 SET order = XMLMODIFY
    ('delete node /order/items/item[@partNo="872-AA"]')
  WHERE …;
```


XMLModify Enhancements (Sequoia)

- Support *XQuery constructors* as a source expression.

  Example:

```
UPDATE T1
 SET order = XMLMODIFY(
  `insert nodes
    <item partNum="999-ZZ">
      <desc>Sapphire Bracelet</desc>
      <quantity>1</quantity>
      <price>199.99</price>
      <shipDate>2012-06-03</shipDate>
    </item>
 as last into /order/items');
```


Partial Revalidation

- When an XML document is modified, instead of re-validating the whole document, *DB2 only re-validate the changed part*.

```
CREATE TABLE T1(order XML(XMLSCHEMA ID
                          SYSXSR.PO1));
Insert into T1(order) values(…);
Update T1 set order = XMLModify(
'replace value of node //item[1]/shipDate
with "2012-05-25"');
```

<?xml version=
.....
.....
.....
<iitem><shipDate>
2012-05-25
</shipDate>.....
.....
.....
.....
.....
.....
.....
.....
.

- only <shipDate> is revalidated.
- Performance improvement depends on the size of the document and the updated portion.
  - 10M documents, up to 92% CPU reduction.
  - 10K documents, up to 60% CPU reduction.

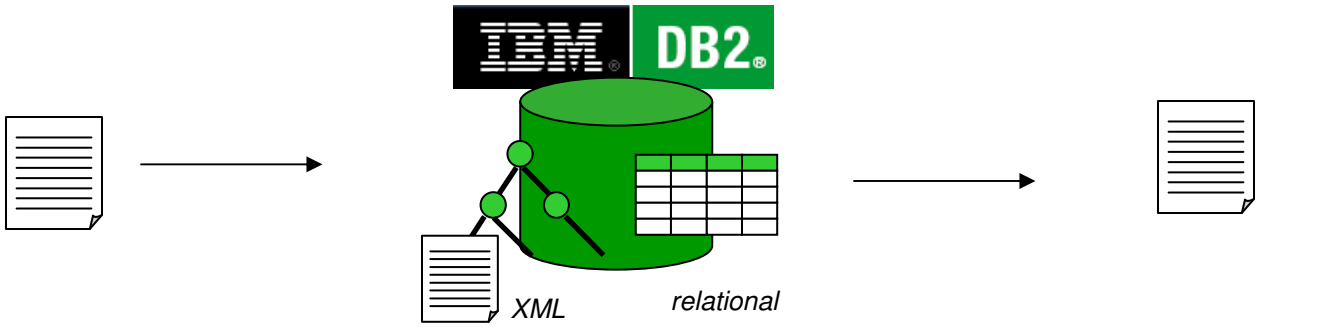
Customer Use Case

InformationOnDemand2013

November 3 – 7
Mandalay Bay | Las Vegas, NV

#ibmiod

Financial Customer in Europe handling SEPA transactions



1. XML document in SEPA format is inserted into XML column.
2. The XML column data is shredded and stored in relational columns in DB2 using XMTABLE function
3. Internal process modify the relational data
4. XML publishing functions are used to create the output XML documents from relational data.

Summary of Features and Enhancements

- **New Features**
  - XQuery *(retrofit to DB2 V10)*
  - XQuery prolog to control whitespace and namespace
  - Addition XQuery constructors
  - Implicitly add doc node during insert/update
- **Performance Enhancements**
  - Eliminate the hotspots during XML insert *(retrofit to DB2 9 and 10)*
  - Binary XML validation *(retrofit to DB2 10)*
  - Avoid validation of validated binary XML data during LOAD
  - Partial validation
  - XQuery performance enhancements
    - XML Operator Improvements
    - XQuery deferred construction
  - XMLTABLE performance enhancements
    - pushdown cast
    - Optimize Index Search Keys
    - Date/Time Predicate Pushdown

Acknowledgements and Disclaimers

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Communities

- **On-line communities, User Groups, Technical Forums, Blogs, Social networks, and more**
  - Find the community that interests you …
    - **Information Management** bit.ly/InfoMgmtCommunity
    - **Business Analytics** bit.ly/AnalyticsCommunity
    - **Enterprise Content Management** bit.ly/ECMCommunity
- **IBM Champions**
  - Recognizing individuals who have made the most outstanding contributions to Information Management, Business Analytics, and Enterprise Content Management communities
    - ibm.com/champion

Thank You

Your feedback is important!

- Access the Conference Agenda Builder to complete your session surveys
  - Any web or mobile browser at http://iod13surveys.com/surveys.html
  - Any Agenda Builder kiosk onsite