IBM Chat with Lab for Greater China Group

- **Host:** Frank Ning, Manager, DB2 LUW Install and Up/Running Development
- **Executive introduction** (audio)

  Sal Vella, Vice President, Development, Distributed Data Servers and Data Warehousing
- **Presentation: DB2 pureXML Technical Overview**

  Shumin Wu, Software Engineer at the Silicon Valley Lab

Jan, 2011

Executive Introduction

Sal Vella

Vice President, Development, Distributed Data Servers and Data Warehousing

IBM Software Group

DB2 pureXML Technical Overview

- Chat with the Silicon Valley Lab for the Greater China Group

Shumin Wu, Software Engineer
and the DB2 Silicon Valley Lab Team

Jan, 2011

Agenda

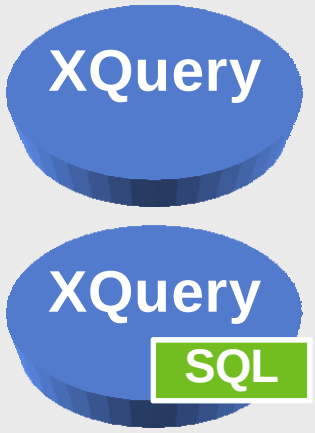
- XQuery
  - XQuery Collection Functions for DB2
  - XQuery Expression
  - XQuery FLWOR expression
  - XQuery Update – XQuery Transform
- SQL/XML
  - XMLQUERY (like SELECT for relational)
  - XMLEXISTS (like WHERE for relational)
  - XMLTABLE (to shred XML into relational or XML fragments)
  - XMLCAST (to cast SQL variables to or from XML data types)
  - Publishing Functions (to transform relational to XML)
  - Update
- Summary

Two Ways to Query XML in DB2

XQuery

– XQuery as the primary language

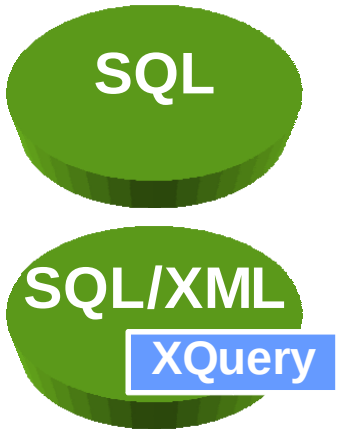
– Optional: SQL embedded in XQuery



SQL/XML

– SQL as the primary language

– Optional: XQuery embedded in SQL



http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606nicola

5

XQuery: DB2 Collection Functions

To access database data with XQuery

– `db2-fn:xmlcolumn ( xml-column-name )`

- Input argument is a string literal that identifies an XML column in a table, case sensitive
- `XQuery db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo`
- Retrieves an entire XML column as a sequence of XML values

– `db2-fn:sqlquery ( full-select-sql-statement )`

- Input argument is interpreted as an SQL statement and parsed by the SQL parser
- SQL statement needs to return a single XML column
- `XQuery db2-fn:sqlquery('SELECT INFO FROM CUSTOMER WHERE CID=6')/customerinfo`
- Returns an XML sequence that results from the full select

The XML sequence that is returned by these functions can contain any XML values, including nodes and atomic values

CID	INFO
1000	<xml/>
1001	<xml/>

6

XQuery on One Slide

- **Prolog: declares namespaces and sets up the query environment (optional)**
- **Body: composable expressions:**
  - Literals & variables
  - Function calls
  - Path expressions
  - Predicates
  - Arithmetic
  - Comparisons
  - and, or
  - Union, intersect, except
  - Constructors
  - FLWOR expressions
  - if ... then ... else
  - some, every
  - Cast
  - Transform

- *XQuery*
  prolog
  body

- *XQuery*

```
declare default element namespace http://sample.ns.com;
for $i in (1,2,5,10)
return $i+1
```

XQuery Expressions

Literals: "Hello" 12 4.9 4.9E-2

Comments

- `(: This is a comment :)`

Constructed Values: true() false() xs:date("2006-10-15")

Variables: $x $myVariable

Constructed Sequences:

- **$a, $b, $c**
- **8 to 11** is the same as **8, 9, 10, 11**
- Empty Sequence: **( )**

Even More XQuery Expressions

Creating document nodes

– **Document { }**

Function calls:

– **three-argument-function(1, 2, 3)**

– **two-argument-function(1, (2, 3))**

Path Expressions: see XPath (in session 1 – The Basics)

Combining Sequences: union intersect except

Path Expressions and Sequence Operators both return a sequence in document order with duplicates removed!

9

XQuery: Comparison Operators

Node Comparisons: `<<` `>>` `is`

– XQuery document order and node identity (`is`)

```
$customer//name[1] << $customer//addr[1]
```

Value comparisons: `eq` `ne` `gt` `ge` `lt` `le`

– Operands need to be single items

– Cast untypedAtomic to string

```
$customer/customerinfo[1]/@Cid eq “C2X”
```

General comparisons: `=` `!=` `>` `>=` `<` `<=`

– Operands are arbitrary sequences

– Have existential semantics (any item in $i equal to 5)

– Cast untypedAtomic to the type of the other operand

```
$customer/customerinfo/name = “Kathy Smith”
```

XQuery: Built-in Functions

Numeric

- `min`, `max`, `ceiling`, `floor` ...

String

- `contains`, `starts-with`, `replace` ...

Sequence

- `distinct-values`, `data`, `last`, `position`, `count` ...

Boolean

- `not`, `true` ...

publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.xml.doc/doc/xqrfncategory.html

XQuery Expressions - Constructors

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

```
<item partNum="C2X">
  <name> Radio </name>
 </item>
```

If the content of an element or attribute must be computed, use a nested expression enclosed in curly brackets

```
<item partNum="{$product/pid}">
  <name> {$product/name} </name>
</item>
```

- You can also use `element` and `attribute` to create new elements and attribute nodes

12

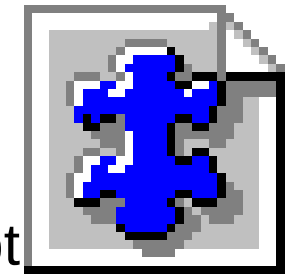
Agenda

- XQuery Collection Functions for DB2
- XQuery Expressions
- XQuery FLWOR Expression
- XQuery Update – XQuery Transform
- Summary

13

XQuery: The FLWOR Expression

```
create table dept(deptID char(8),  deptdoc xml);
```




```
XQUERY for  $d in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept
let $emp := $d//employee/name
where $d/@bldg = > 95
order by $d/@bldg
return   <EmpList>
                  {$d/@bldg, $emp}
          </EmpList>
```

```
<dept bldg="101">
   <employee id="901">
         <name>John Doe</name>
         <phone>408 555 1212</phone>
         <office>344</office>
    </employee>
    <employee id="902">
         <name>Peter Pan</name>
         <phone>408 555 9918</phone>
         <office>216</office>
    </employee>
</dept>
```

FOR: iterates through a sequence, binds variable to items

LET: binds a variable to a whole sequence of items

WHERE: eliminates items of the iteration

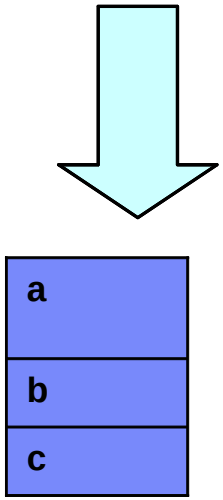
ORDER: reorders items of the iteration

RETURN: constructs query results

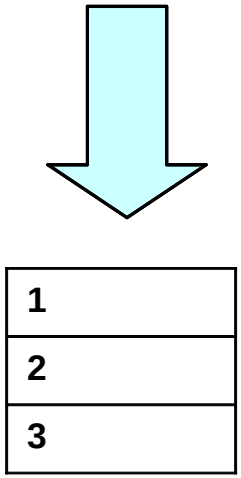
XQuery: The "for" Clause

FOR: iterates through a sequence, binds items to a variable, one at a time

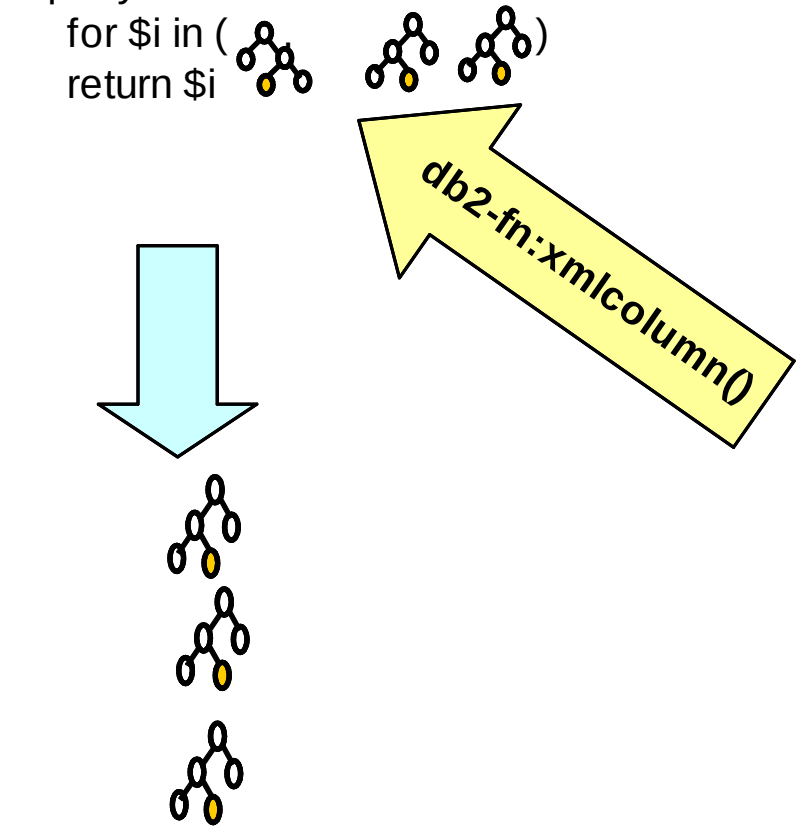
```
xquery
    for $i in (“a”,”b”,”c”)
    return $i
```




```
xquery
    for $i in (1,2,3)
    return $i
```




```
xquery
    for $i in (
    return $i
```



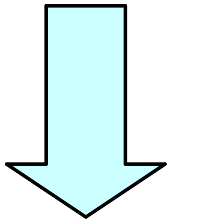
XQuery: "for" vs "let"

FOR: iterates through a sequence, binds items to a variable, one at a time

LET: binds a whole sequence of items to a variable

xquery for $i in (1,2,3)

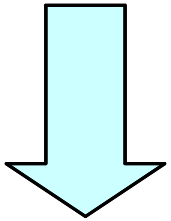
return $i



1
2
3

xquery let $i := (1,2,3)

return $i



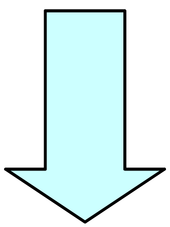
1 2 3

16

XQuery: "let" and "where" Clauses

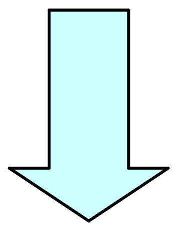
- LET: binds a variable to a whole sequence of items
- WHERE: eliminates items of the iteration

```
xquery
    for $i in (“a”,”b”,”c”)
    let $x := $i
    return $x
```



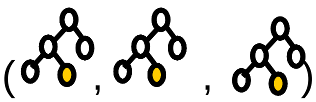
a
b
c

```
xquery
    for $i in (1,2,3)
    where $i > 2
    return $i
```



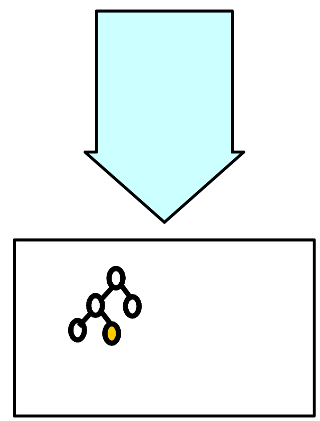
3

xquery

for $i in ()

where  > 2

return $i

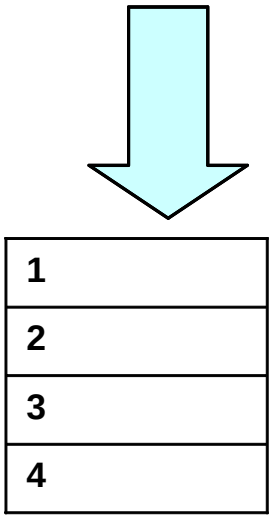


XQuery: "order by" and Element Construction

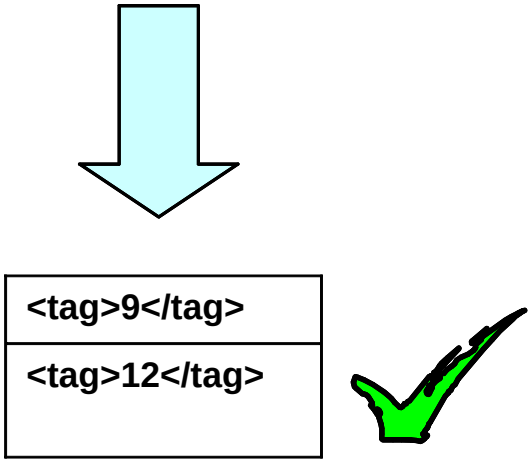
"order by" **sorts the output**

New elements can be constructed, curly brackets force evaluation of an enclosed expression, otherwise it's taken as a literal

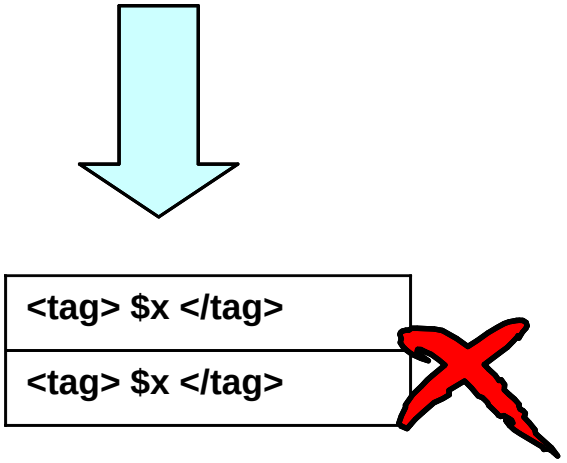
```
xquery
   for $i in (3,2,4,1)
   order by $i
   return $i
```




```
xquery
   for $i in (1,2,3,4)
   let $x := $i * 3
    where $x > 6
   return <tag> { $x } </
   tag>
```

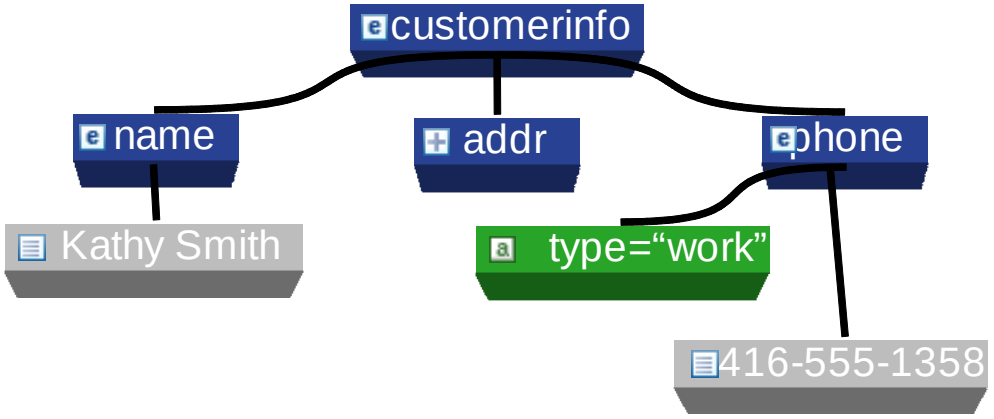



```
xquery
   for $i in (1,2,3,4)
   let $x := $i * 3
    where $x > 6
   return <tag> $x </tag>
```



XQuery: Iteration with DB2 Collection Functions

```
XQuery
  for $c in
    db2-fn:xmlcolumn
  ("CUSTOMER.INFO")
  return $c/name
```




```
<name> Kathy Smith </name>
<name> Jim Noodle </name>
<name> Richard Shoemaker </name>
2 records omitted
```

```
<?xml version="1.0" encoding="UTF-8"?>
<customerinfo Cid="1000">
  <name> Kathy Smith </name>
  <addr country="Canada">
    <street> 5 Rosewood </street>
    <city> Toronto </city>
    <prov-state> Ontario </prov-state>
    <pcode-zip> M6W 1E6 </pcode-zip>
  </addr>
  <phone type="work">
    416-555-1358
  </phone>
</customerinfo>
```

Same as: XQuery

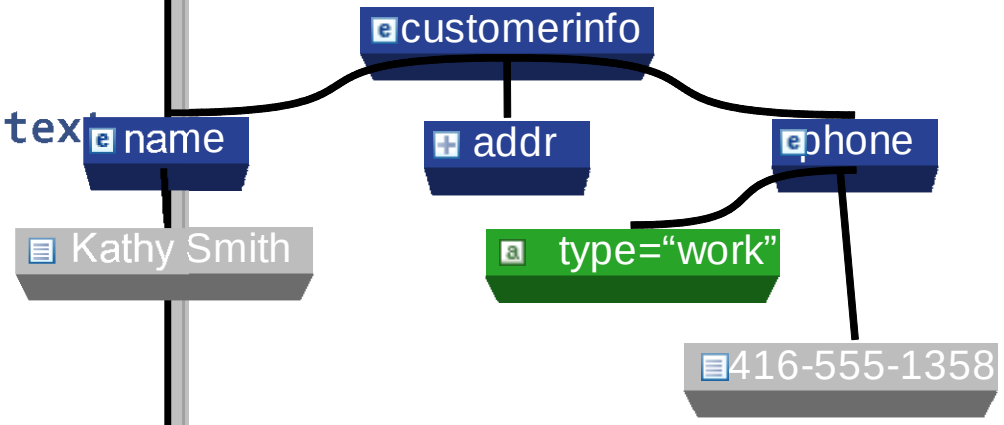
```
db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo/name
```


19

XQuery: Iteration with DB2 Collection Functions

```
XQuery
  for $c in
    db2-fn:xmlcolumn("CUSTOMER.INFO")
  let
    $phone :=    $c/customerinfo/phone/text
  ()

  where
  $c/customerinfo/name = "Kathy Smith"
  return $phone
```




```
416-555-1358
```

Same as: XQuery

```
db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo[name="Kathy Smith"]/phone/text()
```

XQuery: Positional Variables

```
XQuery
  for $c at $i in db2-fn:xmlcolumn('CUSTOMER.INFO')/customerinfo
  return <customer pos="{$i}">
           { $c/name, $c/phone[@type != "work"][1] }
         </customer>
```

```
<customer pos="1">
  <name> Kathy Smith </name>
</customer>
                                        (2 records omitted…)
<customer pos="4">
  <name> Robert Shoemaker </name>
  <phone type="home"> 416-555-2937 </phone>
</customer>
```


21

XQuery: Conditional Expressions

```
XQuery for $c in db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo
  return element customer { $c/name, $c/addr/city,
           element phones { for $p in $c/phone
               return if ($p/@type="work")
                      then <work> { $p/text() } </work>
                      else <misc> { $p/text() } </misc> } }
```

```
<?xml version="1.0" encoding="UTF-8"?>
<customerinfo Cid="1000">
  <name> Kathy Smith </name>
  <addr country="Canada">
    <street> 5 Rosewood </street>
    <city> Toronto </city>
    <prov-state> Ontario </prov-state>
    <pcode-zip> M6W 1E6 </pcode-zip>
  </addr>
  <phone type="work">
    416-555-1358
  </phone>
</customerinfo>
```

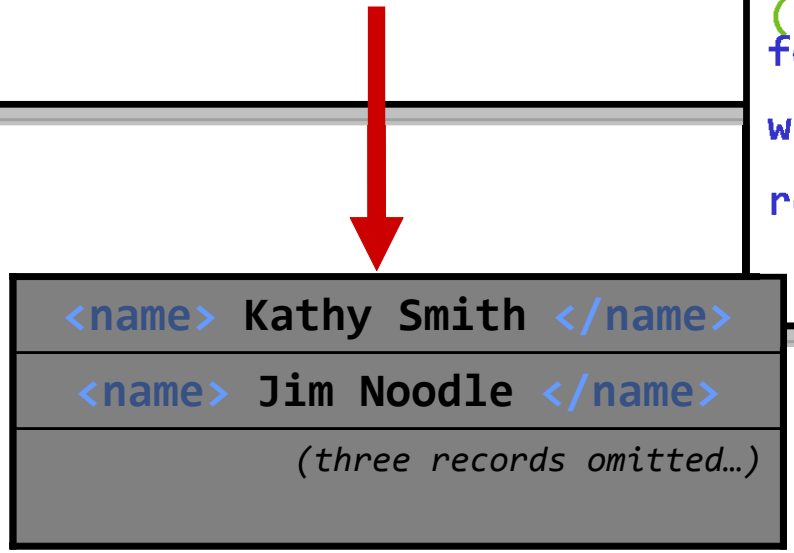
```
<?xml version="1.0" encoding="UTF-8"?>
<customer>
  <name> Kathy Smith </name>
  <city> Toronto </city>
  <phones>
    <work>
      416-555-1358
    </work>
  </phones>
</customer>
```

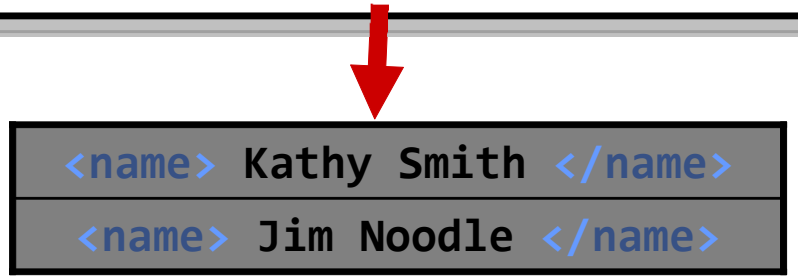
XQuery: Quantifiers

- **Quantifiers allow you to test if all (or at least one) element(s) in a sequence satisfy a given predicate.**
- **With an empty sequence (for a given predicate p):**
  - all yields true (no elements => all of them satisfy p)
  - one of them satisfies p)

```
XQuery
(: Quantifier: Some :)
for $c in
   db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo
where some $a in $c/phone
      satisfies ($a/@type="work")
return $c/name
```

```
XQuery
(: Quantifier: Every :)
for $c in
   db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo
where every $a in $c/phone
      satisfies ($a/@type="work")
return $c/name
```





Agenda - XQuery

- XQuery Collection Functions for DB2
- XQuery Expressions
- XQuery FLWOR Expression
- XQuery Update – XQuery Transform
- Summary

24

The XQuery TRANSFORM Expression: Semantics

Creates modified copies of existing nodes, documents, or sequences.

You typically use 4 clauses:

```
transform
   copy   $new := …
   modify    …$new…
  return   $new
```

Indicates that this is a transform expression

Copies the "input" XML data into a new variable

Specifies one or more updating expressions

Returns the modified copy of the input

Update the Zip Code

create table customer (cid integer, info XML);

```
UPDATE customer
SET info = XMLQUERY( '
            copy $new := $INFO
            modify do replace value of $new/customerinfo/addr/zipcode with 90111
            return  $new' )
WHERE cid = 1000;
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="work">963-289-4136</
phone>
</customerinfo>
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>90111</zipcode>
    </addr>
    <phone type="work">963-289-4136</
phone>
</customerinfo>
```

Update the Zip Code – with Parameter Markers

```
UPDATE customer
SET info = XMLQUERY( '
            copy $new := $INFO
            modify do replace value of $new/customerinfo/addr/zipcode with $p
            return  $new'
          PASSING CAST(? AS INTEGER) AS "p")
WHERE cid = ?;
```

White Paper: "**XQuery Update – Industry Use Cases**",
http://www.ibm.com/developerworks/data/library/techarticle/dm-0810malaika/

Delete the Phone

```
UPDATE customer
SET info = XMLQUERY( 'copy $new := $INFO
                      modify do delete $new/customerinfo/phone
                      return  $new')
WHERE cid = 1000;
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="work">963-289-4136</phone>
</customerinfo>
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>

</customerinfo>
```

Let's Add a Phone Number…

```
UPDATE customer
SET info = XMLQUERY( '
            copy $new := $INFO
            modify do insert <phone type="cell">777-555-3333</phone>
                    as last into $new/customerinfo
            return  $new')
WHERE cid = 1000;
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="work">963-289-4136</phone>
</customerinfo>
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="work">963-289-4136</phone>
    <phone type="cell">777-555-3333</
phone>
</customerinfo
```

Updating Repeating Elements (pick one!)

```
UPDATE customer
SET info = XMLQUERY( '
            copy $new := $INFO
            modify do replace value of $new/customerinfo/phone[@type="cell"]
                  with "111-222-4444"
            return  $new')
WHERE cid = 1000;
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="cell">777-555-3333</phone>
    <phone type="home">416-123-4567</phone>
</customerinfo>
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country="Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="cell">111-222-4444</phone>
    <phone type="home">416-123-4567</phone>
</customerinfo>
```

XQuery “if-then-else” for Conditional Updates

```
UPDATE customer
SET info = XMLQUERY( '
            copy $new := $INFO
            modify (
            if (fn:starts-with($new/customerinfo/phone,"416")) then
            do replace value of $new/customerinfo/phone with "416-123-4567"
               else
            do replace value of $new/customerinfo/phone with "905-871-0024" )
          return  $new')
WHERE cid = 1000;
```


31

Summary

A wide range of XQuery capabilities are available in DB2

Many of the capabilities can be combined with relational data

XQuery Collection Functions for DB2

XQuery Expressions

XQuery FLWOR Expression

XQuery Update – XQuery Transform

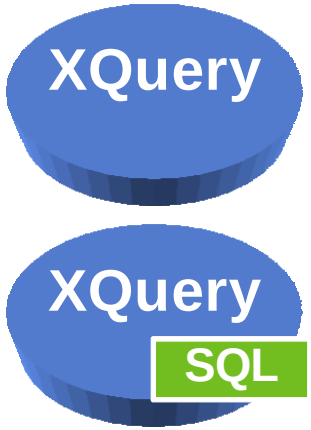
32

Two Ways to Query XML in DB2

XQuery

– XQuery as the primary language

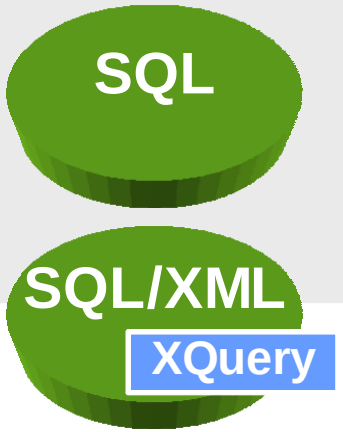
– Optional: SQL embedded in XQuery



SQL/XML

– SQL as the primary language

– Optional: XQuery embedded in SQL



http://www.ibm.com/developerworks/db2/library/techarticle/dm-0606nicola

Some New SQL/XML Functions in DB2 9

XMLQUERY Extracts data (like relational SELECT) and returns a sequence

XMLEXISTS Filters data (like relational WHERE clause)

XMLTABLE Returns the result sequence as a relational table (if possible)

XMLCAST Casts to or from an XML type

XMLPARSE Parses character/BLOB data, produces XML value

XMLVALIDATE Validates XML value against an XML schema

IS VALIDATED Checks if XML value has been validated

XMLQUERY & XMLEXISTS: Parameters

Used to embed XQuery in SQL

Pseudo functions

- Parameters are not separated by commas

1st parameter = XQuery statement

Other parameters bind values to external XQuery variables

XMLQUERY: Example

```
SELECT
   XMLQUERY('$i/customerinfo/name'
     PASSING INFO AS "i")
FROM
   CUSTOMER
```

CUSTOMER

CID	INFO
1000	<xml/>
1001	<xml/>

- **XMLQUERY**
  - Scalar function, applied once to each qualifying document
  - Evaluates an XPath (or XQuery) expression
  - Input arguments can be passed into the XQuery
    (e.g. column names, constants, parameter markers)
  - Returns a sequence of 0, 1 or multiple items from each document

XMLQUERY: How it Work

```
SELECT
   XMLQUERY('$i/customerinfo/name'
     PASSING INFO AS "i")
FROM
   CUSTOMER
```

Also valid in DB2 LUW 9.5+:

```
SELECT XMLQUERY('$INFO/customerinfo/
name')
FROM CUSTOMER
```

CUSTOMER

CID	INFO
1000	<xml/>
1001	<xml/>

- **Select iterates over all rows in the customer table**
  - For each row, "XMLQUERY" is invoked
    - The "passing" clause binds the variable "$i" to the XML document the "INFO" column of the current row
    - The XQuery expression is executed
    - XMLQUERY returns the result of the XQuery expression, a value of type XML

XMLQUERY: Case-sensitive

```
SELECT
   XMLQUERY('$Doc/customerinfo/name'
     PASSING INFO AS "Doc")
FROM
   CUSTOMER
```

- **Select can return a mixture of types (relational and XML)**
- **Note that XQuery is case-sensitive and SQL is not**
  - "CUSTOMER" is the table column name
    - Case-insensitive because it's in the SQL domain
  - "Doc" is the name of the XQuery variable
    - Case-sensitive because it's in the XQuery domain

38

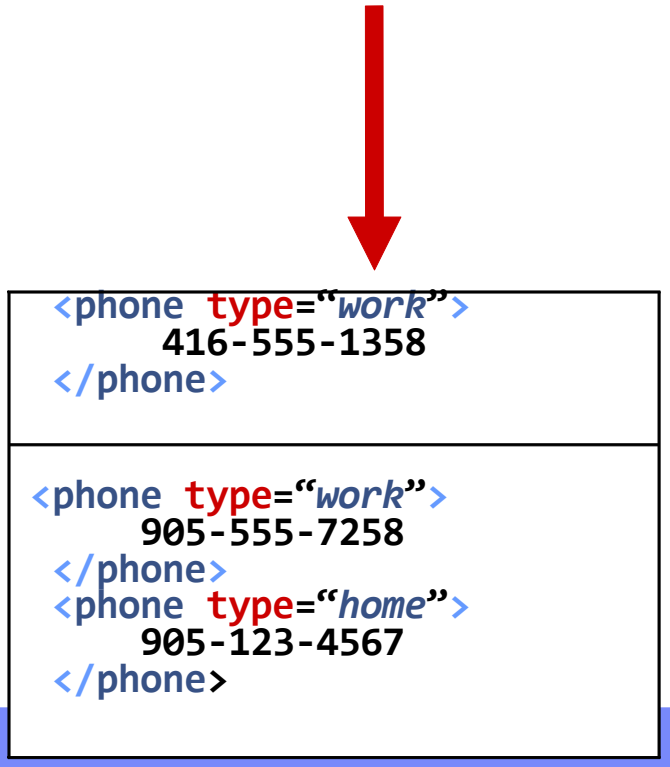
XMLQUERY vs XQuery Results

Contrast

- **XMLQUERY (SQL as primary language) vs. XQuery as primary language**

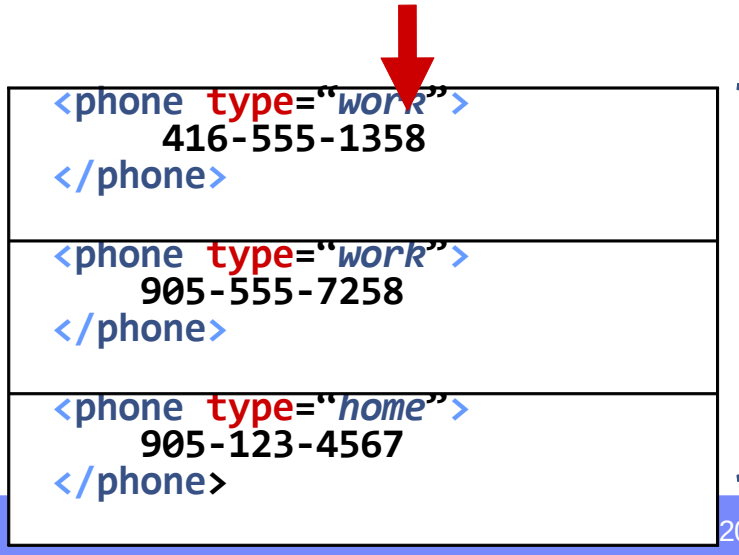
```
SELECT XMLQUERY(
'$INFO/customerinfo/phone')
FROM CUSTOMER
```

CID	INFO
1000	<customerinfo Cid="1000"> <name>Kathy Smith</name> <phone type="work"> 416-555-1358</phone> </customerinfo>
1002	<customerinfo Cid="1002"> <name>Jim Noodle</name> <phone type="work"> 905-555-7258</phone> <phone type="home"> 905-123-4567</phone> </customerinfo>



2 Records

```
XQuery
db2-fn:xmlcolumn('CUSTOMER.INFO')
/customerinfo/phone
```



3 Records

XMLEXISTS: Example

CUSTOMER

CID	INFO
1000	<xml/>
1001	<xml/>

```
SELECT INFO FROM CUSTOMER
WHERE
  XMLEXISTS('$INFO/customerinfo[addr/city="Calgary"]')
```

- **XMLEXISTS**
  - Predicate to test whether XQuery expression returns non-empty result
  - Returns FALSE if the XQuery expression returns empty sequence
  - Returns TRUE if the XQuery returns one or more items
  - Input arguments can be passed into the XQuery
    (e.g. column names, constants, parameter markers)

XMLEXISTS: How it Works

- Parameter syntax is identical to XMLQUERY
- Used as predicates in SQL WHERE clauses
- Acts as a Filter: Returns false if XQuery returns empty sequence, true otherwise

```
SELECT * FROM CUSTOMER WHERE XMLEXISTS('$INFO/customerinfo/name')
```

row is selected if info contains at least one name-element under an customerinfo

```
SELECT * FROM CUSTOMER
WHERE XMLEXISTS('$INFO/customerinfo[phone="905-555-7258"]')
```

row is selected if info contains at least one customerinfo-element with a phone-element equal to 905-555-7258

```
SELECT * FROM CUSTOMER
WHERE XMLEXISTS('$INFO/customerinfo/phone="905-555-7258"')
```

Attention

Row is always selected because XQuery never returns empty sequence

Without the brackets the path is navigating to phone – The path is not filtering

XMLEXISTS: Correctness

Watch out for correctness of XQuery expressions to get the expected result

```
SELECT CID
FROM CUSTOMER
WHERE
XMLEXISTS ('$INFO/customerinfo
[@Cid=1000]')
```

CID
1000

```
SELECT CID
FROM CUSTOMER
WHERE
XMLEXISTS ('$INFO/customerinfo/
@Cid=1000')
```

CID
1000
1001
1002

XMLEXISTS: Predicates With and Without Brackets

```
SELECT XMLQUERY('$INFO/customerinfo/name') FROM CUSTOMER
WHERE XMLEXISTS('$INFO/customerinfo/phone="416-555-1358"')
```

```
<customerinfo>
    <name> Kathy Smith </name>
    <phone> 416-555-1358 </phone>
</customerinfo>

<customerinfo>
    <name> Jim Noodle </name>
    <phone> 905-555-7258 </phone>
</customerinfo>

<customerinfo>
    <name> Robert Shoemaker </
name>
    <phone> 613-555-3278 </phone>
</customerinfo>
```

```
<name> Kathy Smith </name>
<name> Jim Noodle </name>
<name> Robert Shoemaker </name>

3 record(s) selected
```

```
<name> Kathy Smith </name>

1 record(s) selected
```

```
SELECT XMLQUERY('$INFO/customerinfo/name') FROM CUSTOMER
WHERE XMLEXISTS('$INFO/customerinfo[phone="416-555-1358"]')
```

Agenda

SQL/XML: XMLQUERY

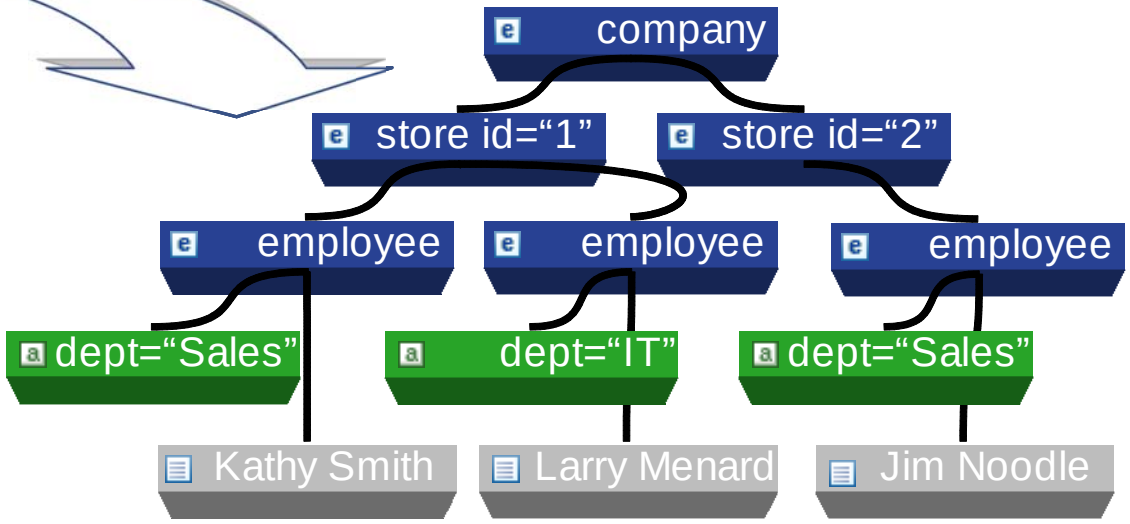
- SQL/XML: XMLEXISTS
- SQL/XML: XMLTABLE
- SQL/XML: XMLCAST
- SQL/XML: Publishing Functions
- SQL/XML: Update
- Summary

44

SQL/XML: Providing Multiple Transformations

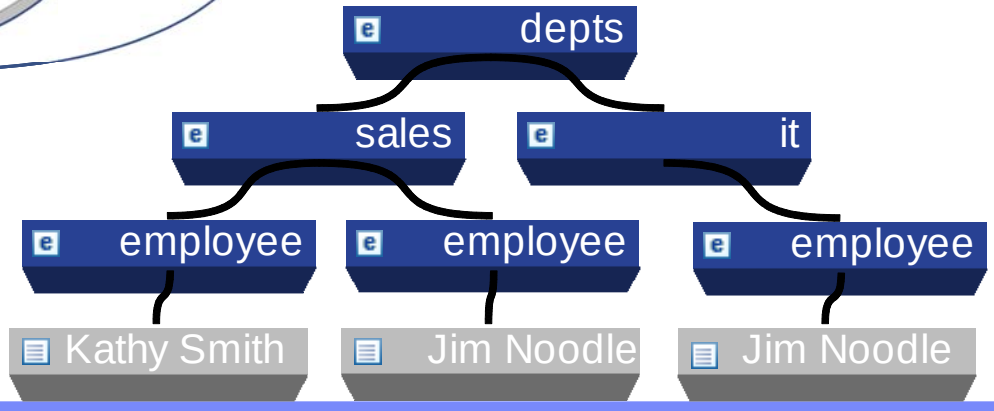
Publishing functions

NAME	STORE	DEPT
Kathy Smith	1	Sales
Larry Menard	1	IT
Jim Noodle	2	Sales



XQuery / XMLQUERY

XMLTABLE



45

XMLTABLE

```
<customerinfo Cid="1000">
  <name> Kathy Smith </
name>
  <addr country="Canada">
    <pcode-zip>
      M6W 1E6
    </pcode-zip>
  </addr>
</customerinfo>
```

XMLTABLE is invoked once per row

– For each row, XMLTABLE returns 0, 1 or more rows…

- with columns that are specified
- with values from the XML document

```
SELECT *
FROM CUSTOMER, XMLTABLE('$INFO/customerinfo'
  COLUMNS CID          INTEGER      PATH '@Cid',
          NAME         VARCHAR(32)  PATH 'name',
          COUNTRY      VARCHAR(32)  PATH 'addr/@country',
          PCODEZIP     XML          PATH 'addr/pcode-zip')
```

NAME	COUNTRY	PCODE-ZIP
Kathy Smith	Canada	pcode-zip M6W 1E6 pcode-zip
Jim Noodle	Canada	pcode-zip N9C 3T6 pcode-zip

XMLTABLE and Location Path

Column names in the XQuery domain are case-sensitive

If path returns empty sequence, then NULL is returned

The Path can return a sequence with multiple items

– If type is XML, then ok

– Otherwise, error

The XQuery statement can be complicated

– This syntax is zOS & LUW

The "PATH" is an XQuery expression also.

BY REF keywords indicate that the XQuery expression returns node references.

```
SELECT * FROM CUSTOMER,
  XMLTABLE('$i/customerinfo[addr/@country="Canada"]' PASSING INFO AS "i"
    COLUMNS
          NAME VARCHAR(32) DEFAULT "N/A"  PATH  'name',
          ZIP  XML BY REF                 PATH  'addr/pcode-zip')
```

More on XMLTABLE and Location Path

Column names in the XQuery domain are case-sensitive

If path returns empty sequence, then NULL is returned

The Path can return a sequence with multiple items

– If type is XML, then ok

– Otherwise, error

The XQuery statement can be complicated

– This syntax is LUW only

The "PATH" is an XQuery expression also.

BY REF keywords indicate that the XQuery expression returns node references.

```
SELECT * FROM
  XMLTABLE('for $c in db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo
            where $c/addr/@country="Canada" return $c'
    COLUMNS
            NAME VARCHAR(32) PATH  'if (name) then name else "N/A"',
            ZIP  XML BY REF  PATH  'for $z in addr/pcode-zip return $z')
```

XMLTABLE: Creating XML Documents

XMLTABLE can perform a simple shred into one table

Store some XML values as relational and some as XML

Only documents can be inserted so we create one

z/os does not support constructor syntax

```
SELECT PHONEINFO.*
FROM CUSTOMER,
     XMLTABLE('$INFO/customerinfo'
      COLUMNS
       CNAME     VARCHAR(32) PATH 'name',
       CONTACT   XML         PATH 'document { <phones> {phone} </
  phones> }')
AS PHONEINFO
```

CNAME	CONTACT
Kathy Smith	<phones> <phone type="work"> 416-555-1358 </phone> </phones>
Matt Foreman	<phones> <phone type="work"> 905-555-4789 </phone> <phone type="home"> 416-555-3376 </phone> </phones>

XMLTABLE: Row Mapping

Using XMLTABLE as a flexible transformer/shredder (row mapping)

z/os requires passing clause

```
SELECT PHONEINFO.*
FROM CUSTOMER,
     XMLTABLE('$INFO/customerinfo/phone'
      COLUMNS
        TYPE       VARCHAR(32) PATH '@type',
        PHONE      VARCHAR(32) PATH '.')
AS PHONEINFO
```

TYPE	PHONE
work	416-555-1358
work	905-555-7258
home	905-123-4567

XMLTABLE: Column Mapping

Using XMLTABLE as a flexible transformer/shredder (column mapping)

z/os requires passing clause

```
SELECT PHONEINFO.*
FROM CUSTOMER,
     XMLTABLE('$INFO/customerinfo/phone'
      COLUMNS
       OFFICE    VARCHAR(32) PATH '.[@type="work"]',
       HOME      VARCHAR(32) PATH '.[@type="home"]')
AS PHONEINFO
```

OFFICE	HOME
416-555-1358	Null
905-555-7258	Null
Null	905-123-4567

XMLCAST

Casts XML Schema types into SQL types and vice versa.

Why needed?

– XQuery only works with XML Schema types
  • "passing" clause often has to XML cast (explicit or implicit)
– XMLQuery only returns XML types

All values that can be represented in SQL types can also be represented as an XML type

– E.g. value “4” can be a SQL integer, SQL float, or an XML value
– XML types have subtypes: the XML Schema defined types

XMLCast

– never parses an XML document
– never serializes an XML value

XMLCAST: XML to SQL

XMLQUERY always returns type XML

To returns other types: XMLCAST

XMLCAST fails if XMLQUERY returns more than one item in a sequence. This sample query fails if a customer has two or more phone elements.

```
SELECT
  XMLCAST(
    XMLQUERY('$INFO/customerinfo/phone')
  AS VARCHAR(32) )
FROM
  CUSTOMER
```

XMLCAST: Ordering

Allows you to sort rows based on XML element or attribute values.

– XMLCAST is required because SQL cannot sort on XML types

```
SELECT * FROM  CUSTOMER
ORDER BY
  XMLCAST(
    XMLQUERY('$INFO/customerinfo/addr/pcode-zip')
  AS VARCHAR(32) )
```

Agenda

SQL/XML: XMLQUERY

- SQL/XML: XMLEXISTS
- SQL/XML: XMLTABLE
- SQL/XML: XMLCAST
- SQL/XML: Publishing Functions
- SQL/XML: Update
- Summary

SQL/XML Publishing Functions: Transforming Relational to XML

In DB2 8:

- XMLELEMENT creates an XML element
- XMLATTRIBUTES used within XMLELEMENT to create attributes
- XMLFOREST produces a forest of XML elements from a list of SQL values
- XMLCONCAT concatenates a list of XML values
- XMLNAMESPACES provides XML namespace declarations in an XML element
- XMLAGG groups a set of XML rows
- XML serialization functions XML2CLOB, XMLSERIALIZE - convert the XML value into a string value

Publishing Functions Example

PURCHASEORDER

POID	CUSTID	STATUS
5000	1002	Unshipped
5001	1003	Shipped
5002	1000	Shipped
5003	1002	Shipped
5004	1005	Shipped
5006	1002	Shipped

```
SELECT XMLELEMENT
  ( NAME "customer",
    XMLATTRIBUTES ( CUSTID AS "id" ),
    XMLAGG (
      XMLELEMENT (NAME "porder",
      XMLATTRIBUTES ( POID as "id" ), STATUS
      )
    )
  )
FROM  PURCHASEORDER
GROUP BY CUSTID
```

```
<customer id="1000">
  <porder id="5002"> Shipped </porder>
</customer>
<customer id="1002">
 <porder id="5000"> Unshipped </porder>
 <porder id="5003"> Shipped </porder>
 <porder id="5006"> Shipped </porder>
</customer>
<customer id="1003">
  <porder id="5001"> Shipped </porder>
</customer>
<customer id="1005">
  <porder id="5004"> Shipped </porder>
</customer>
```

Agenda

SQL/XML: XMLQUERY

- SQL/XML: XMLEXISTS
- SQL/XML: XMLTABLE
- SQL/XML: XMLCAST
- SQL/XML: Publishing Functions
- SQL/XML: Update
- Summary

Update the Zip Code

create table customer (cid integer, info XML);

```
UPDATE customer
SET info = XMLMODIFY( '
            replace value of node /customerinfo/addr/zipcode with “90111” ' )
WHERE cid = 1000;
```

DB2 zOS V10 Syntax

```
<customerinfo>
    <name>John Smith</name>
    <addr country=“Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="work">963-289-4136</
phone>
</customerinfo>
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country=“Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>90111</zipcode>
    </addr>
    <phone type="work">963-289-4136</
phone>
</customerinfo>
```

Summary

SQL/XML
- Incorporates XPath and XQuery in:
  - XMLQUERY
  - XMLEXISTS
  - XMLTABLE
- Note XQuery Constructors cannot be used in zOS

SQL/XML enables XML data to be queried, shredded, joined with relational or XML

SQL/XML also enables relational data to be published as XML

DB2 zOS V10 offers XML update capabilities through

Resources

- **References:**
  - DB2 pureXML enablement wiki: http://www.ibm.com/developerworks/wikis/display/db2xml
  - DB2 pureXML for zOS Wiki: http://www.ibm.com/developerworks/wikis/display/db2xml/DB2+for+zOS+pureXML
  - DB2 pureXML alphaWorks: http://www.alphaworks.ibm.com/tech/purexml (demo downloads)
  - DB2 pureXML cookbook: http://tinyurl.com/pureXML
  - Comprehensive coverage of pureXML in DB2 for Linux, UNIX, Windows andDB2 for z/OS
  - DB2 pureXML devotee site: http://www.ibm.com/developerworks/wikis/display/db2xml/devotee
  - DB2 pureXML bootcamps: http://www.ibm.com/developerworks/wikis/display/db2xml/BootCamps
- **Feedback**
  - Presentation format and contents
  - Additional DB2 topics you are interested
  - Follow on questions for the presentation
- **Contact: fning@ca.ibm.com**

61