



SDXML VT2026

Models and languages for semi-structured data and XML

Product-specific techniques IBM DB2

nikos dimitrakas
nikos@dsv.su.se
08-161295

Corresponding reading
Product documentation
Compendium with introduction to IBM DB2
Section 13.3 of the course book



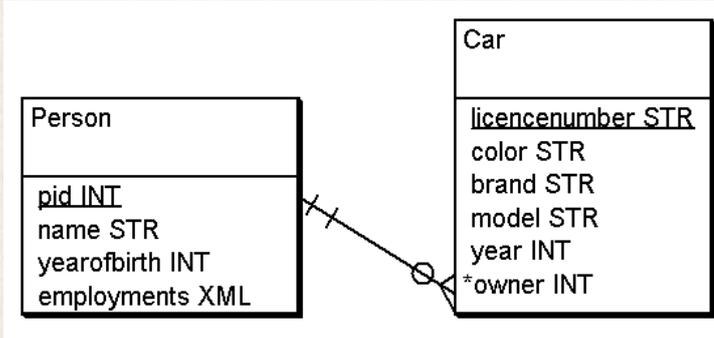
IBM DB2 12.1

- **Support for SQL/XML according to SQL:2006 with some exceptions**
 - No support for XML data types associated to schemas
 - Partial support for XPath and XQuery
- **Custom additions**
 - XMLGROUP, XMLROW, XSLTRANSFORM
 - XQUERY, extra XQuery functions
- **Support in earlier versions**
 - Data types XMLVARCHAR, XMLCLOB, XMLFILE
 - DTD validation
 - Composition and shredding according to templates (DAD files)
 - Modifying data inside XML
 - » update function
 - Extracting data from XML with XPath (functions)
 - » extractInteger, extractDate, extractVarchar, extractCLOB, etc.
 - » extractIntegers, extractDates, extractVarchars, etc.

Sample data

PERSON

pid	name	yearofbirth
1	John Higgins	1975
2	Steven Hendry	1973
3	Matthew Stevens	1982
4	Ronnie O'Sullivan	1980
5	Ken Doherty	1974
6	Steve Davis	1960
7	Paul Hunter	1983
8	Neil Robertson	1982



CAR

licencenumber	color	brand	model	year	owner
ABC123	black	NISSAN	Cherry	1995	1
CCD457	blue	FIAT	Forza	2001	2
DKL998	green	SAAB	9000C	1998	3
RSQ199	black	NISSAN	Micra	1999	4
WID387	red	FIAT	Nova	2003	5
ROO197	blue	SAAB	900i	1982	3
TYD226	black	NISSAN	Cherry	1990	1
PTF357	red	VOLVO	V70	2001	6
DAVIS1	red	VOLVO	V90	2007	6

Sample data

pid employments

1	<root><employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/> <employment startdate="2009-04-15" employer="UPC"/></root>
2	<root><employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB"/> <employment startdate="2003-08-01" employer="UPC"/> <employment startdate="2006-11-01" employer="ABB"/></root>
3	<root><employment startdate="2003-01-10" employer="UPC"/> </root>
4	<root><employment startdate="2002-03-10" enddate="2010-05-22" employer="LKP"/> <employment startdate="2010-08-15" employer="STG"/></root>
5	<root><employment startdate="2002-02-12" enddate="2003-05-11" employer="LKP"/> <employment startdate="2003-05-12" enddate="2003-12-02" employer="ABB"/> <employment startdate="2003-12-06" enddate="2005-02-17" employer="LKP"/> <employment startdate="2005-02-18" enddate="2008-05-16" employer="FFD"/> <employment startdate="2008-06-02" employer="STG"/></root>
6	<root><employment startdate="2001-01-05" enddate="2005-12-31" employer="ABB"/> <employment startdate="2006-01-15" enddate="2009-01-22" employer="LKP"/> <employment startdate="2009-02-01" employer="FFD"/> <employment startdate="2009-02-01" employer="XAB"/></root>
7	<root><employment startdate="2004-01-10" enddate="2008-09-29" employer="FFD"/> <employment startdate="2008-10-01" enddate="2010-11-20" employer="LKP"/></root>
8	<root><employment startdate="2006-02-03" enddate="2008-10-30" employer="UPC"/> <employment startdate="2008-11-20" employer="ABB"/></root>



DB2 - data type

- **XML**
 - No support for explicit association to XML Schema or DTD
 - Accepts well-formed XML documents and XML fragments
- **Validation**
 - Support for XML Schema
 - No support for DTD
 - Function XMLVALIDATE
 - » explicit schema
 - » implicit schema
 - Registration of XML Schema
- **VALIDATED constraint**
 - Checks that an XML value is validated
 - » in general
 - » according to specific XML Schema



DB2 - XML Schema Repository

- **Support for XML Schema and DTD**
- **REGISTER XMLSCHEMA**
- **ADD XMLSCHEMA**
- **COMPLETE XMLSCHEMA**
- **UPDATE XMLSCHEMA**
- **REGISTER XSROBJECT**



DB2 - SQL/XML

- Supports the following functions
 - XMLELEMENT
 - XMLATTRIBUTES
 - XMLFOREST
 - XMLCONCAT
 - XMLCOMMENT
 - XMLPI
 - XMLNAMESPACES
 - XMLAGG
 - XMLTEXT
 - XMLDOCUMENT
 - XMLQUERY (only SEQUENCE)
 - XMLTABLE
 - XMLEXISTS (classified as a predicate, not as a function)
 - XMLSERIALIZE
 - XMLCAST (classified as an expression, not as a function)
 - XMLPARSE
 - XMLVALIDATE



DB2 - XMLROW

- Returns one XML document per row

```
SELECT XMLROW(name, yearofbirth, pid)
FROM Person
```

```
<row><NAME>John Higgins</NAME><YEAROFBIRTH>1975</YEAROFBIRTH><PID>1</PID></row>
<row><NAME>Stephen Hendry</NAME><YEAROFBIRTH>1973</YEAROFBIRTH><PID>2</PID></row>
<row><NAME>Matthew Stevens</NAME><YEAROFBIRTH>1982</YEAROFBIRTH><PID>3</PID></row>
<row><NAME>Ronnie O'Sullivan</NAME><YEAROFBIRTH>1980</YEAROFBIRTH><PID>4</PID></row>
<row><NAME>Ken Doherty</NAME><YEAROFBIRTH>1974</YEAROFBIRTH><PID>5</PID></row>
<row><NAME>Steve Davis</NAME><YEAROFBIRTH>1960</YEAROFBIRTH><PID>6</PID></row>
<row><NAME>Paul Hunter</NAME><YEAROFBIRTH>1983</YEAROFBIRTH><PID>7</PID></row>
<row><NAME>Neil Robertson</NAME><YEAROFBIRTH>1982</YEAROFBIRTH><PID>8</PID></row>
```



DB2 - XMLROW - element names

9

- Offers configuration of element names

```
SELECT XMLROW(name AS "FullName",
              yearofbirth AS "YoB",
              pid AS ID
              OPTION ROW "Someone")
FROM Person
```

```
<Someone><FullName>John Higgins</FullName><YoB>1975</YoB><ID>1</ID></Someone>
<Someone><FullName>Stephen Hendry</FullName><YoB>1973</YoB><ID>2</ID></Someone>
<Someone><FullName>Matthew Stevens</FullName><YoB>1982</YoB><ID>3</ID></Someone>
<Someone><FullName>Ronnie O'Sullivan</FullName><YoB>1980</YoB><ID>4</ID></Someone>
<Someone><FullName>Ken Doherty</FullName><YoB>1974</YoB><ID>5</ID></Someone>
<Someone><FullName>Steve Davis</FullName><YoB>1960</YoB><ID>6</ID></Someone>
<Someone><FullName>Paul Hunter</FullName><YoB>1983</YoB><ID>7</ID></Someone>
<Someone><FullName>Neil Robertson</FullName><YoB>1982</YoB><ID>8</ID></Someone>
```

```
SELECT XMLELEMENT(NAME "Someone",
                  XMLFOREST(name AS "FullName",
                             yearofbirth AS "YoB", pid AS ID))
FROM Person
```



DB2 - XMLROW - attributes

10

- Possible to request attributes instead of elements

```
SELECT XMLROW(name AS "FullName",
              yearofbirth AS "YoB",
              pid AS ID
              OPTION ROW "Someone" AS ATTRIBUTES)
FROM Person
```

```
<Someone FullName="John Higgins" YoB="1975" ID="1"/>
<Someone FullName="Stephen Hendry" YoB="1973" ID="2"/>
<Someone FullName="Matthew Stevens" YoB="1982" ID="3"/>
<Someone FullName="Ronnie O'Sullivan" YoB="1980" ID="4"/>
<Someone FullName="Ken Doherty" YoB="1974" ID="5"/>
<Someone FullName="Steve Davis" YoB="1960" ID="6"/>
<Someone FullName="Paul Hunter" YoB="1983" ID="7"/>
<Someone FullName="Neil Robertson" YoB="1982" ID="8"/>
```

```
SELECT XMLELEMENT(NAME "Someone",
                  XMLATTRIBUTES(name AS "FullName",
                                 yearofbirth AS "YoB", pid AS ID))
FROM Person
```



DB2 - XMLGROUP

- Returns many rows as one XML document
 - Aggregate function

```
SELECT XMLGROUP(name, yearofbirth, pid)
FROM Person
```

```
<rowset>
  <row>
    <NAME>John Higgins</NAME>
    <YEAROFBIRTH>1975</YEAROFBIRTH>
    <PID>1</PID>
  </row>
  <row>
    <NAME>Stephen Hendry</NAME>
    <YEAROFBIRTH>1973</YEAROFBIRTH>
    <PID>2</PID>
  </row>
  <row>
    <NAME>Matthew Stevens</NAME>
    <YEAROFBIRTH>1982</YEAROFBIRTH>
    <PID>3</PID>
  </row>
  ...
</rowset>
```



DB2 - XMLGROUP - element names

- Offers configuration of element names

```
SELECT XMLGROUP(name AS "FullName", yearofbirth AS "YoB",
                pid AS ID OPTION ROOT "Everyone" ROW "Someone")
FROM Person
```

```
<Everyone>
  <Someone>
    <FullName>John Higgins</FullName>
    <YoB>1975</YoB>
    <ID>1</ID>
  </Someone>
  <Someone>
    <FullName>Stephen Hendry</FullName>
    <YoB>1973</YoB>
    <ID>2</ID>
  </Someone>
  <Someone>
    <FullName>Matthew Stevens</FullName>
    <YoB>1982</YoB>
    <ID>3</ID>
  </Someone>
  ...
</Everyone>
```



DB2 - XMLGROUP - attributes

- Possible to request attributes instead of elements

```
SELECT XMLGROUP(name AS "FullName", yearofbirth AS "YoB",
                pid AS ID OPTION AS ATTRIBUTES
                ROOT "Everyone" ROW "Someone")
FROM Person
```

```
<Everyone>
  <Someone FullName="John Higgins" ID="1" YoB="1975"/>
  <Someone FullName="Stephen Hendry" ID="2" YoB="1973"/>
  <Someone FullName="Matthew Stevens" ID="3" YoB="1982"/>
  <Someone FullName="Ronnie O'Sullivan" ID="4" YoB="1980"/>
  <Someone FullName="Ken Doherty" ID="5" YoB="1974"/>
  <Someone FullName="Steve Davis" ID="6" YoB="1960"/>
  <Someone FullName="Paul Hunter" ID="7" YoB="1983"/>
  <Someone FullName="Neil Robertson" ID="8" YoB="1982"/>
</Everyone>
```

```
SELECT XMLELEMENT(NAME "Everyone",
                  XMLAGG(XMLROW(name AS "FullName", yearofbirth AS "YoB",
                                pid AS ID OPTION ROW "Someone" AS ATTRIBUTES)))
FROM Person
```



DB2 - XSLTRANSFORM

- Transforms an XML document according to an XSLT document (XSLT 1.0)
 - XSLTRANSFORM(XML-value USING XSLT-value)

```
<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" />
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:apply-templates select="//@employer" />
    </xsl:element>
  </xsl:template>
  <xsl:template match="@employer">
    <xsl:element name="Employer">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
</xsl:transform>
```



DB2 - XSLTRANSFORM

```
SELECT XSLTRANSFORM(employments USING
'<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:apply-templates select="//@employer"/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="@employer">
    <xsl:element name="Employer">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
</xsl:transform>')
```

FROM Person
WHERE name = 'Stephen Hendry'

```
<Employers>
  <Employer>ABB</Employer>
  <Employer>UPC</Employer>
  <Employer>ABB</Employer>
</Employers>
```



DB2 - XSLTRANSFORM

```
CREATE TABLE xsIt (name VARCHAR(15) NOT NULL PRIMARY KEY,
value XML NOT NULL)
```

```
INSERT INTO xsIt VALUES
('employers',
```

```
'<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:apply-templates select="//@employer"/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="@employer">
    <xsl:element name="Employer">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
</xsl:transform>')
```

```
SELECT XSLTRANSFORM(employments USING xsIt.value)
FROM Person, xsIt
WHERE Person.name = 'Stephen Hendry' AND xsIt.name = 'employers'
```



DB2 - XMLCAST

- Converts between XML and other data types
- Requires a value or a node

```
SELECT name, XMLCAST(XMLQUERY('count( $EMPLOYMENTS//employment)') AS INT)
FROM Person
```

```
SELECT name, XMLCAST(
XMLQUERY('($EMPLOYMENTS//@employer[not(../@enddate)])[1]') AS VARCHAR(50))
FROM Person
```

```
SELECT name, XMLCAST(XMLQUERY(
'string-join($EMPLOYMENTS//@employer[not(../@enddate)], ", "') AS VARCHAR(50))
FROM Person
```

```
SELECT XMLCAST(name AS XML)
FROM Person
```

May behave weirdly when converting sequences from XML to something else.



DB2 - XQUERY

- Support for XQuery
 - XQuery statements after the keyword XQUERY

```
XQUERY
for $a in (1,2,3)
return element Number {$a}
```

Three rows in the result:

```
<Number>1</Number>
<Number>2</Number>
<Number>3</Number>
```

```
XQUERY
element Result {for $a in (1,2,3)
return element Number {$a}}
```

```
<Result>
<Number>1</Number><Number>2</Number><Number>3</Number>
</Result>
```

DB2 - XQUERY - sqlquery

- XQuery function for SQL inside XQuery
 - db2-fn:sqlquery
 - » supports parameters
 - The result is a sequence in the XQuery context
 - The SELECT statement must return one XML column

XQUERY

```
for $a in distinct-values(db2-fn:sqlquery("SELECT
employments FROM Person")//@employer)
return element Company {$a}
```

```
<Company>ABB</Company>
<Company>UPC</Company>
<Company>LKP</Company>
<Company>STG</Company>
<Company>FFD</Company>
<Company>XAB</Company>
```

DB2 - XQUERY - sqlquery

- Parameters

XQUERY

```
for $a in distinct-values(db2-fn:sqlquery("SELECT
employments FROM Person WHERE yearofbirth >
parameter(1)", 1980)//@employer)
return element Company {$a}
```

```
<Company>UPC</Company>
<Company>FFD</Company>
<Company>LKP</Company>
<Company>ABB</Company>
```



DB2 - XQUERY - sqlquery

XQUERY

```
let $a := db2-fn:sqlquery('SELECT XMLELEMENT(NAME  
"Person", name) FROM Person WHERE yearofbirth < 1975')  
return element People {$a}
```

<People>

<Person>Stephen Hendry</Person>

<Person>Ken Doherty</Person>

<Person>Steve Davis</Person>

</People>



DB2 - XQUERY - xmlcolumn

- XQuery function to get data from an XML column
 - db2-fn:xmlcolumn
 - Table name and column name are case-sensitive

XQUERY

```
for $a in distinct-values(db2-  
fn:xmlcolumn('PERSON.EMPLOYMENTS')//@employer)  
return element Company {$a}
```

<Company>ABB</Company>

<Company>UPC</Company>

<Company>LKP</Company>

<Company>STG</Company>

<Company>FFD</Company>

<Company>XAB</Company>



DB2 - XML DML

- **XQuery Update Facility**
 - some syntax differences
- **transform statements**
 - (transform clause)
 - » do delete
 - » do insert
 - » do rename
 - » do replace
 - copy clause
 - modify clause
 - return clause
 - keywords for do insert
 - » before
 - » after
 - » as first into
 - » as last into
 - » into
 - keywords for do replace
 - » (value of) ... with ...
 - keywords for do rename
 - » as



DB2 - XML DML

- **The whole XML value in a cell must be replaced**

UPDATE Person

**SET employments = XMLQUERY('transform statement'
PASSING employments)**

WHERE ...



DB2 - transform - insert

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify do insert <a>123 as first into \$x

return \$x

```
<root>
  <a>123</a>
  <a>456</a>
  <a>789</a>
</root>
```



DB2 - transform - insert

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify do insert <a>123 before \$x/a[text() = 789]

return \$x

```
<root>
  <a>456</a>
  <a>123</a>
  <a>789</a>
</root>
```



DB2 - transform - insert

XQUERY

transform

```
copy $x := <root><a>456</a><a>789</a></root>
```

```
modify do insert attribute c {5} into $x/a[text() = 789]
```

```
return $x
```

```
<root>
```

```
  <a>456</a>
```

```
  <a c="5">789</a>
```

```
</root>
```



DB2 - transform - insert

```
UPDATE Person
```

```
SET employments = XMLQUERY('
```

```
  transform
```

```
  copy $ne := $e
```

```
  modify do insert element employment {
```

```
    attribute startdate {"2011-09-01"},
```

```
    attribute employer {"LBM"}} as last into $ne/root
```

```
  return $ne' PASSING employments AS "e")
```

```
WHERE pid = 3
```

```
SELECT employments FROM Person WHERE pid = 3
```

```
<root>
```

```
  <employment startdate="2003-01-10" employer="UPC"/>
```

```
  <employment startdate="2011-09-01" employer="LBM"/>
```

```
</root>
```



DB2 - transform - delete

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify do delete \$x/a[text() = 789]

return \$x

<root>

<a>456

</root>



DB2 - transform - delete

SELECT XMLQUERY('transform

copy \$ne := \$e

modify do delete \$ne//employment[2]

return \$ne' PASSING employments AS "e")

FROM Person

WHERE pid = 3

<root>

<employment startdate="2003-01-10" employer="UPC"/>

</root>

SELECT employments FROM person WHERE pid = 3

<root>

<employment startdate="2003-01-10" employer="UPC"/>

<employment startdate="2011-09-01" employer="LBM"/>

</root>



DB2 - transform - delete

```
UPDATE Person
SET employments = XMLQUERY('transform
    copy $ne := $e
    modify do delete $ne//employment[2]
    return $ne' PASSING employments AS "e")
```

```
WHERE pid = 3
```

```
SELECT employments FROM Person WHERE pid = 3
```

```
<root>
  <employment startdate="2003-01-10" employer="UPC"/>
</root>
```



DB2 - transform - replace

```
XQUERY
transform
copy $x := <root><a>456</a><a>789</a></root>
modify do replace $x/a[text() = 789] with <b>123</b>
return $x
```

```
<root>
  <a>456</a>
  <b>123</b>
</root>
```



DB2 - transform - replace

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify do replace value of \$x/a[text() = 789] with 123

return \$x

```
<root>
```

```
  <a>456</a>
```

```
  <a>123</a>
```

```
</root>
```



DB2 - transform - replace

XQUERY

transform

copy \$x := <root>456<a>789</root>

modify do replace \$x/a[1]/@b with attribute f {"ddd"}

return \$x

```
<root>
```

```
  <a f="ddd">456</a>
```

```
  <a>789</a>
```

```
</root>
```



DB2 - transform - rename

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify do rename \$x/a[2] as "b"

return \$x

<root>

<a>456

789

</root>



DB2 - transform – many, loop

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify for \$a in \$x/a

return do rename \$a as "b"

return \$x

<root>

456

789

</root>



DB2 - transform - many, list

XQUERY

transform

copy \$x := <root><a>456<a>789</root>

modify (do replace value of \$x/a[text() = 789] with 123,

do rename \$x/a[2] as "b",

do insert attribute c {5} into \$x/a[text() = 789])

return \$x

```
<root>
```

```
  <a>456</a>
```

```
  <b c="5">123</b>
```

```
</root>
```



DB2 & JSON

• Functions (SQL/JSON)

- JSON_VALUE
- JSON_QUERY
- JSON_VAL
- JSON_OBJECT
- JSON_ARRAY
- JSON_TABLE
- JSON_UPDATE
- JSON_EXISTS (predicate)

Summary

- **DB2 follows the SQL standard quite well**
- **No XQuery 3**
- **No dynamic node names**
- **Few supported XPath axes:**
 - child, parent, self, attribute, descendant, and descendant-or-self
- **Use of standard SQL makes migration easier**
 - Avoid product specific solutions when possible
 - Avoid deprecated solutions when possible

What to do next

- **Introduction to DB2 & XML (compendium)**
 - Introduction to DB2 and IBM Data Studio
 - Examples
- **Assignment 6 (DB2 & XML)**
 - After seminar 3