



SDXML VT2024

Models and languages for semi-structured data and XML

SQL/XML

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

Corresponding reading
Section 8.3 and chapter 15 of the course book



SQL/XML

- **Support for XML in relational databases according to SQL:2003, SQL:2006, SQL:2008, SQL:2011**
 - SQL/XML: part of the SQL standard
 - SQLX: the group behind SQL/XML
 - SQLXML: Microsoft's additions to SQL for XML
- **XML data type**
- **Composition of XML (XML as the result SQL SELECT)**
- **Support for execution of XQuery**
 - Including XQuery Update Facility (since SQL:2011)
- **Support for interoperability between XQuery and SQL**

XML data type

- **Store XML, not the serialized version (that would be an ordinary CLOB)**
 - According to the XQuery model since SQL:2006
- **Every cell is an XML document (or possibly a fragment)**
- **Support for validation**

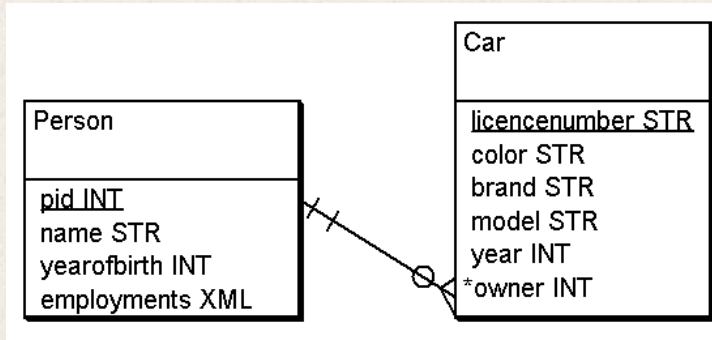
SQL functions

- | | |
|------------------------|-----------------------|
| • XMLELEMENT | • XMLQUERY |
| • XMLATTRIBUTES | • XMLTABLE |
| • XMLFOREST | • XMLEXISTS |
| • XMLCONCAT | |
| • XMLCOMMENT | • XMLSERIALIZE |
| • XMLPI | • XMLPARSE |
| • XMLNAMESPACES | • XMLCAST |
| • XMLAGG | |
| • XMLTEXT | • XMLVALIDATE |
| • XMLDOCUMENT | |

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

- The column employments according to this DTD:

<!ELEMENT root (employment*)>

<!ELEMENT employment EMPTY>

<!ATTLIST employment

startdate CDATA #REQUIRED

enddate CDATA #IMPLIED

employer CDATA #REQUIRED>

Sample data

- The column employments according to this XML Schema:

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="root">
    <complexType>
      <sequence>
        <element name="employment" type="EmploymentType"
          minOccurs="0" maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>
  <complexType name="EmploymentType">
    <attribute name="startdate" type="date" use="required" />
    <attribute name="enddate" type="date" use="optional" />
    <attribute name="employer" type="string" use="required" />
  </complexType>
</schema>
```

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>

XMLELEMENT

- **Creates an XML element**
 - Specify the element's name after the keyword **NAME**
 - Specify the element's content

```
SELECT XMLELEMENT(NAME "Person", name)  
FROM Person
```

```
<Person>John Higgins</Person>  
<Person>Stephen Hendry</Person>  
<Person>Matthew Stevens</Person>  
<Person>Ronnie O'Sullivan</Person>  
<Person>Ken Doherty</Person>  
<Person>Steve Davis</Person>  
<Person>Paul Hunter</Person>  
<Person>Neil Robertson</Person>
```

XMLTEXT

- **Creates an XML text node**

```
SELECT XMLELEMENT(NAME "Person", XMLTEXT(name))  
FROM Person
```

Same effect as

```
SELECT XMLELEMENT(NAME "Person", name)  
FROM Person
```

XMLDOCUMENT

- **Creates an XML document node**

```
SELECT XMLDOCUMENT(XMLELEMENT(NAME "Person", name))  
FROM Person
```

Same effect (visibly) as

```
SELECT XMLELEMENT(NAME "Person", name)  
FROM Person
```

XMLATTRIBUTES

- **Creates XML attributes**
 - Used only inside XMLELEMENT

```
SELECT XMLELEMENT(NAME "Person", XMLATTRIBUTES(yearofbirth), name)  
FROM Person
```

```
<Person YEARTOFBIRTH="1975">John Higgins</Person>  
<Person YEARTOFBIRTH="1973">Stephen Hendry</Person>  
<Person YEARTOFBIRTH="1982">Matthew Stevens</Person>  
<Person YEARTOFBIRTH="1980">Ronnie O'Sullivan</Person>  
<Person YEARTOFBIRTH="1974">Ken Doherty</Person>  
<Person YEARTOFBIRTH="1960">Steve Davis</Person>  
<Person YEARTOFBIRTH="1983">Paul Hunter</Person>  
<Person YEARTOFBIRTH="1982">Neil Robertson</Person>
```

XMLATTRIBUTES

- Can create many attributes
- Attribute names can be specified

```
SELECT XMLELEMENT(NAME "Person",  
    XMLATTRIBUTES(yearofbirth AS "BirthYear",  
        (SELECT COUNT(*)  
        FROM Car  
        WHERE owner = pid) AS "NumberOfCars"),  
    name)  
FROM Person
```

```
<Person BirthYear="1975" NumberOfCars="2">John Higgins</Person>  
<Person BirthYear="1973" NumberOfCars="1">Stephen Hendry</Person>  
<Person BirthYear="1982" NumberOfCars="2">Matthew Stevens</Person>  
<Person BirthYear="1980" NumberOfCars="1">Ronnie O'Sullivan</Person>  
<Person BirthYear="1974" NumberOfCars="1">Ken Doherty</Person>  
<Person BirthYear="1960" NumberOfCars="2">Steve Davis</Person>  
<Person BirthYear="1983" NumberOfCars="0">Paul Hunter</Person>  
<Person BirthYear="1982" NumberOfCars="0">Neil Robertson</Person>
```

XMLELEMENT without content

- Does not require content
- Does not require attributes

```
SELECT XMLELEMENT(NAME "Person")  
FROM Person
```

```
<Person/>  
<Person/>  
<Person/>  
<Person/>  
<Person/>  
<Person/>  
<Person/>  
<Person/>
```

```
SELECT XMLELEMENT(NAME "Person", XMLATTRIBUTES(yearofbirth, name))  
FROM Person
```

```
<Person YEAROFBIRTH="1975" NAME="John Higgins"/>  
<Person YEAROFBIRTH="1973" NAME="Stephen Hendry"/>  
<Person YEAROFBIRTH="1982" NAME="Matthew Stevens"/>  
<Person YEAROFBIRTH="1980" NAME="Ronnie O'Sullivan"/>  
<Person YEAROFBIRTH="1974" NAME="Ken Doherty"/>  
<Person YEAROFBIRTH="1960" NAME="Steve Davis"/>  
<Person YEAROFBIRTH="1983" NAME="Paul Hunter"/>  
<Person YEAROFBIRTH="1982" NAME="Neil Robertson"/>
```

XMLCONCAT

- Merges two XML values into one

- Without:

```
SELECT XMLELEMENT(NAME "Name", name),  
       XMLELEMENT(NAME "YearOfBirth", yearofbirth)  
FROM Person
```

<Name>John Higgins</Name>	<YearOfBirth>1975</YearOfBirth>
<Name>Stephen Hendry</Name>	<YearOfBirth>1973</YearOfBirth>
<Name>Matthew Stevens</Name>	<YearOfBirth>1982</YearOfBirth>
<Name>Ronnie O'Sullivan</Name>	<YearOfBirth>1980</YearOfBirth>
<Name>Ken Doherty</Name>	<YearOfBirth>1974</YearOfBirth>
<Name>Steve Davis</Name>	<YearOfBirth>1960</YearOfBirth>
<Name>Paul Hunter</Name>	<YearOfBirth>1983</YearOfBirth>
<Name>Neil Robertson</Name>	<YearOfBirth>1982</YearOfBirth>

The result has two columns

XMLCONCAT

- With:

```
SELECT XMLCONCAT(XMLELEMENT(NAME "Name", name),  
                 XMLELEMENT(NAME "YearOfBirth", yearofbirth))  
FROM Person
```

```
<Name>John Higgins</Name><YearOfBirth>1975</YearOfBirth>  
<Name>Stephen Hendry</Name><YearOfBirth>1973</YearOfBirth>  
<Name>Matthew Stevens</Name><YearOfBirth>1982</YearOfBirth>  
<Name>Ronnie O'Sullivan</Name><YearOfBirth>1980</YearOfBirth>  
<Name>Ken Doherty</Name><YearOfBirth>1974</YearOfBirth>  
<Name>Steve Davis</Name><YearOfBirth>1960</YearOfBirth>  
<Name>Paul Hunter</Name><YearOfBirth>1983</YearOfBirth>  
<Name>Neil Robertson</Name><YearOfBirth>1982</YearOfBirth>
```

The result has one column!

XMLFOREST

- **Creates multiple simple elements at once**
 - Returns a fragment (sequence of elements)

```
SELECT XMLFOREST(name AS "Name",  
                  yearofbirth AS "YearOfBirth")  
FROM Person
```

```
<Name>John Higgins</Name><YearOfBirth>1975</YearOfBirth>  
<Name>Stephen Hendry</Name><YearOfBirth>1973</YearOfBirth>  
<Name>Matthew Stevens</Name><YearOfBirth>1982</YearOfBirth>  
<Name>Ronnie O'Sullivan</Name><YearOfBirth>1980</YearOfBirth>  
<Name>Ken Doherty</Name><YearOfBirth>1974</YearOfBirth>  
<Name>Steve Davis</Name><YearOfBirth>1960</YearOfBirth>  
<Name>Paul Hunter</Name><YearOfBirth>1983</YearOfBirth>  
<Name>Neil Robertson</Name><YearOfBirth>1982</YearOfBirth>
```

Nesting

```
SELECT XMLELEMENT(NAME "Person",  
                  XMLATTRIBUTES(pid AS "ID"),  
                  XMLELEMENT(NAME "Info",  
                              XMLATTRIBUTES(name AS "About"),  
                              XMLFOREST((SELECT COUNT(*)  
                                         FROM Car  
                                         WHERE owner = pid) AS "Cars",  
                                         yearofbirth AS "YoB")))  
FROM Person
```

```
<Person ID="1"><Info About="John Higgins"><Cars>2</Cars><YoB>1975</YoB></Info></Person>  
<Person ID="2"><Info About="Stephen Hendry"><Cars>1</Cars><YoB>1973</YoB></Info></Person>  
<Person ID="3"><Info About="Matthew Stevens"><Cars>2</Cars><YoB>1982</YoB></Info></Person>  
<Person ID="4"><Info About="Ronnie O'Sullivan"><Cars>1</Cars><YoB>1980</YoB></Info></Person>  
<Person ID="5"><Info About="Ken Doherty"><Cars>1</Cars><YoB>1974</YoB></Info></Person>  
<Person ID="6"><Info About="Steve Davis"><Cars>2</Cars><YoB>1960</YoB></Info></Person>  
<Person ID="7"><Info About="Paul Hunter"><Cars>0</Cars><YoB>1983</YoB></Info></Person>  
<Person ID="8"><Info About="Neil Robertson"><Cars>0</Cars><YoB>1982</YoB></Info></Person>
```

XMLAGG

- Aggregates many XML values into one
- An aggregate function (like COUNT, SUM, etc.)
 - Either all rows become a group
 - Or groups are created based on GROUP BY

```
SELECT XMLAGG(XMLELEMENT(NAME "Person", name))  
FROM Person
```

```
<Person>John Higgins</Person><Person>Stephen  
Hendry</Person><Person>Matthew Stevens</Person><Person>Ronnie  
O'Sullivan</Person><Person>Ken Doherty</Person><Person>Steve  
Davis</Person><Person>Paul Hunter</Person><Person>Neil  
Robertson</Person>
```

The whole result is one row and one column (one cell)

XMLAGG

- And a root element in order to get well-formed XML

```
SELECT XMLELEMENT(NAME "People",  
    XMLAGG(XMLELEMENT(NAME "Person", name)))  
FROM Person
```

```
<People><Person>John Higgins</Person><Person>Stephen  
Hendry</Person><Person>Matthew Stevens</Person><Person>Ronnie  
O'Sullivan</Person><Person>Ken Doherty</Person><Person>Steve  
Davis</Person><Person>Paul Hunter</Person><Person>Neil  
Robertson</Person></People>
```

The previous result indented

```
<People>
  <Person>John Higgins</Person>
  <Person>Stephen Hendry</Person>
  <Person>Matthew Stevens</Person>
  <Person>Ronnie O'Sullivan</Person>
  <Person>Ken Doherty</Person>
  <Person>Steve Davis</Person>
  <Person>Paul Hunter</Person>
  <Person>Neil Robertson</Person>
</People>
```

XMLAGG with GROUP BY

- One result per group
 - Aggregated vs grouped columns (as usual)

```
SELECT XMLELEMENT(NAME "Color",
                  XMLATTRIBUTES(color AS "Name"),
                  XMLAGG(XMLELEMENT(NAME "Car", licencenumber)))
FROM Car
GROUP BY color
```

```
<Color Name="black"><Car>ABC123</Car><Car>TYD226</Car><Car>RSQ199</Car></Color>
<Color Name="blue"><Car>CCD457</Car><Car>ROO197</Car></Color>
<Color Name="green"><Car>DKL998</Car></Color>
<Color Name="red"><Car>PTF357</Car><Car>WID387</Car><Car>DAVIS1</Car></Color>
```



XMLAGG with GROUP BY

```
SELECT XMLELEMENT(NAME "Color",
    XMLATTRIBUTES(color AS "Name"),
    XMLAGG(XMLELEMENT(NAME "Car",
        XMLFOREST(licencenumber AS "Lnr", name AS "Owner"))))
FROM Car, Person
WHERE owner = pid
GROUP BY color
```

```
<Color Name="black"><Car><Lnr>ABC123</Lnr><Owner>John
Higgins</Owner></Car><Car><Lnr>TYD226</Lnr><Owner>John
Higgins</Owner></Car><Car><Lnr>RSQ199</Lnr><Owner>Ronnie
O'Sullivan</Owner></Car></Color>

<Color Name="blue"><Car><Lnr>CCD457</Lnr><Owner>Stephen
Hendry</Owner></Car><Car><Lnr>ROO197</Lnr><Owner>Ken
Doherty</Owner></Car></Color>

<Color Name="green"><Car><Lnr>DKL998</Lnr><Owner>Matthew
Stevens</Owner></Car></Color>

<Color Name="red"><Car><Lnr>PTF357</Lnr><Owner>Steve
Davis</Owner></Car><Car><Lnr>WID387</Lnr><Owner>Matthew
Stevens</Owner></Car><Car><Lnr>DAVIS1</Lnr><Owner>Steve
Davis</Owner></Car></Color>
```



XMLAGG with GROUP BY

```
SELECT XMLELEMENT(NAME "CarColors", XMLAGG(colorxml))
FROM (SELECT XMLELEMENT(NAME "Color",
    XMLATTRIBUTES(color AS "Name"),
    XMLAGG(XMLELEMENT(NAME "Car",
        XMLFOREST(licencenumber AS "Lnr",
            name AS "Owner")))) AS colorxml
FROM Car, Person
WHERE owner = pid
GROUP BY color) AS innertable
```

```
<CarColors><Color Name="black"><Car><Lnr>ABC123</Lnr><Owner>John
Higgins</Owner></Car><Car><Lnr>TYD226</Lnr><Owner>John
Higgins</Owner></Car><Car><Lnr>RSQ199</Lnr><Owner>Ronnie
O'Sullivan</Owner></Car></Color><Color
Name="blue"><Car><Lnr>CCD457</Lnr><Owner>Stephen
Hendry</Owner></Car><Car><Lnr>ROO197</Lnr><Owner>Ken
Doherty</Owner></Car></Color><Color
Name="green"><Car><Lnr>DKL998</Lnr><Owner>Matthew
Stevens</Owner></Car></Color><Color
Name="red"><Car><Lnr>PTF357</Lnr><Owner>Steve
Davis</Owner></Car><Car><Lnr>WID387</Lnr><Owner>Matthew
Stevens</Owner></Car><Car><Lnr>DAVIS1</Lnr><Owner>Steve
Davis</Owner></Car></Color></CarColors>
```

- **Creates an XML comment**

```
SELECT XMLCOMMENT('Nice weather...')
FROM Person
```

[illegible]

XMLCOMMENT dynamic content

```
SELECT XMLELEMENT(NAME "People",  
    XMLAGG(XMLCONCAT(XMLCOMMENT('Here comes  
an element for the person with pid ' CONCAT pid),  
        XMLFOREST(name AS "Person"))))  
FROM Person
```

```
<People><!--Here comes an element for the person with pid 1--  
><Person>John Higgins</Person><!--Here comes an element for the  
person with pid 2--><Person>Stephen Hendry</Person><!--Here comes  
an element for the person with pid 3--><Person>Matthew  
Stevens</Person><!--Here comes an element for the person with pid 4--  
><Person>Ronnie O'Sullivan</Person><!--Here comes an element for  
the person with pid 5--><Person>Ken Doherty</Person><!--Here comes  
an element for the person with pid 6--><Person>Steve  
Davis</Person><!--Here comes an element for the person with pid 7--  
><Person>Paul Hunter</Person><!--Here comes an element for the  
person with pid 8--><Person>Neil Robertson</Person></People>
```

The previous result indented

```
<People>  
  <!--Here comes an element for the person with pid 1-->  
  <Person>John Higgins</Person>  
  <!--Here comes an element for the person with pid 2-->  
  <Person>Stephen Hendry</Person>  
  <!--Here comes an element for the person with pid 3-->  
  <Person>Matthew Stevens</Person>  
  <!--Here comes an element for the person with pid 4-->  
  <Person>Ronnie O'Sullivan</Person>  
  <!--Here comes an element for the person with pid 5-->  
  <Person>Ken Doherty</Person>  
  <!--Here comes an element for the person with pid 6-->  
  <Person>Steve Davis</Person>  
  <!--Here comes an element for the person with pid 7-->  
  <Person>Paul Hunter</Person>  
  <!--Here comes an element for the person with pid 8-->  
  <Person>Neil Robertson</Person>  
</People>
```

- ```
SELECT XMLPI(NAME "Congrats", 'to="everybody"')
FROM Person
```

[illegible]

- ```
SELECT XMLPI(NAME "Congrats", 'ladies and gentlemen')
FROM Person
```

[illegible]

XMLPI

- Dynamic values

```
SELECT XMLPI(NAME "Who is", name)
FROM Person
```

```
<?Who is John Higgins?>
<?Who is Stephen Hendry?>
<?Who is Matthew Stevens?>
<?Who is Ronnie O'Sullivan?>
<?Who is Ken Doherty?>
<?Who is Steve Davis?>
<?Who is Paul Hunter?>
<?Who is Neil Robertson?>
```

XMLNAMESPACES

- Creates namespaces inside and element
 - Must be specified inside XMLELEMENT before XMLATTRIBUTES

```
SELECT XMLELEMENT(NAME "sdxml:Data",
  XMLNAMESPACES('http://ns.dsv.su.se/SDXML' AS "sdxml"),
  XMLFOREST(COUNT(*) AS "sdxml:People",
    (SELECT COUNT(*) FROM Car) AS "sdxml:Cars"))
FROM Person
```

```
<sdxml:Data xmlns:sdxml="http://ns.dsv.su.se/SDXML">
  <sdxml:People>8</sdxml:People>
  <sdxml:Cars>9</sdxml:Cars>
</sdxml:Data>
```



XMLNAMESPACES DEFAULT

```
SELECT XMLELEMENT(NAME "Data",  
    XMLNAMESPACES(DEFAULT 'http://ns.dsv.su.se/SDXML'),  
    XMLFOREST(COUNT(*) AS "People",  
        (SELECT COUNT(*) FROM Car) AS "Cars"))
```

FROM Person

```
<Data xmlns="http://ns.dsv.su.se/SDXML">  
    <People>8</People>  
    <Cars>9</Cars>  
</Data>
```



XMLPARSE

- Converts a serialized XML value to XML
 - DOCUMENT or CONTENT

```
XMLPARSE(DOCUMENT '<root a="1" />')
```

The returned type is XML

```
'<root a="1" />'
```

This is STRING

XMLSERIALIZE

- **Serializes an XML value according to the specified data type**
 - DOCUMENT or CONTENT (default)
 - Can generate the XML declaration

XMLSERIALIZE(xmlvalue AS CLOB)

**XMLSERIALIZE(xmlvalue AS CLOB
INCLUDING XMLDECLARATION)**

XMLCAST

- **Converts values from and to XML**

XMLCAST(value AS data type)

XMLVALIDATE

- **Validates an XML value according to a schema**
 - The schema can be specified as a parameter
 - The schema may be part of the XML value
- **Returns the validated XML value**
 - Or an error

XMLVALIDATE(xmlvalue ACCORDING TO xmlschema)

XMLVALIDATE(xmlvalue)

XMLQUERY

- **Executes an XQuery statement**
 - Returns XML

```
SELECT XMLQUERY('for $a in (1, 2, 3) return element Number {$a}')  
FROM Person  
WHERE pid = 1
```

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



XMLQUERY

– Can pass column values as parameters

```
SELECT name, XMLQUERY('for $a in distinct-values($e//@employer)
                        return element Employer {$a}'
                        PASSING employments AS "e")
FROM Person
```

John Higgins	<Employer>ABB</Employer><Employer>UPC</Employer>
Stephen Hendry	<Employer>ABB</Employer><Employer>UPC</Employer>
Matthew Stevens	<Employer>UPC</Employer>
Ronnie O'Sullivan	<Employer>LKP</Employer><Employer>STG</Employer>
Ken Doherty	<Employer>LKP</Employer><Employer>ABB</Employer> <Employer>FFD</Employer><Employer>STG</Employer>
Steve Davis	<Employer>ABB</Employer><Employer>LKP</Employer> <Employer>FFD</Employer><Employer>XAB</Employer>
Paul Hunter	<Employer>FFD</Employer><Employer>LKP</Employer>
Neil Robertson	<Employer>UPC</Employer><Employer>ABB</Employer>



XMLQUERY

```
SELECT name, XMLQUERY('count(distinct-values($e//@employer))'
                        PASSING employments AS "e") AS NumberOfEmployers
FROM Person
```

John Higgins	2
Stephen Hendry	2
Matthew Stevens	1
Ronnie O'Sullivan	2
Ken Doherty	4
Steve Davis	4
Paul Hunter	2
Neil Robertson	2

XMLTABLE

- Creates a table from the result of an XQuery statement

SELECT *

**FROM XMLTABLE('for \$a in (2, 5, 9)
return element Result {\$a}')**

<Result>2</Result>

<Result>5</Result>

<Result>9</Result>

XMLTABLE implicit join

SELECT name, t2.*

**FROM Person, XMLTABLE('for \$a in distinct-values(\$e/@employer)
return element Employer {\$a}'
PASSING employments AS "e") AS t2**

John Higgins	<Employer>ABB</Employer>
John Higgins	<Employer>UPC</Employer>
Stephen Hendry	<Employer>ABB</Employer>
Stephen Hendry	<Employer>UPC</Employer>
Matthew Stevens	<Employer>UPC</Employer>
Ronnie O'Sullivan	<Employer>LKP</Employer>
Ronnie O'Sullivan	<Employer>STG</Employer>
Ken Doherty	<Employer>LKP</Employer>
Ken Doherty	<Employer>ABB</Employer>
Ken Doherty	<Employer>FFD</Employer>
Ken Doherty	<Employer>STG</Employer>
Steve Davis	<Employer>ABB</Employer>
Steve Davis	<Employer>LKP</Employer>
Steve Davis	<Employer>FFD</Employer>
Steve Davis	<Employer>XAB</Employer>
Paul Hunter	<Employer>FFD</Employer>
Paul Hunter	<Employer>LKP</Employer>
Neil Robertson	<Employer>UPC</Employer>
Neil Robertson	<Employer>ABB</Employer>



XMLTABLE columns

```
SELECT name, t2.*
FROM Person, XMLTABLE('$e//employment'
    PASSING employments AS "e"
    COLUMNS employer VARCHAR(10) PATH '@employer',
    startdate DATE PATH '@startdate',
    enddate DATE PATH '@enddate') AS t2
```

NAME	EMPLOYER	STARTDATE	ENDDATE
John Higgins	ABB	2001-08-20	2009-02-28
John Higgins	UPC	2009-04-15	NULL
Stephen Hendry	ABB	2002-08-20	2003-06-30
Stephen Hendry	UPC	2003-08-01	NULL
Stephen Hendry	ABB	2006-11-01	NULL
Matthew Stevens	UPC	2003-01-10	NULL
Ronnie O'Sullivan	LKP	2002-03-10	2010-05-22
Ronnie O'Sullivan	STG	2010-08-15	NULL
Ken Doherty	LKP	2002-02-12	2003-05-11
Ken Doherty	ABB	2003-05-12	2003-12-02
Ken Doherty	LKP	2003-12-06	2005-02-17
Ken Doherty	FFD	2005-02-18	2008-05-16
Ken Doherty	STG	2008-06-02	NULL
Steve Davis	ABB	2001-01-05	2005-12-31
Steve Davis	LKP	2006-01-15	2009-01-22
Steve Davis	FFD	2009-02-01	NULL
Steve Davis	XAB	2009-02-01	NULL
Paul Hunter	FFD	2004-01-10	2008-09-29
Paul Hunter	LKP	2008-10-01	2010-11-20
Neil Robertson	UPC	2006-02-03	2008-10-30
Neil Robertson	ABB	2008-11-20	NULL



XMLTABLE

```
SELECT name AS FullName, COUNT(*) AS NumberOfJobsNow
FROM Person, XMLTABLE('$e//employment[not(@enddate)]'
    PASSING employments AS "e" ) AS t2
GROUP BY name
```

FullName	NumberOfJobsNow
John Higgins	1
Ken Doherty	1
Matthew Stevens	1
Neil Robertson	1
Ronnie O'Sullivan	1
Stephen Hendry	2
Steve Davis	2

```
SELECT t2.*
FROM Person, XMLTABLE('$e//employment[not(@enddate)]'
    PASSING employments AS "e" ) AS t2
WHERE name = 'Stephen Hendry'
```

```
<employment startdate="2003-08-01" employer="UPC"/>
<employment startdate="2006-11-01" employer="ABB"/>
```



XMLTABLE

```
SELECT name, startdate
FROM Person, XMLTABLE('$e//employment[not(@enddate)]/@employer'
                      PASSING employments AS "e"
                      COLUMNS employer VARCHAR(10) PATH '.',
                               startdate VARCHAR(10) PATH '../@startdate') AS t2
WHERE employer = 'UPC'
```

NAME	STARTDATE
John Higgins	2009-04-15
Stephen Hendry	2003-08-01
Matthew Stevens	2003-01-10



XMLTABLE, OUTER JOIN

```
SELECT name, startdate, enddate
FROM Person LEFT OUTER JOIN XMLTABLE('$e//employment[@employer = "ABB"]'
                                     PASSING employments AS "e"
                                     COLUMNS startdate DATE PATH '@startdate',
                                              enddate VARCHAR(10) PATH '@enddate') AS t2
ON 1=1
```

NAME	STARTDATE	ENDDATE
John Higgins	2001-08-20	2009-02-28
Stephen Hendry	2002-08-20	2003-06-30
Stephen Hendry	2006-11-01	NULL
Matthew Stevens	NULL	NULL
Ronnie O'Sullivan	NULL	NULL
Ken Doherty	2003-05-12	2003-12-02
Steve Davis	2001-01-05	2005-12-31
Paul Hunter	NULL	NULL
Neil Robertson	2008-11-20	NULL



XMLTABLE FOR ORDINALITY

- **FOR ORDINALITY**

- Special column that enumerates the rows of the result of each call to XMLTABLE

```
SELECT name, ordernr, employer
FROM Person, XMLTABLE('distinct-values($e//employment[@enddate]/@employer)'
                      PASSING employments AS "e"
                      COLUMNS ordernr FOR ORDINALITY,
                               employer VARCHAR(5) PATH 'string()') AS t2
```

John Higgins	1	ABB
Stephen Hendry	1	ABB
Ronnie O'Sullivan	1	LKP
Ken Doherty	1	LKP
Ken Doherty	2	ABB
Ken Doherty	3	FFD
Steve Davis	1	ABB
Steve Davis	2	LKP
Paul Hunter	1	FFD
Paul Hunter	2	LKP
Neil Robertson	1	UPC



XMLEXISTS

- Checks if the result of an XQuery statement is empty

```
SELECT name
FROM Person
WHERE
XMLEXISTS('$e//employment[@employer="ABB"]'
          PASSING employments AS "e")
```

John Higgins
Stephen Hendry
Ken Doherty
Steve Davis
Neil Robertson

DBMS Support

- **Full support according to SQL:2003 and partial support according to SQL:2006, SQL:2008 and SQL:2011**
 - IBM DB2 10, 11 (not really full support for XPath)
 - Oracle Database 11g R2, 12c, 18c, 19c, 21c, 23c (some limitations)
 - PostgreSQL 11-14 (limited support, only XPath, no XQuery)
 - OpenLink Virtuoso (limited support)
 - SAP (formerly Sybase) SQL Anywhere 17 (limited support)
- **Alternative ways**
 - Microsoft

SQL/XML and Java

- **Support for the XML data type**
 - `java.sql.SQLXML`
 - can be converted to other Java XML objects with
 - » `XMLStreamReader`
 - » `SAXParser`
 - » `Transformer` (for XSLT)
- **Similar facilities in other programming languages**

SQL/JSON

- **SQL:2016 and SQL:2023**
 - Data type JSON
 - JSON-related functions
- **Support (more or less)**
 - DB2
 - Oracle
 - SQL Server
 - MySQL
 - PostgreSQL
 - MariaDB

What to do next

- **Quiz about SQL/XML (Quiz 7)**
- **Lesson exercises (4)**
- **Seminar exercises (Assignment 3)**
 - Wait until after the lessons