



SDXML VT2024

Models and languages for semi-structured data and XML

Product-specific techniques Microsoft SQL Server

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

Corresponding reading
Product documentation
Compendium with introduction to Microsoft SQL Server
Section 13.3 of the course book



Microsoft SQL Server 2022

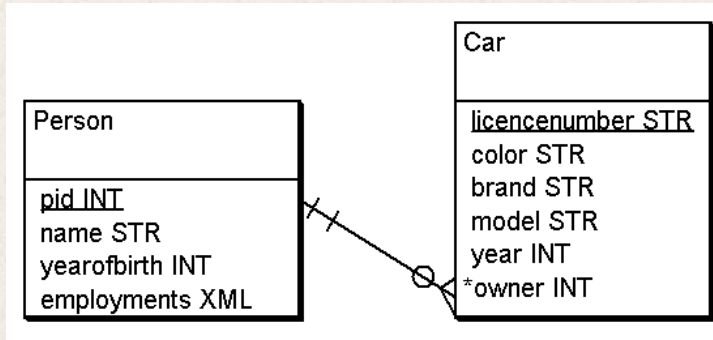
- **Data type**
 - Support for validation
 - According to the SQL standard
- **FOR XML**
- **OPENXML**
- **XML methods**
 - query (XQuery)
 - value
 - exist
 - modify (DML)
 - nodes
- **Limited support for XQuery**
 - only few functions
 - not all XPath axes
 - partial support for the let clause



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>



SQL Server - data type

- **XML**

- untyped - well-formed XML and XML fragments
- typed - associated to an XML Schema
- No support for DTD
 - » Supports inline DTD for things like default values

- **XML SCHEMA COLLECTION**

- register and name XML Schema
- used to define typed XML
- CREATE XML SCHEMA COLLECTION name AS 'the full schema'

- **Methods**

- query, value, exist, modify, nodes

- **Limited support for XPath and XQuery**

- Does not support all XPath axes
- Does not support all XPath/XQuery functions
- Does not fully support computed constructors and not in the let clause



SQL Server - typed XML

- **The schema must be an XML SCHEMA COLLECTION**

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

SQL Server - typed XML

- The schema must be an XML SCHEMA COLLECTION
 - default namespaces

```
CREATE XML SCHEMA COLLECTION employments_xsd AS
'<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ns="http://ns" targetNamespace="http://ns">
  <element name="root">
    <complexType>
      <sequence>
        <element name="employment" type="ns: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>'
```

SQL Server - typed XML

- A column can be created as typed XML based on an existing XML SCHEMA COLLECTION

```
CREATE TABLE Person (
  pid INTEGER NOT NULL IDENTITY (1,1) PRIMARY KEY,
  name VARCHAR (30) NOT NULL,
  yearofbirth INTEGER NOT NULL,
  employments XML(employments_xsd));
```

- The content on the column is validated on INSERT and on UPDATE

SQL Server & SQL/XML

- **No support for the publishing functions**
 - FOR XML clause in the SELECT statement instead
- **No support for XMLQUERY, XMLTABLE, XMLEXISTS**
 - XML methods instead (partially aslo OPENXML)
- **No support for XMLVALIDATE**
 - validation is automatic for typed XML
- **No support for XMLCAST**
 - The general function CAST is enough (and CONVERT)

SQL Server - OPENXML

- **Converts XML data to relational data**
- **Requires the use of procedures**
- **Can generate one row per node**
 - Default, the result is a so called "edge table"
- **Can generate one column per specified expression**
 - With the keyword WITH
- **Suitable for batch shredding of XML files**



SQL Server - FOR XML

- **Extra clause in SELECT statements**
- **Converts the result to XML according to different modes**
 - RAW
 - AUTO
 - EXPLICIT (should be avoided)
 - PATH
- **The result is always an XML value**
 - Either a fragment or well-formed XML with the keyword ROOT
 - Either serialized or of the XML data type with the keyword TYPE



SQL Server - FOR XML RAW

- **Every row becomes one element**
 - default element name: "row"
 - every column become one attribute

```
SELECT pid, name, yearofbirth  
FROM Person  
WHERE pid < 4  
FOR XML RAW
```

```
<row pid="1" name="John Higgins" yearofbirth="1975" />  
<row pid="2" name="Stephen Hendry" yearofbirth="1973" />  
<row pid="3" name="Matthew Stevens" yearofbirth="1982" />
```

SQL Server - FOR XML RAW

- **ROOT** specifies that a root element should be created
 - default element name: "root"
- **The element name can be specified**
 - root element name after ROOT
 - row element name after RAW

```
SELECT pid, name, yearofbirth  
FROM Person  
WHERE pid < 4  
FOR XML RAW ('Person'), ROOT ('People')
```

```
<People>  
  <Person pid="1" name="John Higgins" yearofbirth="1975" />  
  <Person pid="2" name="Stephen Hendry" yearofbirth="1973" />  
  <Person pid="3" name="Matthew Stevens" yearofbirth="1982" />  
</People>
```

SQL Server - FOR XML RAW

- **ELEMENTS** specifies that elements should be created instead of attributes (for each column)
 - The element names (or attribute names) are based on the SELECT clause

```
SELECT pid AS PersonID, name AS FullName,  
       yearofbirth AS YoB  
FROM Person  
WHERE pid = 2  
FOR XML RAW ('Person'), ROOT ('People'), ELEMENTS
```

```
<People>  
  <Person>  
    <PersonID>2</PersonID>  
    <FullName>Stephen Hendry</FullName>  
    <YoB>1973</YoB>  
  </Person>  
</People>
```



SQL Server - FOR XML AUTO

- **Elements are created automatically**
 - In simple situations, almost identical to RAW
 - default element names: table names or table aliases

```
SELECT pid, name, yearofbirth  
FROM Person  
WHERE pid < 4  
FOR XML AUTO
```

```
<Person pid="1" name="John Higgins" yearofbirth="1975" />  
<Person pid="2" name="Stephen Hendry" yearofbirth="1973" />  
<Person pid="3" name="Matthew Stevens" yearofbirth="1982" />
```



SQL Server - FOR XML AUTO

- **Automatic nesting**
 - One element per used table (in FROM and SELECT)
 - Nesting is based on the order of the columns in the SELECT clause
 - Subelement grouping is based on the ORDER BY clause

```
SELECT pid, Name, model, brand, yearofbirth YoB,  
        licencenumber, color AS Colour  
FROM Person, Car AS Vehicle  
WHERE pid < 3 AND owner = pid  
FOR XML AUTO
```

```
<Person pid="1" Name="John Higgins" YoB="1975">  
  <Vehicle model="Cherry" brand="NISSAN" licencenumber="ABC123" Colour="black" />  
</Person>  
<Person pid="2" Name="Stephen Hendry" YoB="1973">  
  <Vehicle model="Forza" brand="FIAT" licencenumber="CCD457" Colour="blue" />  
</Person>  
<Person pid="1" Name="John Higgins" YoB="1975">  
  <Vehicle model="Cherry" brand="NISSAN" licencenumber="TYD226" Colour="black" />  
</Person>
```

SQL Server - FOR XML AUTO

```
SELECT model, pid, Name, brand, yearofbirth YoB,  
        licencenumber, color AS Colour  
FROM Person, Car AS Vehicle  
WHERE pid < 3 AND owner = pid  
FOR XML AUTO
```

```
<Vehicle model="Cherry" brand="NISSAN" licencenumber="ABC123" Colour="black">  
  <Person pid="1" Name="John Higgins" YoB="1975" />  
</Vehicle>  
<Vehicle model="Forza" brand="FIAT" licencenumber="CCD457" Colour="blue">  
  <Person pid="2" Name="Stephen Hendry" YoB="1973" />  
</Vehicle>  
<Vehicle model="Cherry" brand="NISSAN" licencenumber="TYD226" Colour="black">  
  <Person pid="1" Name="John Higgins" YoB="1975" />  
</Vehicle>
```

SQL Server - FOR XML AUTO

- Subelement grouping is based on the ORDER BY clause

```
SELECT pid, Name, model, brand, yearofbirth YoB,  
        licencenumber, color AS Colour  
FROM Person, Car AS Vehicle  
WHERE pid < 3 AND owner = pid  
ORDER BY pid  
FOR XML AUTO
```

```
<Person pid="1" Name="John Higgins" YoB="1975">  
  <Vehicle model="Cherry" brand="NISSAN" licencenumber="ABC123" Colour="black" />  
  <Vehicle model="Cherry" brand="NISSAN" licencenumber="TYD226" Colour="black" />  
</Person>  
<Person pid="2" Name="Stephen Hendry" YoB="1973">  
  <Vehicle model="Forza" brand="FIAT" licencenumber="CCD457" Colour="blue" />  
</Person>
```



SQL Server - FOR XML AUTO

- Nest in order force the desired result

```
SELECT Name, LicenceNr, Brand, Owner
FROM (SELECT DISTINCT color AS name FROM Car) AS Colour,
     (SELECT licencenumber AS licencenr, color, brand, name AS owner
      FROM Car, Person WHERE owner = pid) AS Vehicle
WHERE name = color
ORDER BY name
FOR XML AUTO
```

```
<Colour Name="black">
  <Vehicle LicenceNr="ABC123" Brand="NISSAN" Owner="John Higgins" />
  <Vehicle LicenceNr="RSQ199" Brand="NISSAN" Owner="Ronnie O'Sullivan" />
  <Vehicle LicenceNr="TYD226" Brand="NISSAN" Owner="John Higgins" />
</Colour>
<Colour Name="blue">
  <Vehicle LicenceNr="CCD457" Brand="FIAT" Owner="Stephen Hendry" />
  <Vehicle LicenceNr="ROO197" Brand="SAAB" Owner="Ken Doherty" />
</Colour>
<Colour Name="green">
  <Vehicle LicenceNr="DKL998" Brand="SAAB" Owner="Matthew Stevens" />
</Colour>
<Colour Name="red">
  <Vehicle LicenceNr="DAVIS1" Brand="VOLVO" Owner="Steve Davis" />
  <Vehicle LicenceNr="PTF357" Brand="VOLVO" Owner="Steve Davis" />
  <Vehicle LicenceNr="WID387" Brand="FIAT" Owner="Matthew Stevens" />
</Colour>
```



SQL Server - FOR XML AUTO

- Supports ROOT, TYPE and ELEMENTS

```
SELECT pid, Name, Brand, Model, yearofbirth AS YoB, LicenceNumber, color AS Colour
FROM Person, Car AS Vehicle
WHERE pid < 2 AND owner = pid
ORDER BY pid
FOR XML AUTO, ROOT, TYPE, ELEMENTS
```

```
<root>
  <Person>
    <pid>1</pid>
    <Name>John Higgins</Name>
    <YoB>1975</YoB>
    <Vehicle>
      <Brand>NISSAN</Brand>
      <Model>Cherry</Model>
      <LicenceNumber>ABC123</LicenceNumber>
      <Colour>black</Colour>
    </Vehicle>
    <Vehicle>
      <Brand>NISSAN</Brand>
      <Model>Cherry</Model>
      <LicenceNumber>TYD226</LicenceNumber>
      <Colour>black</Colour>
    </Vehicle>
  </Person>
</root>
```

SQL Server - FOR XML PATH

- **More flexible than RAW and AUTO**
 - Possible to mix elements and attributes
 - Manual nesting
- **Column aliases in the SELECT clause are interpreted as XPath expressions**
 - Default: similar to RAW, ELEMENTS

```
SELECT licencenumber AS LicenceNr, Color, Brand, name AS Owner
FROM Car, Person
WHERE owner = pid
FOR XML PATH
```

```
SELECT licencenumber AS LicenceNr, Color, Brand, name AS Owner
FROM Car, Person
WHERE owner = pid
FOR XML RAW, ELEMENTS
```

SQL Server - FOR XML PATH

```
SELECT licencenumber AS "@LicenceNr", color AS "@Colour",
      brand AS "@Brand", name AS Owner
FROM Car, Person
WHERE owner = pid AND owner < 3
FOR XML PATH ('Vehicle')
```

```
<Vehicle LicenceNr="ABC123" Colour="black" Brand="NISSAN">
  <Owner>John Higgins</Owner>
</Vehicle>
<Vehicle LicenceNr="CCD457" Colour="blue" Brand="FIAT">
  <Owner>Stephen Hendry</Owner>
</Vehicle>
<Vehicle LicenceNr="TYD226" Colour="black" Brand="NISSAN">
  <Owner>John Higgins</Owner>
</Vehicle>
```



SQL Server - FOR XML PATH

```
SELECT licencenumber AS "@LicenceNr", color AS "@Colour",  
       brand AS "Type/@Brand", model AS "Type/@Model",  
       name AS "Owner/@Name", pid AS "Owner/@PID"  
FROM Car, Person  
WHERE owner = pid AND owner < 3  
FOR XML PATH ('Vehicle')
```

```
<Vehicle LicenceNr="ABC123" Colour="black">  
  <Type Brand="NISSAN" Model="Cherry" />  
  <Owner Name="John Higgins" PID="1" />  
</Vehicle>  
<Vehicle LicenceNr="CCD457" Colour="blue">  
  <Type Brand="FIAT" Model="Forza" />  
  <Owner Name="Stephen Hendry" PID="2" />  
</Vehicle>  
<Vehicle LicenceNr="TYD226" Colour="black">  
  <Type Brand="NISSAN" Model="Cherry" />  
  <Owner Name="John Higgins" PID="1" />  
</Vehicle>
```



SQL Server - FOR XML PATH

- Use * to create a text node (without extra element)

```
SELECT name "!", ' owns ' "!",  
       (SELECT COUNT(*) FROM Car WHERE owner = pid) "!", ' cars' "!"  
FROM Person  
FOR XML PATH ('Statement'), ROOT ('Info')
```

```
<Info>  
  <Statement>John Higgins owns 2 cars</Statement>  
  <Statement>Stephen Hendry owns 1 cars</Statement>  
  <Statement>Matthew Stevens owns 2 cars</Statement>  
  <Statement>Ronnie O'Sullivan owns 1 cars</Statement>  
  <Statement>Ken Doherty owns 1 cars</Statement>  
  <Statement>Steve Davis owns 2 cars</Statement>  
  <Statement>Paul Hunter owns 0 cars</Statement>  
  <Statement>Neil Robertson owns 0 cars</Statement>  
</Info>
```



SQL Server - FOR XML PATH

- **Grouping requires nesting**
 - ORDER BY does not have the same effect as with AUTO
 - Use TYPE in the nested statement to return XML

```
SELECT color AS "@name",  
      (SELECT licencenumber AS "@LicenceNr", brand AS "@Brand",  
        model AS "@Model", name AS "Owner/@Name"  
        FROM Car, Person  
        WHERE owner = pid AND color = ct.color  
        FOR XML PATH ('Vehicle'), TYPE) AS "*"   
FROM (SELECT DISTINCT color FROM Car WHERE color <> 'black') AS ct  
FOR XML PATH ('Colour'), ROOT ('CarsByColour')
```



SQL Server - FOR XML PATH

```
<CarsByColour>  
  <Colour name="blue">  
    <Vehicle LicenceNr="CCD457" Brand="FIAT" Model="Forza">  
      <Owner Name="Stephen Hendry" />  
    </Vehicle>  
    <Vehicle LicenceNr="ROO197" Brand="SAAB" Model="900i">  
      <Owner Name="Ken Doherty" />  
    </Vehicle>  
  </Colour>  
  <Colour name="green">  
    <Vehicle LicenceNr="DKL998" Brand="SAAB" Model="9000C">  
      <Owner Name="Matthew Stevens" />  
    </Vehicle>  
  </Colour>  
  <Colour name="red">  
    <Vehicle LicenceNr="DAVIS1" Brand="VOLVO" Model="V90">  
      <Owner Name="Steve Davis" />  
    </Vehicle>  
    <Vehicle LicenceNr="PTF357" Brand="VOLVO" Model="V70">  
      <Owner Name="Steve Davis" />  
    </Vehicle>  
    <Vehicle LicenceNr="WID387" Brand="FIAT" Model="Nova">  
      <Owner Name="Matthew Stevens" />  
    </Vehicle>  
  </Colour>  
</CarsByColour>
```



SQL Server - FOR XML

- **NULL causes the node to not get created**
 - With the keyword **XSI:nil** (after **ELEMENTS**) a node will be created with the attribute **xsi:nil="true"**
- **The keywords XMLDATA and XMLSCHEMA generate a schema for the result**
 - Quite limited
- **Use " as element name to eliminate a level**
- **Use WITH XMLNAMESPACES (before the SELECT clause) to define namespaces**
 - **WITH XMLNAMESPACES ('nsuri' AS nsalias)**
 - **WITH XMLNAMESPACES (DEFAULT 'nsuri')**



SQL Server - value()

- **Method that evaluates an XQuery statement and returns a value**
 - **XML-object.value(xquery, data type)**
 - **xquery must always give one value/node. Use [1] if the result could theoretically be a sequence**
 - » **//employment[1]/@employer**
must be specified as
(//employment[1]/@employer)[1]
or
(//employment)[1]/@employer

```
SELECT name, employments.value('(//employment[1]/@employer)[1]', 'varchar(10)')
FROM Person
WHERE pid < 4
```

John Higgins	ABB
Stephen Hendry	ABB
Matthew Stevens	UPC

SQL Server - query()

- Evaluates an XQuery statement and returns XML

```
SELECT name, employments.query('//employment[1]')
FROM Person
WHERE pid < 4
```

John Higgins	<employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB" />
Stephen Hendry	<employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB" />
Matthew Stevens	<employment startdate="2003-01-10" employer="UPC" />

```
SELECT name, employments.query('for $x in //employment[@employer="ABB"]
                                return element X {$x/@startdate}')
FROM Person
WHERE pid < 4
```

John Higgins	<X startdate="2001-08-20" />
Stephen Hendry	<X startdate="2002-08-20" /><X startdate="2006-11-01" />
Matthew Stevens	

SQL Server - value() vs query()

```
SELECT name, employments.query('count(//employment)'),
       employments.query('count(distinct-values(//@employer))')
FROM Person
WHERE pid < 4
```

```
SELECT name, employments.value('count(//employment)', 'int'),
       employments.value('count(distinct-values(//@employer))', 'int')
FROM Person
WHERE pid < 4
```

John Higgins	2	2
Stephen Hendry	3	2
Matthew Stevens	1	1

The result of the method query is XML



SQL Server - nodes()

- **Converts a sequence of nodes to a table**
 - One column
 - Every row corresponds to one node
 - The result is a relative node position in the original XML object
 - Can be used together with the keywords CROSS (or OUTER) APPLY (in order to use a column in the FROM clause)
 - Corresponds to SQL/XML:s XMLTABLE

```
SELECT name, c.query('.'), c.value('@employer', 'varchar(10)')
FROM Person CROSS APPLY employments.nodes('//employment') AS X(c)
WHERE pid < 4
```

John Higgins	<employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB" />	ABB
John Higgins	<employment startdate="2009-04-15" employer="UPC" />	UPC
Stephen Hendry	<employment startdate="2002-08-20" enddate="2003-06-30" employer="ABB" />	ABB
Stephen Hendry	<employment startdate="2003-08-01" employer="UPC" />	UPC
Stephen Hendry	<employment startdate="2006-11-01" employer="ABB" />	ABB
Matthew Stevens	<employment startdate="2003-01-10" employer="UPC" />	UPC



SQL Server - nodes() & CROSS APPLY

```
SELECT name, c.query('.')
FROM Person CROSS APPLY
      employments.nodes('//employment[not(@enddate)]') AS X(c)
```

John Higgins	<employment startdate="2009-04-15" employer="UPC" />
Stephen Hendry	<employment startdate="2003-08-01" employer="UPC" />
Stephen Hendry	<employment startdate="2006-11-01" employer="ABB" />
Matthew Stevens	<employment startdate="2003-01-10" employer="UPC" />
Ronnie O'Sullivan	<employment startdate="2010-08-15" employer="STG" />
Ken Doherty	<employment startdate="2008-06-02" employer="STG" />
Steve Davis	<employment startdate="2009-02-01" employer="FFD" />
Steve Davis	<employment startdate="2009-02-01" employer="XAB" />
Neil Robertson	<employment startdate="2008-11-20" employer="ABB" />



SQL Server - nodes() & OUTER APPLY

```
SELECT name, c.query('.')  
FROM Person OUTER APPLY  
      employments.nodes('//employment[not(@enddate)]') AS X(c)
```

John Higgins	<employment startdate="2009-04-15" employer="UPC" />
Stephen Hendry	<employment startdate="2003-08-01" employer="UPC" />
Stephen Hendry	<employment startdate="2006-11-01" employer="ABB" />
Matthew Stevens	<employment startdate="2003-01-10" employer="UPC" />
Ronnie O'Sullivan	<employment startdate="2010-08-15" employer="STG" />
Ken Doherty	<employment startdate="2008-06-02" employer="STG" />
Steve Davis	<employment startdate="2009-02-01" employer="FFD" />
Steve Davis	<employment startdate="2009-02-01" employer="XAB" />
Paul Hunter	NULL
Neil Robertson	<employment startdate="2008-11-20" employer="ABB" />



SQL Server - exist()

- **Evaluates an XQuery statement and returns**
 - 1 if the result was not empty
 - 0 if the result was empty
 - NULL if the XML object was NULL

```
SELECT name  
FROM Person  
WHERE employments.exist('//employment[@employer="ABB"]') = 1
```

John Higgins
Stephen Hendry
Ken Doherty
Steve Davis
Neil Robertson

SQL Server - modify()

- **Changes an XML object**
 - Does not return anything
- **Supported DML operations**
 - insert
 - delete
 - replace value of
- **Automatic validation for typed XML**
- **Used only in the SET clause of UPDATE statements**
 - or with an object in a variable

SQL Server - modify() - insert

- **Add new nodes**
 - as first
 - as last
 - into
 - after
 - before
- **The nodes can be created, deserialized or retrieved from a variable.**

UPDATE Person

SET employments.modify('insert <employment startdate="2011-11-12" employer="KFC"/> as last into /root[1]')

WHERE pid = 6



SQL Server - modify() - insert

UPDATE Person

**SET employments.modify('insert <!-- new employment added --> before
(/root/employment[@employer="KFC"][@startdate="2011-11-12"])[1]')**
WHERE pid = 6

UPDATE Person

**SET employments.modify('insert attribute enddate {"2012-04-12"} into
(/root/employment[@employer="KFC"][@startdate="2011-11-12"])[1]')**
WHERE pid = 6

if the attribute startdate is xs:date (in the XML Schema for typed XML):

[@startdate= "2011-11-12" cast as xs:date?]

or

[@startdate= xs:date("2011-11-12")]



SQL Server - modify() - delete

- **Removes nodes that match an XPath expression**
 - The root node may not be removed
 - Wildcards are allowed

UPDATE Person

**SET employments.modify('delete
/root/employment[@employer="KFC"]')**
WHERE pid = 6

UPDATE Person

SET employments.modify('delete //comment()')
WHERE pid = 6

SQL Server - modify() - replace

- Replaces the value of one node
 - replace value of xpath-expression with new-value

```
UPDATE Person
SET employments.modify('
replace value of (/root/employment[@employer="ABB"]/@startdate)[1]
with "2001-01-04"')
WHERE pid = 6
```

For typed XML:

```
UPDATE person
SET employments.modify('
replace value of (/root/employment[@employer="ABB"]/@startdate)[1]
with xs:date("2001-01-04")')
WHERE pid = 6
```

SQL Server - XQuery

- The methods query, nodes, etc. require an XML object
- In order to execute XQuery independently of tables, use an XML object in a variable:

```
DECLARE @x xml
SET @x="
SELECT @x.query('for $a in (2,3,4), $b in (5,6,7)
where $a+$b = 9
return element X {$a*$b}')
```

```
<X>14</X><X>18</X><X>20</X>
```

Or:

```
SELECT CONVERT(XML, "").query(...)
```



SQL Server - sql:column

- SQL Server-specific XQuery function
 - Makes a column value available to XQuery

```
SELECT employments.query('element Person {attribute  
age {2024-sql:column("yearofbirth")}, attribute name  
{sql:column("name")}}')
```

FROM Person

```
<Person age="49" name="John Higgins" />  
<Person age="51" name="Stephen Hendry" />  
<Person age="42" name="Matthew Stevens" />  
<Person age="44" name="Ronnie O'Sullivan" />  
<Person age="50" name="Ken Doherty" />  
<Person age="64" name="Steve Davis" />  
<Person age="41" name="Paul Hunter" />  
<Person age="42" name="Neil Robertson" />
```



SQL Server - sql:column

```
SELECT name, employments.query(  
  for $e in //employment  
  let $y := sql:column("p.yearofbirth"),  
    $sy := substring($e/@startdate, 1, 4) cast as xs:integer?  
  return element Job {attribute age {$sy - $y}, $e/@employer}  
)
```

FROM Person p
WHERE pid = 2

Stephen Hendry	<Job age="29" employer="ABB" />
	<Job age="30" employer="UPC" />
	<Job age="33" employer="ABB" />

SQL Server does not support many XPath/XQuery functions, so we are forced to use substring and cast (or number after substring) instead of year-from-date

SQL Server - sql:variable

- SQL Server-specific XQuery function
 - Makes an SQL variable available in XQuery

```
DECLARE @x XML
SET @x = "
DECLARE @v VARCHAR(10)
SET @v = 'Course'
SELECT @x.query('for $x in (1,2,3)
                return element X {sql:variable("@v"), $x}')
```

<X>Course 1</X><X>Course 2</X><X>Course 3</X>

SQL Server - combinations

- Almost everything can be combined.

```
SELECT name AS "@Namn", employments.query('
    for $e in //employment
    let $y := sql:column("p.yearofbirth"),
    $sy := substring($e/@startdate, 1, 4) cast as xs:integer?
    return element Job {attribute Age {$sy - $y},
    attribute Company {$e/@employer}}')
```

```
FROM Person p
WHERE pid < 3
FOR XML PATH ('Person'), ROOT ('People')
```

```
<People>
  <Person Namn="John Higgins">
    <Job Age="26" Company="ABB" />
    <Job Age="34" Company="UPC" />
  </Person>
  <Person Namn="Stephen Hendry">
    <Job Age="29" Company="ABB" />
    <Job Age="30" Company="UPC" />
    <Job Age="33" Company="ABB" />
  </Person>
</People>
```

SQL Server - combinations

```
SELECT (SELECT name AS "@Name", COUNT(*) AS "@NrOfCars",  
        jobcount AS "@NrOfJobs"  
FROM (SELECT pid, name,  
        employments.value('count(//employment)', 'integer') as jobcount  
FROM Person) AS p, Car  
WHERE pid = owner  
GROUP BY name, jobcount  
FOR XML PATH ('Person'),  
        ROOT ('People'),  
        TYPE).query('element Root {//Person[@NrOfCars = 1]}')
```

<Root>

<Person Name="Ronnie O'Sullivan" NrOfCars="1" NrOfJobs="2" />

<Person Name="Stephen Hendry" NrOfCars="1" NrOfJobs="3" />

<Person Name="Ken Doherty" NrOfCars="1" NrOfJobs="5" />

</Root>

SQL Server & JSON

- **FOR JSON**
 - Similar to FOR XML
 - PATH and AUTO
- **Functions**
 - JSON_VALUE, JSON_QUERY, JSON_MODIFY, JSON_PATH_EXISTS
- **OPENJSON**
 - Similar to OPENXML

Summary

- **SQL Server does not follow the SQL standard**
 - No functions
 - Maybe in the future (all SQL/XML keywords are reserved words)
- **Limited support for XQuery 1 and XPath**
- **Still possible to achieve approximately the same functionality**

What to do next

- **Quiz about SQL Server & SQLXML (Quiz 8)**
- **Introduction to SQL Server & XML (compendium)**
 - Introduction to SQL Server and SQL Server Management Studio
 - Examples
- **Assignment 8 (SQL Server & XML)**