IV1023 vt2021
Avancerad Datahantering med XML

Proprietära lösningar

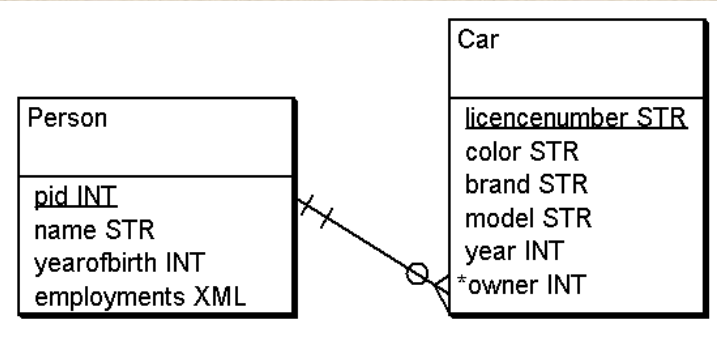
nikos dimitrakas
nikosd@kth.se
08-161295
Rum 2423

Läsanvisningar
Respektive produkts dokumentation
Kompendierna med introduktion till respektive produkt
Kapitel 13.3 i kursboken

Leverantörer/Produkter

- **IBM**
  - **DB2 11.5**
- **Oracle**
  - **Oracle Database 19c**
- **Microsoft**
  - **SQL Server 2019**

Exempeldata



PERSON

pid	name	yearofbirth
1	John Higgins	1975
2	Steven Hendry	1973
3	Mathew 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

Exempeldata

- **Kolumnen employments enligt följande 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>
```

Exempeldata

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"/></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>

IBM DB2 11.5

- **Stöd för SQL/XML enligt SQL 2006 med vissa undantag**
  - **Inget stöd för XML-datatyper med kopplat schema**
  - **Inget fullt stöd för XPath, XQuery**
- **Extra tillägg**
  - **XMLGROUP, XMLROW, XSLTRANSFORM**
  - **XQUERY, extra XQuery-funktioner**
- **Stöd i tidigare versioner**
  - **Datatyper XMLVARCHAR, XMLCLOB, XMLFILE**
  - **DTD-validering**
  - **Composition och shredding enligt templates (DAD-filer)**
  - **Uppdatering av data inuti XML-strukturen**
    - » **update-funktion**
  - **Funktioner för XPath-användning**
    - » **extractInteger, extractDate, extractVarchar, extractCLOB, etc**
    - » **extractIntegers, extractDates, extractVarchars, etc.**

DB2 - datatyp

- XML
  - Inget stöd för explicit associering av XML Schema, DTD
  - Accepterar well-formed XML och XML-fragment
- Validering
  - Stöd för XML Schema
  - Ej stöd för DTD
  - Funktionen XMLVALIDATE
    - » explicit schema
    - » implicit schema
  - Registrering av XML Schema
- VALIDATED constraint
  - Testar om ett XML-värde är validerat
    - » generellt
    - » enligt specifik XML Schema

DB2 - XML Schema Repository

- Stöd för XML Schema och DTD
- REGISTER XMLSCHEMA
- ADD XMLSCHEMA
- COMPLETE XMLSCHEMA
- UPDATE XMLSCHEMA
- REGISTER XSROBJECT

DB2 - SQL/XML

- Stödjer följande funktioner
  - XMLELEMENT
  - XMLATTRIBUTES
  - XMLFOREST
  - XMLCONCAT
  - XMLCOMMENT
  - XMLPI
  - XMLAGG
  - XMLTEXT
  - XMLDOCUMENT
  - XMLPARSE
  - XMLVALIDATE
  - XMLNAMESPACES
  - XMLSERIALIZE
  - XMLTABLE
  - XMLQUERY (endast SEQUENCE)
  - XMLCAST (klassificerad som expression, inte som funktion)
  - XMLEXISTS (klassificerad som predikat, inte som funktion)

DB2 - XMLROW

- Returnerar ett XML-dokument per rad

```
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 - elementnamn

- Tillåter konfigurering av elementnamn

```
SELECT XMLROW(name AS "Namn",
                    yearofbirth AS "Födelseår",
                    pid AS ID
                    OPTION ROW "Någon")
FROM person
```

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

```
SELECT XMLELEMENT(NAME "Någon", XMLFOREST(name AS "Namn",
                                  yearofbirth AS "Födelseår", pid AS ID))
FROM person
```


DB2 - XMLROW - attribut

- Möjligt att välja attribut istället för element

```
SELECT XMLROW(name AS "Namn",
                    yearofbirth AS "Födelseår",
                    pid AS ID
                    OPTION ROW "Någon" AS ATTRIBUTES)
FROM person
```

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

```
SELECT XMLELEMENT(NAME "Någon", XMLATTRIBUTES(name AS "Namn",
                                  yearofbirth AS "Födelseår", pid AS ID))
FROM person
```

DB2 - XMLGROUP

- Returnerar många rader som ett XML-dokument
  - Aggregeringsfunktion

```
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 - elementnamn

- Tillåter konfigurering av elementnamn

```
SELECT XMLGROUP(name AS "Namn", yearofbirth AS "Födelseår",
                pid AS ID OPTION ROOT "Alla" ROW "Någon")
FROM person
<Alla>
  <Någon>
    <Namn>John Higgins</Namn>
    <Födelseår>1975</Födelseår>
    <ID>1</ID>
  </Någon>
  <Någon>
    <Namn>Stephen Hendry</Namn>
    <Födelseår>1973</Födelseår>
    <ID>2</ID>
  </Någon>
  <Någon>
    <Namn>Matthew Stevens</Namn>
    <Födelseår>1982</Födelseår>
    <ID>3</ID>
  </Någon>
  …
</Alla>
```

DB2 - XMLGROUP - attribut

- Möjligt att välja attribut istället för element

```
SELECT XMLGROUP(name AS "Namn", yearofbirth AS "Födelseår",
                pid AS ID OPTION AS ATTRIBUTES
                ROOT "Alla" ROW "Någon")
FROM person
```

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

```
SELECT XMLELEMENT(NAME "Alla",
          XMLAGG(XMLROW(name AS "Namn", yearofbirth AS "Födelseår",
          pid AS ID OPTION ROW "Någon" AS ATTRIBUTES)))
FROM person
```


DB2 - XSLTRANSFORM

- Transformerar ett XML-dokument enligt ett XSLT-dokument (XSLT 1.0)
  - XSLTRANSFORM(XML-värde USING XSLT-värde)

```
<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 = 'Steve Davis'

<Employers>
    <Employer>ABB</Employer>
    <Employer>LKP</Employer>
    <Employer>FFD</Employer>
</Employers>
```


DB2 - XSLTRANSFORM

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

INSERT INTO xslt 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 xslt.value)
FROM person, xslt
WHERE person.name = 'Steve Davis' AND xslt.name = 'employers'
```

DB2 - XMLCAST

- Konverterar mellan XML och andra datatyper
- Kräver ett värde (eller en nod)

```
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
```

Obs! Beter sig konstigt med sekvenser som ska konverteras till annat än XML.

DB2 - XQUERY

- Stöd för XQuery-uttryck
  - XQuery-uttryck efter nyckelordet XQUERY

```
xquery
   for $a in (1,2,3)
   return element Nummer {$a}
```

Tre rader i resultatet:

```
   <Nummer>1</Nummer>
   <Nummer>2</Nummer>
   <Nummer>3</Nummer>
```

```
   xquery
   element Resultat  {for $a in (1,2,3)
                      return element Nummer {$a}}

   <Resultat>
    <Nummer>1</Nummer><Nummer>2</Nummer><Nummer>3</Nummer>
   </Resultat>
```

DB2 - XQUERY - sqlquery

- XQuery-funktion för SQL i XQuery
  - db2-fn:sqlquery
    - » stöd för parametrar
  - resultatet är en sekvens i XQuery-kontext
  - SELECT-satsen måste returnera en kolumn av XML-typ

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


  <Företag>ABB</Företag>
  <Företag>UPC</Företag>
  <Företag>LKP</Företag>
  <Företag>STG</Företag>
  <Företag>FFD</Företag>
```


DB2 - XQUERY - sqlquery

- Parametrar

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


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

DB2 - XQUERY - sqlquery

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

```
<Folk>
        <Person>Stephen Hendry</Person>
        <Person>Ken Doherty</Person>
        <Person>Steve Davis</Person>
</Folk>
```


DB2 - XQUERY - xmlcolumn

- XQuery-funktion för att hämta data från en XML-kolumn
  - db2-fn:xmlcolumn
  - Tabellnamn och kolumnnamn är case-sensitive

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

```
  <Företag>ABB</Företag>
  <Företag>UPC</Företag>
  <Företag>LKP</Företag>
  <Företag>STG</Företag>
  <Företag>FFD</Företag>
```

DB2 - XML DML

- XQuery Update Facility
  - några syntaxskillnader
- transform-uttryck
  - (transform-klausul)
  - copy-klausul
  - modify-klausul
    - » do delete
    - » do insert
    - » do rename
    - » do replace
  - return-klausul
  - nyckelord för do insert
    - » before
    - » after
    - » as first into
    - » as last into
    - » into
  - nyckelord för do replace
    - » (value of) … with …
  - nyckelord för do rename
    - » as

DB2 - XML DML

- Hela XML-värdet måste uppdateras

```
UPDATE Person
SET employments = XMLQUERY('transform-uttryck'
                           PASSING employments)
WHERE …
```

DB2 - transform - insert

```
xquery
transform
copy $x := <root><a>456</a><a>789</a></root>
modify do insert <a>123</a> as first into $x
return $x
```

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


IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

DB2 - transform - insert

```
xquery
transform
copy $x := <root><a>456</a><a>789</a></root>
modify do insert <a>123</a> 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 $nye := $e
        modify do insert element employment {
                        attribute startdate {"2011-09-01"},
                        attribute employer {"LBM"}} as last into $nye/root
        return $nye' 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><a>789</a></root>
modify do delete $x/a[text() = 789]
return $x
```

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


IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

DB2 - transform - delete

```
SELECT XMLQUERY('transform
                copy $nye := $e
                modify do delete $nye//employment[2]
                return $nye' 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 $nye := $e
                    modify do delete $nye//employment[2]
                    return $nye' 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><a>789</a></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><a b="ccc">456</a><a>789</a></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><a>789</a></root>
modify do rename $x/a[2] as "b"
return $x
```

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


DB2 - transform – många, loop

```
xquery
transform
copy $x := <root><a>456</a><a>789</a></root>
modify for $a in $x/a
        return do rename $a as "b"
return $x
```

```
<root>
        <b>456</b>
        <b>789</b>
</root>
```

DB2 - transform – många, lista

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


Oracle Database 19c

- Stöd för det mesta av SQL/XML enligt SQL 2006
- Extra tillägg
  - Metoder för XML-datatypen (kvalificering krävs!)
    - » Extract
    - » Transform
    - » …
  - Funktioner (många är deprecated fr o m version 12)
    - » Extract, ExtractValue, existsNode
    - » UpdateXML, InsertXML, DeleteXML, …
    - » XMLTransform, XMLColAttVal
    - » …
  - XQuery-tillägg (deprecated fr o m version 12)
    - » ora:view, ora:contains, ora:matches …
- Ännu flera Oracle-specifika lösningar i tidigare versioner, som en efter en fasas ut.

Oracle - datatyp

- XMLTYPE
  - Stöd för explicit associering till XML Schema
  - Strukturell validering
  - Sparar valideringstillståndet
  - Konstruktorfunktion XMLTYPE
    - » XMLTYPE('<a/>')
    - » Bara XML-dokument
  - Endast XML-dokument i kolumner
- Full validering med
  - proceduren SchemaValidate
    - » sparar resultatet
    - » som testas med metoden IsSchemaValidated
  - metoden IsSchemaValid
    - » returnerar resultatet
- Hanterar inte attributnoder så bra

Oracle - SQL/XML

- Stödjer följande funktioner
  - XMLELEMENT (delvis)
  - XMLATTRIBUTES
  - XMLFOREST
  - XMLCONCAT
  - XMLCOMMENT
  - XMLPI
  - XMLAGG
  - XMLPARSE
  - XMLNAMESPACES (stöds inte)
    - namespaces skapas med XMLATTRIBUTES!
  - XMLSERIALIZE
  - XMLTABLE (delvis)
  - XMLQUERY (endast CONTENT?)
  - XMLCAST
  - XMLEXISTS
  - Dynamiska nodnamn med nyckelordet EVALNAME
    - » Fungerar med XMLELEMENT, XMLATTRIBUTES, XMLFOREST, XMLPI

Oracle - andra funktioner

- XMLCDATA
- XMLISVALID
- XMLCOLATTVAL
- XMLTRANSFORM

Deprecated:

- EXTRACT
- EXTRACTVALUE
- EXISTSNODE
- XMLSEQUENCE

- Deprecated:
  - UPDATEXML
  - APPENDCHILDXML
  - INSERTCHILDXML
  - INSERTCHILDXMLAFTER
  - INSERTCHILDXMLBEFORE
  - INSERTXMLAFTER
  - INSERTXMLBEFORE
  - DELETEXML

- XMLROOT

Oracle - XMLTYPE

- Metoder (ibland kallade medlemsfunktioner "member functions")
  - extract
  - existsNode
  - transform
  - isSchemaValidated
  - isSchemaValid
  - isSchemaBased
  - isFragment
  - getStringVal
  - getNumberVal
  - getCLOBVal
  - getBLOBVal
  - getNamespace
  - getRootElement
  - getSchemaURL

Oracle - XMLCDATA

- Motsvarar SQL/XML:s XMLTEXT
  - Lägger texten inuti <![CDATA[värdet]]>

```
SELECT XMLELEMENT(NAME Person, XMLCDATA(name))
FROM person
WHERE pid = 1
```

```
<PERSON><![CDATA[John Higgins]]></PERSON>
```

```
SELECT XMLELEMENT(NAME Tecken, '<') FROM dual
<TECKEN>&lt;</TECKEN>

SELECT XMLELEMENT(NAME Tecken, XMLCDATA('<')) FROM dual
<TECKEN><![CDATA[<]]></TECKEN>
```


Oracle - XMLISVALID

- Motsvarar SQL/XML:s XMLVALIDATE
  - implicit schema
    XMLISVALID(xmlvärde)
  - explicit schema
    XMLISVALID(xmlvärde, schemanamn)

Oracle - XMLCOLATTVAL

- Genererar ett XML-fragment med ett column-element för varje kolumn/värde

```
SELECT XMLCOLATTVAL(name, yearofbirth)
FROM person WHERE pid = 1
```

```
<column name = "NAME">John Higgins</column>
<column name = "YEAROFBIRTH">1975</column>
```

```
SELECT XMLCOLATTVAL('nikos' AS "subjekt", 'hälsar' AS "predikat")
FROM dual
```

```
<column name = "subjekt">nikos</column>
<column name = "predikat">hälsar</column>
```


Oracle - EXTRACT

- Applicerar ett XPath-uttryck på ett XML-värde
  - Returnerar XML
  - Deprecated. Ersätts av SQL/XML:s XMLQUERY

```
SELECT name, EXTRACT(employments, '//employment[1]')
FROM person
```

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"/>
Ronnie O'Sullivan	<employment startdate="2002-03-10" enddate="2010-05-22" employer="LKP"/>
Ken Doherty	<employment startdate="2002-02-12" enddate="2003-05-11" employer="LKP"/>
Steve Davis	<employment startdate="2001-01-05" enddate="2005-12-31" employer="ABB"/>
Paul Hunter	<employment startdate="2004-01-10" enddate="2008-09-29" employer="FFD"/>
Neil Robertson	<employment startdate="2006-02-03" enddate="2008-10-30" employer="UPC"/>

Oracle - EXTRACT

```
SELECT name,
    EXTRACT(employments, '//employment[1]/@employer')
FROM person
```

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

Oracle - EXTRACTVALUE

- Applicerar ett XPath-uttryck på ett XML-värde
  - Returnerar ett värde
  - Deprecated. Ersätts av SQL/XML:s XMLQUERY

```
SELECT name,
    EXTRACTVALUE(employments, '//employment[1 ]/@employer')
FROM person
```

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

Oracle - EXISTSNODE

- Testar om ett XPath-uttryck matchar någon nod i ett XML-värde
  - Deprecated. Ersätts av XMLEXISTS
  - Returnerar 1 (sant) eller 0 (falskt)

```
SELECT name
FROM person
WHERE EXISTSNODE(employments,
'//employment[@employer="ABB"]') = 1
```

```
John Higgins
Stephen Hendry
Ken Doherty
Steve Davis
Neil Robertson
```


Oracle - XMLSEQUENCE

- Delar upp ett XML-fragment i flera XML-dokument
  - Deprecated. Ersätts av SQL/XML:s XMLTABLE

```
SELECT e.*
FROM person,
  TABLE(XMLSEQUENCE(EXTRACT(employments, '//employment'))) e
WHERE pid = 5
```

```
<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"/>
```

Obs! Fem rader i resultatet

Oracle - XMLTRANSFORM

- Omvandlar ett XML-dokument enligt ett XSLT-dokument (XSLT 1.0)

```
SELECT XMLTRANSFORM(employments ,
'<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:for-each select="//employment/@employer">
       <xsl:element name="Employer">
        <xsl:value-of select="."/>
       </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:transform>')
FROM person
WHERE name = 'Ken Doherty'
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Employers>
  <Employer>LKP</Employer>
  <Employer>ABB</Employer>
  <Employer>LKP</Employer>
  <Employer>FFD</Employer>
  <Employer>STG</Employer>
</Employers>
```


Oracle - XMLTRANSFORM

```
SELECT XMLTRANSFORM(employments ,
'<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:for-each select="//employment[not (@employer =
                                    preceding::employment/@employer)]/@employer">
       <xsl:element name="Employer">
        <xsl:value-of select="."/>
       </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:transform>')
FROM person
WHERE name = 'Ken Doherty'
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Employers>
  <Employer>LKP</Employer>
  <Employer>ABB</Employer>
  <Employer>FFD</Employer>
  <Employer>STG</Employer>
</Employers>
```

Oracle - XMLTYPE DML

- **DEPRECATED**
  Funktioner som skapar ett nytt förändrat XML-värde
  - UpdateXML
  - DeleteXML
  - AppendXML
  - InsertChildXML, InsertChildXMLBefore, InsertChildXMLAfter
  - InsertXMLBefore, InsertXMLAfter

- **Hela värdet i kolumnen måste uppdateras**

```
UPDATE person
SET employments = DML-funktion(employments, …)
WHERE …
```


Oracle - XML DML

- **XQuery Update Facility**
  - några syntaxskillnader
- **transform-uttryck**
  - copy-klausul
  - modify-klausul
    - » delete node(s)
    - » insert node(s)
    - » rename node
    - » replace node
  - return-klausul
  - nyckelord för insert
    - » before
    - » after
    - » as first into
    - » as last into
    - » into
  - nyckelord för replace
    - » (value of) … with …
  - nyckelord för rename
    - » as

Oracle - XML DML

- Hela XML-värdet måste uppdateras

```
UPDATE Person
SET employments = XMLQUERY('transform-uttryck'
                           PASSING employments
                           RETURNING CONTENT)
WHERE …
```


Oracle - transform - insert

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify insert node <a>123</a> as first into $x
return $x
```

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

Oracle - transform - insert

XQUERY

```
copy $x := <root><a>456</a><a>789</a></root>
modify insert node <a>123</a> before $x/a[2]
return $x
```

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

Obs! Buggigt i versioner före Oracle 18:
Funkar inte med t ex predikatet [text() = 789] eller [. = 789]
Dock funkar [number(.) = 789] eller [string(.) = "789"]

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

Oracle - transform - insert

XQUERY

```
copy $x := <root><a>456</a><a>789</a></root>
modify insert node attribute c {5} into $x/a[2]
return $x
```

```
<root>
        <a>456</a>
        <a c="5">789</a>
</root>
```

Oracle - transform - insert

```
UPDATE Person
SET employments =  XMLQUERY('
        copy $nye := $e
        modify insert node element employment {
                        attribute startdate {"2011-09-01"},
                        attribute employer {"LBM"}} as last into $nye/root
        return $nye' PASSING employments AS "e"
                RETURNING CONTENT)
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>
```


Oracle - transform - delete

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify delete node $x/a[2]
return $x


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

Obs! Buggigt i versioner före Oracle 18:
Funkar inte med t ex predikatet [text() = 789] eller [. = 789]
Dock funkar [number(.) = 789] eller [string(.) = "789"]

Oracle - transform - delete

```
SELECT XMLQUERY('copy $nye := $e
                 modify delete node $nye//employment[2]
                 return $nye' PASSING employments AS "e"
                 RETURNING CONTENT)
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>
```


Oracle - transform - delete

```
UPDATE Person
SET employments = XMLQUERY('copy $nye := $e
                            modify delete node $nye//employment[2]
                            return $nye' PASSING employments AS "e"
                                    RETURNING CONTENT)
WHERE pid = 3
```

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

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

Oracle - transform - replace

XQUERY

```
copy $x := <root><a>456</a><a>789</a></root>
modify replace node $x/a[2] with <b>123</b>
return $x
```

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

Obs! Buggigt i versioner före Oracle 18:
Funkar inte med t ex predikatet [text() = 789] eller [. = 789]
Dock funkar [number(.) = 789] eller [string(.) = "789"]

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

Oracle - transform - replace

XQUERY

```
copy $x := <root><a>456</a><a>789</a></root>
modify replace value of node $x/a[2] with 123
return $x
```

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

Oracle - transform - replace

```
XQUERY
copy $x := <root><a b="ccc">456</a><a>789</a></root>
modify replace node $x/a[1]/@b with attribute f {"ddd"}
return $x
```

```
<root>
        <a f="ddd">456</a>
        <a>789</a>
</root>
```


Oracle - transform - rename

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify rename node $x/a[2] as "b"
return $x
```

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

Oracle - transform - rename

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify for $a in $x/a
        return rename node $a as "b"
return $x
```

Fungerar inte. Returnerar $x oförändrad:

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


Oracle - transform - rename

Dock fungerar följande:

```
XQUERY
copy $x := <root><a>456</a><a>789</a></root>
modify (rename node $x/a[1] as "b“,
            rename node $x/a[2] as "b“)
return $x
```

Resultat:

```
<root>
        <b>456</b>
        <b>789</b>
</root>
```

Oracle - transform - rename

Men följande funkar inte:

XQUERY

```
copy $x := <root><a>456</a><a>789</a></root>
modify for $i in 1 to count($x/a)
        return rename node $x/a[position() = $i] as "b"
return $x
```

Resultatet är oförändrat:

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


Oracle - XMLTYPE-metoder

- Om objektet är i en kolumn, måste kolumnen kvalificeras med tabellaliasnamnet
  - tabellalias.kolumn.metod()
- Flera metoder motsvarar funktioner där XML-objektet är det första argumentet
  - EXTRACT(xmlobjekt, xpath-uttryck)
  - xmlobjekt.extract(xpath-uttryck)

Oracle - extract

- Motsvarar funktionen EXTRACT
- Returnerar XML givet ett XPath-uttryck

```
SELECT extract(employments, '//employment[1]'),
       p.employments.extract('//employment[1]')
FROM person p
```

De två kolumnerna i resultatet är identiska.

Oracle - existsNode

- Motsvarar funktionen EXISTSNODE
- Kontrollerar om ett XPath-uttryck matchar någon nod
  - Returnerar 1 (sant) eller 0 (falskt)

```
SELECT name,
       EXISTSNODE(employments, '//employment[@employer="ABB"]'),
       p.employments.existsNode('//employment[@employer="ABB"]')
FROM person p
```

John Higgins	1	1
Stephen Hendry	1	1
Matthew Stevens	0	0
Ronnie O'Sullivan	0	0
Ken Doherty	1	1
Steve Davis	1	1
Paul Hunter	0	0
Neil Robertson	1	1

Oracle - transform

- Motsvarar funktionen XMLTRANSFORM
  - Den tar inte emot XSLT-dokumentet som string

```
SELECT p.employments.transform(XMLTYPE('
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml"/>
  <xsl:template match="/">
    <xsl:element name="Employers">
      <xsl:for-each select="//employment/@employer">
       <xsl:element name="Employer">
         <xsl:value-of select="."/>
       </xsl:element>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:transform>'))
FROM person p
WHERE name = 'Steve Davis'
```

```
<Employers>
  <Employer>ABB</Employer>
  <Employer>LKP</Employer>
  <Employer>FFD</Employer>
</Employers>
```


Oracle - get…Val

- Metoder för omvandling till andra datatyper
  - getNumberVal
  - getStringVal Deprecated. Ersätts av SQL/XML:s XMLSERIALIZE
  - getBLOBVal Deprecated. Ersätts av SQL/XML:s XMLSERIALIZE
  - getCLOBVal Deprecated. Ersätts av SQL/XML:s XMLSERIALIZE
- Returnerar noden som ett värde av den valda typen
  - getNumberVal kräver en nod som kan omvandlas till nummer

```
SELECT XMLTYPE('<a v="44">55</a>').extract('//@v').getNumberVal()
FROM DUAL
```

```
SELECT XMLTYPE('<a v="44">55</a>').extract('//text()').getNumberVal()
FROM DUAL
```

Oracle - getRootElement

- Returnerar namnet av rotelementet
  - NULL om XML-objektet är ett XML-fragment
  - Deprecated. Ersätts av XQuery-funktionen local-name

```
SELECT XMLTYPE('<roten />').getRootElement()
FROM DUAL
```

Returnerar "roten"

```
SELECT EXTRACT(XMLTYPE('<a><b/><b/></a>'),'//b').getRootElement()
FROM DUAL
```

eller

```
SELECT XMLQUERY('for $a in (1,2) return <b/>'
                RETURNING CONTENT).getRootElement()
FROM DUAL
```

Returnerar NULL

Obs! XMLTYPE('<b/><b/>') fungerar inte.

Oracle - getSchemaURL

- Returnerar URL:en till det aktuella XML-dokumentets XML Schema
  - Eller NULL om inget schema är associerat

```
SELECT XMLTYPE('<a />').getSchemaURL()
FROM DUAL
```

Returnerar NULL

Oracle - schema-metoder

- isSchemaValid
  - Motsvarar funktionen XMLISVALID
  - Man kan ange specifikt XML Schema
  - Returnerar resultatet (1 eller 0)
- isSchemaValidated
  - returnerar det sparade värdet
- isSchemaBased
  - returnerar 0 eller 1

Oracle - isFragment

- Kollar om ett XML-värde är ett fragment eller ett dokument
  - Returnerar 1 (fragment) eller 0 (dokument)

```
SELECT XMLQUERY('for $a in (1,2) return <b/>'
                          RETURNING CONTENT).isFragment()
FROM DUAL
```

Returnerar 1

```
SELECT XMLTYPE('<a><b/><b/></a>').isFragment()
FROM DUAL
```

Returnerar 0

Oracle - ora:view

- Oracle-specifik XQuery-funktion
  - tar emot namnet av en tabell/vy (och eventuellt schemat)
  - returnerar ett XML-fragment där
    - » varje rad är ett ROW-element
    - » varje kolumn är ett element med kolumnens namn som elementnamn och värdet som textnod

```
SELECT XMLQUERY('ora:view("person")' RETURNING CONTENT)
FROM dual
```

```
<ROW><PID>1</PID><NAME>John Higgins</NAME><YEAROFBIRTH>1975</YEAROFBIRTH><EMPLOYMENTS><root>
 <employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/>
 <employment startdate="2009-04-15" employer="UPC"/>
</root>
</EMPLOYMENTS></ROW>
<ROW><PID>2</PID><NAME>Stephen Hendry</NAME><YEAROFBIRTH>1973</YEAROFBIRTH><EMPLOYMENTS><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>
</EMPLOYMENTS></ROW>
...
```


Oracle - ora:view

```
SELECT XMLQUERY('for $r in ora:view("person")/ROW[PID = (1,2,3)]
                 let $pd := $r//PID|$r//NAME
                 return element Person {$pd}'
                 RETURNING CONTENT)
FROM dual
```

```
<Person><PID>1</PID><NAME>John Higgins</NAME></Person>
<Person><PID>2</PID><NAME>Stephen Hendry</NAME></Person>
<Person><PID>3</PID><NAME>Matthew Stevens</NAME></Person>
```

Oracle - ora:view

- DEPRECATED. Ersätts av fn:collection
  - tar en URI som parameter:
  - oradb:/schemanamn/tabellnamn
  - oradb:/PUBLIC/tabellnamn
  - Case-sensitive

```
SELECT XMLQUERY('fn:collection("oradb:/PUBLIC/PERSON")'
RETURNING CONTENT)
FROM dual
```

```
<ROW><PID>1</PID><NAME>John Higgins</NAME><YEAROFBIRTH>1975</YEAROFBIRTH><EMPLOYMENTS><root>
 <employment startdate="2001-08-20" enddate="2009-02-28" employer="ABB"/>
 <employment startdate="2009-04-15" employer="UPC"/>
</root>
</EMPLOYMENTS></ROW>
<ROW><PID>2</PID><NAME>Stephen Hendry</NAME><YEAROFBIRTH>1973</YEAROFBIRTH><EMPLOYMENTS><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>
</EMPLOYMENTS></ROW>
...
```


Oracle - XQUERY

- Stöd för XQuery-uttryck
  - XQuery-uttryck efter nyckelordet XQUERY

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

Tre rader i resultatet:

```
   <Nummer>1</Nummer>
   <Nummer>2</Nummer>
   <Nummer>3</Nummer>
```

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

```
   <Resultat>
    <Nummer>1</Nummer><Nummer>2</Nummer><Nummer>3</Nummer>
   </Resultat>
```

Microsoft SQL Server 2019

- Datatyp
  - Stöd för validering
  - Enligt SQL-standarden
- FOR XML
- OPENXML
- XML-metoder
  - query (XQuery)
  - value
  - exist
  - modify (DML)
  - nodes
- Begränsat stöd för XQuery
  - få funktioner
  - inte alla axes
  - delvis stöd för let-klausulen

SQL Server - datatyp

- XML
  - untyped - well-formed XML och XML-fragment
  - typed - kopplad till ett XML Schema
  - Stödjer inte DTD
  - Stödjer inline DTD för t ex default-värden
- XML SCHEMA COLLECTION
  - registrerar och namnger XML Schema
  - används för att definiera typed XML
  - CREATE XML SCHEMA COLLECTION namn AS 'schemat'
- Metoder
  - query, value, exist, modify, nodes
- Begränsat stöd för XPath och XQuery
  - Stödjer inte alla axes
  - Stödjer inte alla funktioner
  - Stödjer inte computed constructors fullt och inte i let-klausulen

SQL Server - typed XML

- Schemat måste läggas i en 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

- Schemat måste läggas i en 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

- Skapa kolumnen som en typed XML baserad på den redan skapade 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));
```

- Innehållet valideras vid INSERT och UPDATE

SQL Server & SQL/XML

- Stödjer inte publiceringsfunktionerna
  - FOR XML-klausul i SELECT-satsen istället
- Stödjer inte XMLQUERY, XMLTABLE, XMLEXISTS
  - XML-metoder istället (delvis även OPENXML)
- Stödjer inte XMLVALIDATE
  - validering sker automatiskt för typed XML
- Stödjer inte XMLCAST
  - Den generella SQL-funktionen CAST räcker (även CONVERT)

SQL Server - OPENXML

- Omvandlar XML-data till relationsdata
- Kräver användning av procedurer
- Kan generera en rad per nod
  - Default, resultatet är en så kallad edge table
- Kan generera en kolumn per angivet uttryck
  - Med nyckelordet WITH
- Lämplig för batch-shredding av XML-filer

SQL Server - FOR XML

- En extra klausul i SELECT-satser
- Omvandlar resultatet till XML enligt olika modes
  - RAW
  - AUTO
  - EXPLICIT (bör undvikas)
  - PATH
- Resultatet är alltid ett XML-värde
  - Antingen fragment eller well-formed XML med nyckelordet ROOT
  - Antingen serialiserat eller av XML-typen med nyckelordet TYPE

SQL Server - FOR XML RAW

- Varje rad blir ett element
  - default elementnamn: "row"
  - varje kolumn blir ett attribut

```
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 anger att ett rotelement skall skapas
  - default elementnamn: "root"
- Elementnamn kan specificeras
  - rotelementet efter ROOT
  - row-elementet efter RAW

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

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

SQL Server - FOR XML RAW

- ELEMENTS anger att element skall skapas istället för attribut
  - elementnamnen (eller attributnamn) kan styras i SELECT-klausulen

```
SELECT pid AS PersonID, name AS Namn,
            yearofbirth AS FödelseÅr
FROM person
WHERE pid = 2
FOR XML RAW ('Person'), ROOT ('Folk'), ELEMENTS
```

```
<Folk>
 <Person>
  <PersonID>2</PersonID>
  <Namn>Stephen Hendry</Namn>
  <FödelseÅr>1973</FödelseÅr>
 </Person>
</Folk>
```


SQL Server - FOR XML AUTO

- Skapar automatiskt element
  - I enkla situationer nästan identisk med RAW
  - default elementnamn: tabellens/aliasets namn

```
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

- **Nästlar automatiskt**
  - Ett element per använd tabell (i FROM och SELECT)
  - Nästlingen styrs av kolumnernas ordning i SELECT-klausulen
  - ORDER BY-klausulen styr grupperingen av subelement

```
SELECT pid, name AS Namn, model, brand, yearofbirth,
            licencenumber, color AS Färg
FROM person AS Person, car AS Bil
WHERE pid < 3 AND owner = pid
FOR XML AUTO
```

```
<Person pid="1" Namn="John Higgins" yearofbirth="1975">
  <Bil model="Cherry" brand="NISSAN" licencenumber="ABC123" Färg="black" />
</Person>
<Person pid="2" Namn="Stephen Hendry" yearofbirth="1973">
  <Bil model="Forza" brand="FIAT" licencenumber="CCD457" Färg="blue" />
</Person>
<Person pid="1" Namn="John Higgins" yearofbirth="1975">
  <Bil model="Cherry" brand="NISSAN" licencenumber="TYD226" Färg="black" />
</Person>
```


IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

SQL Server - FOR XML AUTO

```
SELECT model, pid, name AS Namn, brand, yearofbirth,
            licencenumber, color AS Färg
FROM person AS Person, car AS Bil
WHERE pid < 3 AND owner = pid
FOR XML AUTO
```

```
<Bil model="Cherry" brand="NISSAN" licencenumber="ABC123" Färg="black">
  <Person pid="1" Namn="John Higgins" yearofbirth="1975" />
</Bil>
<Bil model="Forza" brand="FIAT" licencenumber="CCD457" Färg="blue">
  <Person pid="2" Namn="Stephen Hendry" yearofbirth="1973" />
</Bil>
<Bil model="Cherry" brand="NISSAN" licencenumber="TYD226" Färg="black">
  <Person pid="1" Namn="John Higgins" yearofbirth="1975" />
</Bil>
```

SQL Server - FOR XML AUTO

- ORDER BY-klausulen styr grupperingen av subelement

```
SELECT pid, name AS Namn, brand, model, yearofbirth,
            licencenumber, color AS Färg
FROM person AS Person, car AS Bil
WHERE pid < 3 AND owner = pid
ORDER BY pid
FOR XML AUTO
```

```
<Person pid="1" Namn="John Higgins" yearofbirth="1975">
  <Bil brand="NISSAN" model="Cherry" licencenumber="ABC123" Färg="black" />
  <Bil brand="NISSAN" model="Cherry" licencenumber="TYD226" Färg="black" />
</Person>
<Person pid="2" Namn="Stephen Hendry" yearofbirth="1973">
  <Bil brand="FIAT" model="Forza" licencenumber="CCD457" Färg="blue" />
</Person>
```


SQL Server - FOR XML AUTO

- Nästla för att tvinga fram önskat resultat

```
SELECT färg, regnr, märke, ägare
FROM (SELECT DISTINCT color AS färg FROM car) AS Färg,
      (SELECT licencenumber AS regnr, color, brand AS märke, name AS ägare
      FROM car, person WHERE owner = pid) AS Bil
WHERE color = färg
ORDER BY färg
FOR XML AUTO
```

```
<Färg färg="black">
  <Bil regnr="ABC123" märke="NISSAN" ägare="John Higgins" />
  <Bil regnr="RSQ199" märke="NISSAN" ägare="Ronnie O'Sullivan" />
  <Bil regnr="TYD226" märke="NISSAN" ägare="John Higgins" />
</Färg>
<Färg färg="blue">
  <Bil regnr="CCD457" märke="FIAT" ägare="Stephen Hendry" />
  <Bil regnr="ROO197" märke="SAAB" ägare="Ken Doherty" />
</Färg>
<Färg färg="green">
  <Bil regnr="DKL998" märke="SAAB" ägare="Matthew Stevens" />
</Färg>
<Färg färg="red">
  <Bil regnr="PTF357" märke="VOLVO" ägare="Steve Davis" />
  <Bil regnr="WID387" märke="FIAT" ägare="Matthew Stevens" />
</Färg>
```

SQL Server - FOR XML AUTO

- **Stödjer ROOT, TYPE och ELEMENTS**

```
SELECT pid, name AS Namn, brand, model, yearofbirth, licencenumber, color AS Färg
FROM person AS Person, car AS Bil
WHERE pid < 2 AND owner = pid
ORDER BY pid
FOR XML AUTO, ROOT, TYPE, ELEMENTS
```

```
<root>
 <Person>
  <pid>1</pid>
  <Namn>John Higgins</Namn>
  <yearofbirth>1975</yearofbirth>
  <Bil>
   <brand>NISSAN</brand>
   <model>Cherry</model>
   <licencenumber>ABC123</licencenumber>
   <Färg>black</Färg>
  </Bil>
  <Bil>
   <brand>NISSAN</brand>
   <model>Cherry</model>
   <licencenumber>TYD226</licencenumber>
   <Färg>black</Färg>
  </Bil>
 </Person>
</root>
```


SQL Server - FOR XML PATH

- **Mer flexibel än RAW och AUTO**
  - Man kan blanda element och attribut
  - Man kan själv bestämma hur element nästlas
- **Kolumnalias i SELECT-klausulen tolkas som XPath**
  - Default liknar RAW, ELEMENTS

```
SELECT licencenumber, color, brand AS märke, name AS ägare
FROM car, person
WHERE owner = pid
FOR XML PATH
```

```
SELECT licencenumber, color, brand AS märke, name AS ägare
FROM car, person
WHERE owner = pid
FOR XML RAW, ELEMENTS
```

SQL Server - FOR XML PATH

```
SELECT licencenumber AS "@regnr", color AS "@färg",
                brand AS "@märke", name AS ägare
FROM car, person
WHERE owner = pid AND owner < 3
FOR XML PATH ('Bil')
```

```
<Bil regnr="ABC123" färg="black" märke="NISSAN">
  <ägare>John Higgins</ägare>
</Bil>
<Bil regnr="CCD457" färg="blue" märke="FIAT">
  <ägare>Stephen Hendry</ägare>
</Bil>
<Bil regnr="TYD226" färg="black" märke="NISSAN">
  <ägare>John Higgins</ägare>
</Bil>
```


SQL Server - FOR XML PATH

```
SELECT licencenumber AS "@regnr", color AS "@färg",
        brand AS "Typ/@märke", model AS "Typ/@modell",
        name AS "Ägare/@namn", pid AS "Ägare/@pid"
FROM car, person
WHERE owner = pid AND owner < 3
FOR XML PATH ('Bil')
```

```
<Bil regnr="ABC123" färg="black">
  <Typ märke="NISSAN" modell="Cherry" />
  <Ägare namn="John Higgins" pid="1" />
</Bil>
<Bil regnr="CCD457" färg="blue">
  <Typ märke="FIAT" modell="Forza" />
  <Ägare namn="Stephen Hendry" pid="2" />
</Bil>
<Bil regnr="TYD226" färg="black">
  <Typ märke="NISSAN" modell="Cherry" />
  <Ägare namn="John Higgins" pid="1" />
</Bil>
```

SQL Server - FOR XML PATH

- Använd * för att skapa en textnod (utan eget element)

```
SELECT name "*", ' äger ' "*",
      (SELECT COUNT(*) FROM car WHERE owner = pid) "*", ' bilar' "*"
FROM person
FOR XML PATH ('Påstående'), ROOT ('Info')
```

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


SQL Server - FOR XML PATH

- Nästla för att gruppera
  - ORDER BY fungerar inte som i AUTO
  - Använd TYPE i den nästlade satsen för att returnera XML

```
SELECT color AS "@namn",
      (SELECT  licencenumber AS "@regnr", brand AS "@märke",
                        model AS "@modell", name AS "Ägare/@namn"
       FROM car, person
       WHERE owner = pid AND color = färg.color
       FOR XML PATH ('Bil'), TYPE) AS "*"
FROM (SELECT DISTINCT color FROM car) AS färg
FOR XML PATH ('Färg'), ROOT ('BilarPerFärg')
```

SQL Server - FOR XML PATH

```
<BilarPerFärg>
	<Färg namn="black">
		<Bil regnr="ABC123" märke="NISSAN" modell="Cherry">
			<Ägare namn="John Higgins" />
		</Bil>
		<Bil regnr="RSQ199" märke="NISSAN" modell="Micra">
			<Ägare namn="Ronnie O'Sullivan" />
		</Bil>
		<Bil regnr="TYD226" märke="NISSAN" modell="Cherry">
			<Ägare namn="John Higgins" />
		</Bil>
	</Färg>
	<Färg namn="blue">
		<Bil regnr="CCD457" märke="FIAT" modell="Forza">
			<Ägare namn="Stephen Hendry" />
		</Bil>
		<Bil regnr="ROO197" märke="SAAB" modell="900i">
			<Ägare namn="Ken Doherty" />
		</Bil>
	</Färg>
	<Färg namn="green">
		<Bil regnr="DKL998" märke="SAAB" modell="9000C">
			<Ägare namn="Matthew Stevens" />
		</Bil>
	</Färg>
	<Färg namn="red">
		<Bil regnr="PTF357" märke="VOLVO" modell="V70">
			<Ägare namn="Steve Davis" />
		</Bil>
		<Bil regnr="WID387" märke="FIAT" modell="Nova">
			<Ägare namn="Matthew Stevens" />
		</Bil>
	</Färg>
</BilarPerFärg>
```


SQL Server - FOR XML

- NULL leder till att noden inte genereras
  - Med nyckelordet XSINIL (efter ELEMENTS) genereras istället en elementnod med attributet xsi:nil="true"
- Nyckelorden XMLDATA och XMLSCHEMA genererar ett schema för resultatet
  - Ganska begränsat
- Använd '' som elementnamn för att eliminera den nivån
- Använd WITH XMLNAMESPACES (före SELECT-klausulen) för att definiera namespaces
  - WITH XMLNAMESPACES ('nsuri' AS nsalias)
  - WITH XMLNAMESPACES (DEFAULT 'nsuri')

SQL Server - value()

- Metod som returnerar värdet som finns på ett angivet XPath-uttryck (eller resultatet av ett XQuery-uttryck)
  - XML-objekt.value(xquery, datatyp)
  - xquery måste ge ett värde och måste avslutas med [1] om uttrycket kan vara en sekvens
    - » //employment[1]/@employer
      måste skrivas om till
      (//employment[1]/@employer)[1]

```
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()

- Applicerar ett XQuery-uttryck och returnerar 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

Resultatet av metoden query är XML

SQL Server - nodes()

- Omvandlar en sekvens av noder till en tabell med en kolumn
  - Varje rad motsvarar en nod
  - Resultatet är en relativ nodposition i det ursprungliga XML-objektet
  - Kan användas ihop med nyckelordet CROSS (eller OUTER) APPLY (för att få använda en kolumn från en tabell direkt i FROM-klausulen)
  - Motsvarar 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" />
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" />
Paul Hunter	NULL
Neil Robertson	<employment startdate="2008-11-20" employer="ABB" />

SQL Server - exist()

- Utvärderar ett XQuery-uttryck och returnerar
  - 1 om resultatet inte var tomt
  - 0 om resultatet var tomt
  - NULL om objektet är 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()

- Förändrar ett XML-objekt
- Man kan utföra DML-operationer med följande nyckelord
  - insert
  - delete
  - replace value of
- Validering görs på resultatet vid typed XML
- Används endast i SET-klausulen i UPDATE-satser
  - eller för objekt i variabler

SQL Server - modify() - insert

- Lägger till nya noder
  - as first
  - as last
  - into
  - after
  - before
- Noderna kan skapas, anges som text eller hämtas från en variabel

```
UPDATE person
SET employments.modify('insert <employment startdate="2011-11-12"
employer="KFC"/> as last into /root[1]')
WHERE pid = 6
```


IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

SQL Server - modify() - insert

```
UPDATE person
SET employments.modify('insert <!-- ny anställning tillagd --> 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
```

Om attributet startdate är xs:date (i XML Schema vid typed XML):

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

eller

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

SQL Server - modify() - delete

- Tar bort noder som matchar ett XPath-uttryck
  - Rotnoden får inte tas bort
  - Accepterar wildcards

```
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

- Ersätter en nods värde
  - replace value of xpath-uttryck with nyttvärde

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

För 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

- Metoderna query, nodes, etc kräver ett XML-objekt
- För att köra XQuery oberoende av tabeller, skapa ett XML-objekt enligt följande:

```
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>
```

Eller:

```
SELECT CONVERT(XML, '').query(…)
```


SQL Server - sql:column

- SQL Server-specifik XQuery-funktion
  - Gör ett kolumnvärde tillgängligt i XQuery

```
SELECT employments.query('element Person {attribute
  ålder {2015-sql:column("yearofbirth")}, attribute namn
  {sql:column("name")}}')
FROM person
```

```
<Person ålder="36" namn="John Higgins" />
<Person ålder="38" namn="Stephen Hendry" />
<Person ålder="29" namn="Matthew Stevens" />
<Person ålder="31" namn="Ronnie O'Sullivan" />
<Person ålder="37" namn="Ken Doherty" />
<Person ålder="51" namn="Steve Davis" />
<Person ålder="28" namn="Paul Hunter" />
<Person ålder="29" namn="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 Anställning {attribute ålder {$sy - $y},
                                                $e/@employer}
                                                ')
FROM person p
WHERE pid = 2
```

Stephen Hendry

```
<Anställning ålder="29" employer="ABB" />
<Anställning ålder="30" employer="UPC" />
<Anställning ålder="33" employer="ABB" />
```

Obs! SQL Server stödjer inte så många XQuery-funktioner, så vi tvingas använda substring och cast istället för year-from-date (eller number efter substring)

SQL Server - sql:variable

- SQL Server-specifik XQuery-funktion
  - Gör en variabel tillgänglig i XQuery

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

```
<X>1 hus</X><X>2 hus</X><X>3 hus</X>
```

SQL Server - kombinationer

- Det mesta går att kombinera. Vissa begränsningar finns.

```
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 Anställning {attribute Ålder {$sy - $y},
                                    attribute Arbetsgivare  {$e/@employer}}')
FROM person p
WHERE pid < 3
FOR XML PATH ('Person'), ROOT ('Folk')
```

```
<Folk>
 <Person Namn="John Higgins">
  <Anställning Ålder="26" Arbetsgivare="ABB" />
  <Anställning Ålder="34" Arbetsgivare="UPC" />
 </Person>
 <Person Namn="Stephen Hendry">
  <Anställning Ålder="29" Arbetsgivare="ABB" />
  <Anställning Ålder="30" Arbetsgivare="UPC" />
  <Anställning Ålder="33" Arbetsgivare="ABB" />
 </Person>
</Folk>
```


SQL Server - kombinationer

```
SELECT (SELECT name AS "@Namn", COUNT(*) AS "@AntalBilar",
                AntalAnst AS "@AntalAnställningar"
        FROM (SELECT pid, name,
                employments.value('count(//employment)', 'integer') as AntalAnst
                FROM person) AS p, car
        WHERE pid = owner
        GROUP BY name, antalanst
        FOR XML PATH ('Person'),
                ROOT ('Folk'),
                TYPE).query('element Rot {//Person[@AntalBilar = 1]}')
```

```
<Rot>
 <Person Namn="Ronnie O'Sullivan" AntalBilar="1" AntalAnställningar="2" />
 <Person Namn="Stephen Hendry" AntalBilar="1" AntalAnställningar="3" />
 <Person Namn="Steve Davis" AntalBilar="1" AntalAnställningar="3" />
 <Person Namn="Ken Doherty" AntalBilar="1" AntalAnställningar="5" />
</Rot>
```

Sammanfattning

- Oracle och DB2 följer SQL-standarden i hög grad
- SQL Server följer SQL-standarden endast vad gäller datatypen och delvis XQuery
  - Hopp finns: SQL Server anger att alla SQL/XML-nyckelord är reserverade ord inför kommande versioner.
- Användning av standarden förenklar migrering
  - Undvik produktspecifika lösningar och deprecated funktionalitet

Fortsättning

- Labb om DB2 & XML (kompendium)
- Labb om Oracle & XML (kompendium)
- Labb om SQL Server & XML (kompendium)
- Inlupp 4 (DB2 & XML)
- Inlupp 5 (Oracle & XML)
- Inlupp 6 (SQL Server & XML)