IV1023 vt2022
Avancerad Datahantering med XML

Frågespråk för SSD och XML

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

Läsanvisningar
Utdrag från Data on the Web
Kapitel 2, 3, 9, 10, 11, 13.3, A.3.1, C.2 i kursboken
Delar av kapitel 31 i Database Systems (Connolly, Begg) upplaga 5
Artiklar om XML-frågespråk
Kompendium om XQuery

Frågespråk

- **Traversera datastrukturen**
  - **Path-uttryck (SSD) (Path expression)**
  - **XPath (XML)**
- **Ställa frågor**
  - **Lorel (SSD)**
  - **XQuery (XML)**
  - **Även på metadata!**
- **Uppdatera/förändra data/strukturen**
  - **XQuery Update Facility**

Frågespråk

- Generella egenskaper
  - Ställa frågor mot databasen
  - Villkor
  - Aggregering
  - Funktioner och operationer
  - Slutet språk
- Speciellt för SSD och XML
  - Traversering av strukturen
  - Ställa frågor mot metadata

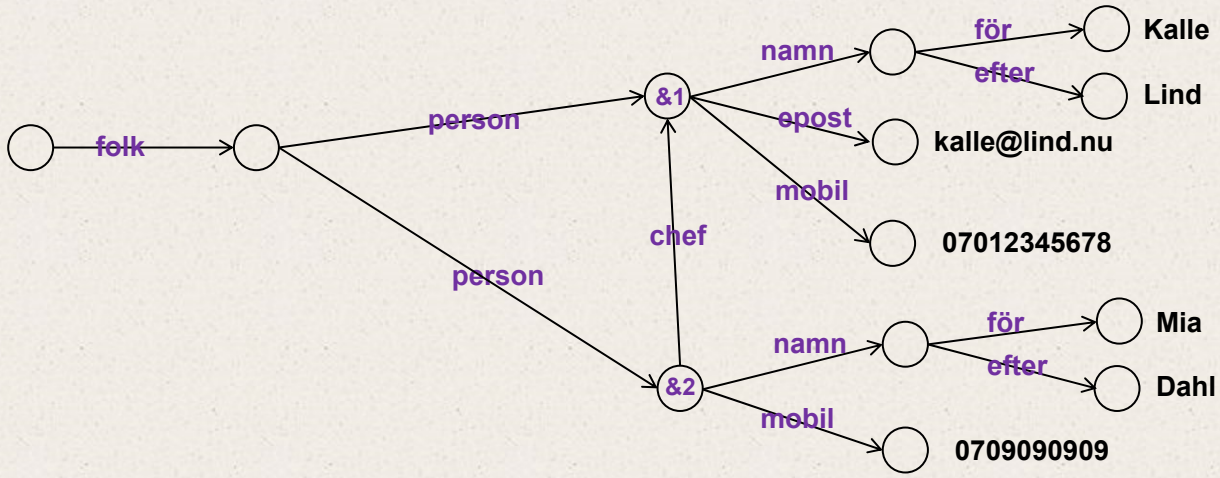
Path-uttryck

- Traversering av strukturen
  - Sekvens av labels (SSD) eller nodnamn (XML)
- Resultatet är en nodsekvens (eller nodmängd)
- Ett begränsat frågespråk

SSD Path-uttryck

- Sekvens av etiketter (labels)
  - x.y.z
- Wildcards
  - _
  - _+
  - _*
- Alternativ
  - x|y
- Variabler
  - x.L.z (variabler med versaler)
- Resultatet är en nodmängd

Path-uttryck - Exempel



- folk.person.namn.för
- folk.person.(epost|mobil)
- folk._*.mobil
- folk._.namn

Lorel

- Lore Language
  - Lore (Lightweight Object REpository)
- Bygger på OQL (Object Query Language)
  - OQL är baserat på (inspirerat av) SQL
- select … from … where …
- Input: SSD
- Output: SSD

- Även stöd för övrig DML

Exempeldata

```
db: {person:{namn:{för:"Kalle", efter:"Lind"},
             epost:"kalle@lind.se",
             mobil:"070111222",
             hemtel:"08151515"},
    person:{namn:{för:"Maria", efter:"Berg"},
             epost:"mb@home.se",
             mobil:"070444555"},
    person:{namn:{för:"Peter", smek:"Blixten" efter:"Larsson"},
             epost:"blixten@gmail.com",
             hemtel:"08789789"},
    person:{namn:{för:"Lisa", efter:"Lind"},
             epost:"lisa@lind.se",
             mobil:"070636363",
             hemtel:"08151515"},
    person:{namn:{för:"Mia", smek:"Stickan" efter:"Persson"},
             mobil:"070199991",
             epost:"miap@gmail.com",
             hemtel:"08199991"}
}
```

Lorel - Exempel

```
select namn:N
from db.person.namn N
```

Ett varv per möjlig N-nod.

Resultatet:

```
{namn:{för:"Kalle", efter:"Lind"},
namn:{för:"Maria", efter:"Berg"},
namn:{för:"Peter", smek:"Blixten" efter:"Larsson"},
namn:{för:"Lisa", efter:"Lind"},
namn:{för:"Mia", smek:"Stickan" efter:"Persson"}}
```


Lorel - Exempel

```
select namn:N
from db.person P, P.namn N
```

```
{namn:{för:"Kalle", efter:"Lind"},
namn:{för:"Maria", efter:"Berg"},
namn:{för:"Peter", smek:"Blixten" efter:"Larsson"},
namn:{för:"Lisa", efter:"Lind"},
namn:{för:"Mia", smek:"Stickan" efter:"Persson"}}
```

Lorel - Exempel

```
select namn:N
from db.person P, P.namn N, N.smek S
where S = "Blixten"
```

```
select namn:N
from db.person P, P.namn N
where N.smek = "Blixten"
```

N.smek är formellt en mängd och bör därför hanteras som en mängd:

```
select namn:N
from db.person P, P.namn N
where "Blixten" in N.smek
```


Lorel - Exempel, exists

```
select namn:N
from db.person.namn N
where exists E in N.efter : E = "Lind"
```

```
select namn:N
from db.person.namn N
where "Lind" in N.efter
```

```
select namn:N
from db.person.namn N
where "Lind" = N.efter
```

Lorel - Exempel, nästlade

```
select person:(select smeknamn:S
                      from N.smek S)
from db.person.namn N
```

```
select person:{smeknamn:S}
from db.person.namn N, N.smek S
```

Är resultatet identiskt?

Lorel - Exempel, join

```
select namn:N
from db.person P, P.namn N, db.person P2, P2.namn N2
where not (P = P2)
and N2.efter = N.efter
```

```
select namn:N
from db.person P, P.namn N
where exists P2 in db.person:
                        not (P = P2)
                        and N.efter = P2.namn.efter
```

Lorel - Exempel, labels

```
select typ:L
from db.person.namn N, N.L X
where X = "Lisa"
```

```
select L:V
from db.person P, P.L V
where L in ("mobil", "hemtel")
and "Kalle" in P.namn.för
```


Lorel - Exempel, resultat

```
select person:{namn:NF,
                    kontakt:{tel:M, mejl:EP}}
from db.person P,
     P.namn N,
     P.epost EP,
     P.mobil M,
     N.för NF
```

```
select person:{namn:N.för,
                    kontakt:{tel:P.mobil, mejl:P.epost}}
from db.person P, P.namn N
```

XPath

- XPath 1.0
  - Begränsad
  - Skapad i samband med XSLT 1.0
  - Bygger på Infoset-modellen
  - Använder nodmängder
- XPath 2.0
  - Flera nya funktioner, operationer, etc.
  - Anpassad till XQuery 1.0-modellen
  - Används även av XSLT 2.0
  - Använder nodsekvenser
- XPath 3.0 och 3.1
  - ihop med XQuery 3.0 och XSLT 3.0
  - dynamiska funktioner, mm

XQuery 1.0

- Standard
- Modell
- Frågespråk
  - bygger på SQL, XQL, XML-QL, Lorel, YATL, etc.
  - deklarativt (ej procedurellt)
  - inkluderar XPath 2.0
  - XQueryX - XQuery i XML-syntax
  - FLWOR (for let where order by return)
    - » Motsvarar SQL SELECT
  - transform-satser för övrig DML (från mars 2011)
    - » separat specifikation
- Nästa version XQuery 3.0
  - ihop med XPath 3.0 och XSLT 3.0

Exempeldata

```
<Filmer>
    <Film Titel="Driven" År="2001">
        <Skådis Namn="Burt Reynolds" Födelseår="1936" Land="USA"/>
        <Skådis Namn="Silvester Stallone" Födelseår="1946" Land="USA"/>
        <Skådis Namn="Kip Pardue" Födelseår="1976" Land="Canada"/>
        <Regissör Namn="Silvester Stallone" Födelseår="1946" Land="USA"/>
        <Produktionsbolag>Tri-Star</Produktionsbolag>
    </Film>
    <Film Titel="Antz" År="1998">
        <Skådis Namn="Woody Allen" Födelseår="1935" Land="USA"/>
        <Skådis Namn="Silvester Stallone" Födelseår="1946" Land="USA"/>
        <Skådis Namn="Sharon Stone" Födelseår="1958" Land="USA"/>
        <Regissör Namn="Eric Darnell" Födelseår="1961" Land="Ireland"/>
        <Produktionsbolag>Universal</Produktionsbolag>
    </Film>
    <Film Titel="Picking Up the Pieces" År="2000">
        <Skådis Namn="Woody Allen" Födelseår="1935" Land="USA"/>
        <Skådis Namn="Sharon Stone" Födelseår="1958" Land="USA"/>
        <Skådis Namn="Alfonso Arau" Födelseår="1948" Land="USA"/>
        <Regissör Namn="Eric Darnell" Födelseår="1961" Land="Ireland"/>
        <Produktionsbolag>Tri-Star</Produktionsbolag>
    </Film>
    …
</Filmer>
```


XML Path-uttryck (XPath)

- **sekvens av elementnamn/nodnamn**
  - **elementX/elementY/elementZ**
- **attribut**
  - **@attribut**
- **Union | och Konkatenering ,**
  - **Union | endast med noder**
  - **Konkatenering , noder och värden**
- **Snitt och differens (fr o m XPath 2)**
  - **Endast noder**
- **Axes**
  - **child, parent, ancestor, descendant, following, preceding, …**
  - **förkortningar: . och .. ("current node" och "parent node")**
- **Predikat**
  - **[villkor]**

XPath Exempel

- Alla Filmer (Film-noder):
  - /Filmer/Film
  - //Film
- Alla Filmer (Film-noder) från år 2000
  - //Film[@År=2000]
- År för filmer av Universal
  - //Film[Produktionsbolag='Universal']/@År
- Regissörer för filmer från 2000 och 2003
  - //Film[@År=2000]/Regissör | //Film[@År=2003]/Regissör
- Titel på filmer med Woody Allen
  - //Skådis[@Namn='Woody Allen']/../@Titel

XPath Axes

- child
  - //Film/child::Regissör
  - förkortning: //Film/Regissör
- descendant
  - child, eller child's child, etc.
  - /Filmer/descendant::Regissör
  - förkortning: /Filmer//Regissör
- parent
  - //Regissör/parent::Film
  - förkortning: //Regissör/.. (inte garanterat att det är Film)
  - //Regissör/parent::*
- ancestor
  - parent, eller parent's parent, etc
  - //Regissör/ancestor::Filmer

XPath Axes

- attribute
  - //Film/attribute::Titel
  - förkortning: //Film/@Titel
- namespace (deprecated i XPath 2.0)
  - /Filmer/namespace::*
  - ersatt av funktioner
- self
  - //Film/self::Film
  - förkortning: //Film/.
- descendant-or-self
  - //Film/descendant-or-self::Regissör
  - //Film/descendant-or-self::Film
- ancestor-or-self
  - //Regissör/ancestor-or-self::Regissör
  - //Regissör/ancestor-or-self::Film

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

XPath Axes

- following-sibling
  - //Film/Skådis/following-sibling::Skådis
- preceding-sibling
  - //Film/Skådis/preceding-sibling::Skådis
- following
  - noder som följer dock inte descendants eller attribute och namespace
  - //Skådis/following::Film
- preceding
  - noder som kommer före, dock inte ancestors eller attribute och namespace
  - //Skådis/preceding::Film

XQuery/XPath-funktioner

- **Sekvensfunktioner:**
  - distinct-values(s)
    - » baserad på string-value()
  - count(s), min(s), max(s), sum(s), avg(s)
  - empty(s), exists(s)
  - reverse(s)

- **Nodfunktioner:**
  - name(n), local-name(n), node-name(n)

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

XQuery/XPath-funktioner

- **String-funktioner:**
  - matches(s, regexp)
  - concat(s1,s2)
    - » operator || fr o m XPath 3 och XQuery 3
    - » s1 || s2
  - starts-with(s1,s2), ends-with(s1,s2), contains(s1,s2)
  - substring(s, start), substring(s, start, length)
  - lower-case(s), upper-case(s)
  - replace(s, pattern, replacement)
  - tokenize(s, pattern)

XQuery/XPath-funktioner

- Andra funktioner:
  - doc(URI)
  - put(n, URI)
  - not(e)
  - Många datum/tidfunktioner
  - Många numeriska funktioner
  - data(ns) – sekvens av noder till sekvens av enkla (atomic) värden
  - number(n) – värdet av noden som nummer eller NaN
  - string(n) – nodens värde som string
  - current-time(), current-date(), current-dateTime()
  - position() - nodens position i den aktuella sekvensen
  - last() - returnerar positionen av den sista noden i sekvensen

XQuery-funktioner

- Wildcards
  - node() (alla noder utom attribut och namespace)
  - text()
  - comment()
  - processing-instruction()
  - element(), *
  - attribute(), @*

XQuery/XPath-operatorer

- +, -, *, div, mod
- =, !=, >, <, <=, >= (generella jämförelser)
- eq, ne, lt, le, gt, ge (värdejämförelser)
- or, and
- is, >>, << (nodjämförelser)

- to (skapar sekvenser)
  - 1 to 5 = (1,2,3,4,5)

- union, intersect, except
  - kräver nodsekvenser
  - använder nodidentiteterna

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

XQuery/XPath-operatorer

- / (Path operator)
  - tar bort dubbletter av noder
  - använder nodidentiteterna

- , (Comma operator) och () (skapar sekvenser)
  - (1,2,3,4)
  - ((1,2), 3, (4,5)) blir (1,2,3,4,5)
  - () tom sekvens
  - (//Skådis, //Regissör)

XPath - Predikat

- Villkor som begränsar uttryckets matchande noder
- Anges inuti [ ]
- Uttryck som blir sant eller ej tomt
  - //Film[Produktionsbolag="Tri-Star"]
  - //Film[Produktionsbolag]
  - //Film[position()=3] (den tredje filmen) Förkortning: //Film[3]
  - //Film[Skådis/@Namn="Woody Allen" or @Titel="Catwoman"]
  - //Film[@Titel eq "Catwoman"]/Skådis[@Land="USA"]
  - //Film[Skådis/@Land="USA"][Skådis/@Land="Canada"]
    samma som
    //Film[Skådis/@Land="USA" and Skådis/@Land="Canada"]
  - //Film[count(Skådis[@Land="USA"])>2]
  - //Film/Skådis[last()-1]
  - (//Film/Skådis)[last()-1]

XQuery - FLWOR

- For
  - Loopar igenom en nodsekvens (eller värdesekvens)
- Let
  - Tilldelningar
- Where
  - Villkor
- Order By
  - Sortering av resultatet
  - ascending (default) eller descending
- Return
  - Konstruktion av resultatet

XQuery

- FLWOR-uttryck kan nästlas.
- Ingen klausul är obligatorisk.
- for och let kan förekomma flera gånger i godtycklig ordning (före where-klausulen).
- XPath-uttryck kan användas i alla klausuler.
- Resultatet kan vara well-formed XML, men behöver inte vara det.
  - resultatet är det som stöds av XQuery-modellen, dvs sekvens av noder och/eller värden
- Funktionen doc() kan användas för att definiera källan (ett XML-dokument). Annars kan man använda exekveringsmiljön för att konfigurera källan.

XQuery

- Variabler börjar med $:
  - for $s in //Film/Skådis
  - let $n := $s/@Namn
- Sekvenser:
  - for $x in (1, 2, 3)
    - » samma som for $x in (1 to 3)
  - let $y := (1, 2, 3)
- Utvärdering av uttryck:
  - Lägg uttrycket inuti { }:
  - <resultat>{$x*3}</resultat>

XQuery – Computed Constructors

- element
  - element name value:

    let $a := "a", $b := 2

    return <x>{element {$a} {$b}}</x>
    - » ger <x><a>2</x>
- attribute
  - attribute name value:

    let $a := "a", $b := 2

    return <x>{attribute {$a} {$b}}</x>
    - » ger <x a="2"/>
- comment
  - comment value
  - comment {"Hej"}
    - » ger <!--Hej-->

IV1023 vt2022
nikos dimitrakas
KTH/EECS/SCS

XQuery – Computed Constructors

- processing-instruction
  - proccession-instruction name value
  - processing-instruction Säg {"Hej"}
    - » ger <?Säg Hej?>
- text
  - text value
  - text {"hej"}
    - » skapar en textnod
    - » samma resultat som "hej"
- document
  - document innehållet
  - document {comment {"Mina Filmer"}, element Filmer {…}}

XQuery – Flödeskontroll

- if-then-else

```
for $a in (1 to 5)
return if ($a mod 2 = 0)
            then <even>{$a}</even>
            else <odd>{$a}</odd>
```

```
<odd>1</odd>
<even>2</even>
<odd>3</odd>
<even>4</even>
<odd>5</odd>
```


XQuery – Kvantifierare

- some

```
for $a in //Film
where some $b in $a/Skådis/@Land satisfies string($b) = "Austria"
return  $a
```

- every

```
for $a in //Film
where every $b in $a/Skådis/@Land satisfies string($b) = "USA"
return  $a
```

XQuery – Nästlade utryck

- **Ett uttrycks resultat blir källan till ett annat uttryck:**

```
for $x in distinct-values (for $a in (1 to 6), $b in (1 to 6)
                           return <summa>{$a + $b}</summa>)
return <unik>{$x}</unik>
```

- **Ett uttrycks resultat läggs i en variabel:**

```
for $x in (1 to 5)
let $content := for $a in (1 to 5)
                return element Plus {attribute värde {$a}, $x+$a}
return element Nummer {attribute värde {$x}, $content}
```


XQuery - Namespaces

- **Kan inte nås via någon axis**
- **namespace-alias följer inte med**
- **För att jobba med namespaces och alias behöver man deklarera dem i XQuery prolog:**
  - **declare namespace alias = "URI";**
  - **declare default element namespace "URI"**

```
declare namespace aaa = "URI-1";
declare default element namespace "URI-2";
element Resultat {for $x in (1 to 5)
                  return element aaa:Nummer {attribute värde {$x}}}
```

```
<Resultat xmlns="URI-2">
  <aaa:Nummer xmlns:aaa="URI-1" värde="1"/>
  <aaa:Nummer xmlns:aaa="URI-1" värde="2"/>
  <aaa:Nummer xmlns:aaa="URI-1" värde="3"/>
  <aaa:Nummer xmlns:aaa="URI-1" värde="4"/>
  <aaa:Nummer xmlns:aaa="URI-1" värde="5"/>
</Resultat>
```

XQuery prolog

- Deklarera
  - funktioner
  - variabler
  - namespaces
  - collation
  - sortering
  - etc.

XQuery 3.0 och 3.1

- Nyheter
  - group by clause
  - klausulerna kan förekomma i mer flexibel ordning
  - switch-uttryck
  - count clause
  - try/catch-uttryck
  - dynamiska och inline-funktioner
  - computed constructor för namespace
  - flera nya funktioner, bl a för matematik
  - XQuery nodtestfunktioner formellt i XPath
  - och mycket annat

XQuery 3 - group by

- Grupperar iterationerna så att
  - return-klausulen utförs en gång per grupp
  - iterationsvariablerna blir sekvenser av alla värden i gruppens iterationer

```
for $x in 1 to 5
let $jämn := ($x mod 2) = 0
group by $jämn
return element Nummer {attribute jämn {$jämn}, $x}
```

Resultat:

```
<Nummer jämn="false">1 3 5</Nummer>
<Nummer jämn="true">2 4</Nummer>
```


XQuery 3 - group by

- group by kan gruppera på en eller flera variabler
- variablerna kan deklareras tidigare eller i group by
  - Obs! Använder man samma variabel, skriver man över dess innehåll

```
for $x in (1,1,2,3,2,1,2,3,2)
group by $x
return element Nummer {
        attribute gånger {count($x)}, $x}
```

Resultat:

```
<Nummer gånger="1">1</Nummer>
<Nummer gånger="1">2</Nummer>
<Nummer gånger="1">3</Nummer>
```

```
for $x in (1,1,2,3,2,1,2,3,2)
group by $y := $x
return element Nummer {
        attribute gånger {count($x)}, $y}
```

Resultat:

```
<Nummer gånger="3">1</Nummer>
<Nummer gånger="4">2</Nummer>
<Nummer gånger="2">3</Nummer>
```

XQuery 3 - Blandade klausuler

```
for $x in (1,1,2,3,2,1,2,3,2,4,5,3)
where $x < 5
group by $v := $x
where count($x) > 1
let $s := sum($x)
return <Nummer värde="{$v}" summa="{$s}" />
```

Resultat:

```
<Nummer värde="1" summa="3"/>
<Nummer värde="2" summa="8"/>
<Nummer värde="3" summa="9"/>
```

Börja med let eller for och avsluta med return. Fritt i övrigt.

XQuery 3 - switch

- Istället för nästlade if-then-else när man har flera fall
  - flera case
  - en default

```
for $x in 0 to 3
return element {switch ($x)
                case 0 return "Noll"
                case 1 return "En"
                default return "Många"} {$x}
```

Resultat:

```
<Noll>0</Noll>
<En>1</En>
<Många>2</Många>
<Många>3</Många>
```

XQuery 3 - count

- Räknar varven som når klausulen

```
for $x in 4 to 10
count $loop
let $nästlad := let $a := $x count $i return $i
where $x mod 2 = 0
count $rätt
return <Nummer x="{$x}" loop="{$loop}" rätt="{$rätt}" nästlad="{$nästlad}"/>
```

Resultat:

```
<Nummer x="4" loop="1" rätt="1" nästlad="1"/>
<Nummer x="6" loop="3" rätt="2" nästlad="1"/>
<Nummer x="8" loop="5" rätt="3" nästlad="1"/>
<Nummer x="10" loop="7" rätt="4" nästlad="1"/>
```


XQuery 3 - dynamiska funktioner

- Funktioner kan hanteras som värden och tilldelas variabler
- Alla funktioner kan refereras med sitt namn och sin kardinalitet
  - T ex funktionen count som tar emot en parameter är count#1

```
let $f := (upper-case#1, lower-case#1)
for $x in 1 to 4
return element Resultat {$f[$x mod 2 + 1]("STockHOLm")}
```

Resultat:

```
<Resultat>stockholm</Resultat>
<Resultat>STOCKHOLM</Resultat>
<Resultat>stockholm</Resultat>
<Resultat>STOCKHOLM</Resultat>
```

XQuery 3 - inline funktioner

- Funktioner kan definieras på plats

```
let $f := function($a) {sum(let $len := string-length(string($a))
                            for $i in 1 to $len
                            return xs:integer(substring(string($a), $i, 1))) }
for $x in (1999, 5005005, 123456789)
return element Resultat {$f($x)}
```

Resultat:

```
<Resultat>28</Resultat>
<Resultat>15</Resultat>
<Resultat>45</Resultat>
```


XQuery 3 - higher-order funktioner

- for-each (sekvens, funktion)
  - Anropar en funktion på alla medlemmar av en sekvens

```
sum(for-each(1 to 4, function($a) {$a*$a}))
```

Returnerar 30

- filter (sekvens, funktion)
  - Returnerar bara de medlemmar som uppfyller en funktion (funktionen returnerar true)

```
filter(("Maria", "Lisa", "Rebecka", "Ylva", "Evelina"),
       function($s) {contains($s, "e")})
```

Returnerar sekvensen ("Rebecka", "Evelina")

XQuery 3 - Computed Constructors

- namespace
  - namespace prefix URI
  - namespace iv1023 {"http://ns.kth.se/IV1023"}
  - namespace {""} {"http://ns.kth.se/IV1023"}
  - Namespaces skapade på det sättet kan inte användas direkt
  - Använd istället declare i XQuery prolog eller hårdkoda som xmlns-attribut

XQuery Update Facility

- Enligt XQUF 1.0:
- transform-sats
  - copy-klausul
  - modify-klausul
    - » delete-uttryck
    - » insert-uttryck
    - » rename-uttryck
    - » replace-uttryck
  - return-klausul
- Stöd för nästlade FLWOR
  - som returnerar uppdateringsuttryck
  - nyckelord för insert
    - » node eller nodes
    - » before
    - » after
    - » as first into
    - » as last into
    - » into
  - nyckelord för delete
    - node eller nodes
  - nyckelord för replace
    - » (value of) node … with …
  - » nyckelord för rename
    - » node … as

XQuery - transform

- copy
  - skapa en kopia av ett XML-dokument
- modify
  - ett eller flera uttryck som uppdaterar/förändrar kopian
- return
  - oftast kopian, men kan även konstruera något annat

XQuery - transform - insert

```
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>
```

XQuery - transform - insert

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

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


XQuery - transform - insert

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

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

XQuery - transform - delete

```
copy $x := <root><a>456</a><a>789</a></root>
modify delete node $x/a[text() = 789]
return $x
```

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


XQuery - transform - delete

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

```
<root/>
```

XQuery - transform - replace

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

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


XQuery - transform - replace

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

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

XQuery - transform - replace

```
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>
```


XQuery - transform - rename

```
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>
```

XQuery - transform – flera, loop

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

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


XQuery - transform – flera, lista

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

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

XQuery - transform – flera, lista

```
copy $x := <root><a>456</a><a>789</a></root>
modify (rename node $x/a[1] as "b",
        insert node attribute f {"200"} into $x/a[1],
        replace value of node $x/a[1] with "123")
return $x
```

```
<root>
        <b f="200">123</b>
        <a>789</a>
</root>
```

Obs! Elementet b finns inte förrän efter att hela modify-klausulen är klar.

XQuery Update Facility

- Enligt XQUF 3.0:
- Copy-modify-sats
  - Som tidigare:
  - copy-klausul
  - modify-klausul
  - return-klausul
- Transform-with-sats
  - … transform with {uttryck}
  - Uttrycket kan vara
  - insert
  - delete
  - replace
  - rename
- Uttryck i return
  - insert
  - delete
  - replace
  - rename
- "Updating expressions"
  - Modifierar den aktuella noden

XQuery – transform with

```
<root><a>456</a><a>789</a></root>
transform with {insert node <a>123</a> as first into .}
```

Har samma effekt som

```
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>
```


XQuery – transform with

```
let $x := <root><a>456</a><a>789</a></root>
return $x transform with {
        insert node <a>123</a> as first into .,
        rename node ./a[2] as "b"
}
```

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

Obs! Det andra a baseras på originalet då insert-modifikationen inte har införts ännu.

XQuery – Updating expressions

```
rename node //Film[1] as "Movie"
```

Det berörda elementet ändras och uttrycket returnerar inget.

```
for $f in //Film
return rename node $f as "Movie"
```

Alla Film ändras till Movie och inget returneras.

Egentligen returneras en eller flera "pending updates", men de är inget man ser.

XQuery Update Facility - Stöd

- Stöd finns i BaseX
  - Enligt XQuery Update Facility 3.0
  - Copy-modify
  - Transform-with
  - Updating expressions
  - Ändrar inte filerna som default. Ändrar kopian i minnet.
- Stöd finns i DB2 och Oracle (från version 12)
  - Enligt XQuery Update Facility 1.0
    - » Copy-modify
  - Vissa syntaktiska skillnader
- Stöds i några få andra produkter

Fortsättning

- Quiz om XQuery & XPath
- Labb om XQuery (komp. Querying XML Data with XQuery)
- Lektionsuppgifter
- Seminarieuppgifter (Inlupp 1)