XQuery

HOHDE

- XQuery
  - uudet funktiot
  - sekvenssit
  - muuttujat
- Iterointi
  - järjestys
  - suodatus
  - järjestäminen
- Ehtorakenteet
- Muita toimintoja

XQuery

- XQuery on helposti ymmärrettävä kyselykieli, jolla voidaan poimia tietoa XML-dokumenteista
  - XQuery on XPath 2.0 –suosituksen laajennus
    - tarkalleen ottaen molemmat suositukset ovat vielä kesken
    - XQuery käyttää XPathin funktioita ja pystyy hyödyntämään myös XPath-lausekkeita
  - tyypitykseltään XQuery tukeutuu XML Schemaan
- XQueryyn on laadittu sekä ihmisille että koneille sopivat syntaksit
  - tässä käsitellään ihmisille sopivaa
  - koneille sopivassa syntaksissa XQuery-lausekkeet ilmaistaan XML:na

Uudet funktiot

- XPath 2.0 määrittelee joukon funktioita, esimerkiksi:
  - doc(URI) – palauttaa URI:n perusteella haettavan dokumentin solmujoukon (node-set)
  - empty(ns) – tarkistaa, onko annettu joukko tyhjä
  - max(ns) – palauttaa suurimman annetusta joukosta
  - min(ns) – palauttaa pienimmän annetusta joukosta
- Näiden lisäksi määritellään suuri joukko ajan käsittelyyn liittyviä funktioita
- Muihinkin tyyppeihin liittyviä funktioita on lisätty

Sekvenssit

- XQuery pohjautuu pitkälti sekvenssien käsittelyyn
  - sekvenssit voidaan ymmärtää järjestetyksi oliojoukoksi
  - XPath-lausekkeet ja funktiot palauttavat solmujoukon sekvenssinä
  - sekvenssi määritellään kirjoittamalla sen sisältämät kohteet sulkujen sisään

```
(1, 2, 3)
```

  - sekvenssit voivat olla sisäkkäisiä

```
(1, (2, 3), 4)
```

  - tyhjä sekvenssi merkitään ()

Sekvenssien luonti

- Sekvenssejä voidaan luoda myös määrittelemällä sarja

```
(1 to 5) -> (1, 2, 3, 4, 5)

reverse(1 to 5) -> (5, 4, 3, 2, 1)
```

- Mikäli sekvenssiin sijoitetaan XPath-lausekkeita, sekvenssin alkioita tulee lausekkeiden solmujoukot

Sekvenssien yhdistäminen

- Sekvenssejä voidaan yhdistää seuraavilla operaatioilla
  - union - palauttaa sekvenssin, jossa on kaikki alkiot, jotka ovat jommassa kummassa yhdistettävässä sekvenssissä

    ```
    (a, b) union (b, c) -> (a, b, c)
    ```

  - intersect – palauttaa sekvenssin, jossa on kaikki alkiot, jotka ovat molemmissa yhdistettävissä sekvenseissä

    ```
    (a, b) intersect (b, c) -> (b)
    ```

  - except – palauttaa sekvenssin, jossa on kaikki alkiot, jotka ovat ensimmäisessä, mutta eivät ole toisessa yhdistettävässä sekvenssissä

    ```
    (a, b) except (b, c) -> (a)
    ```

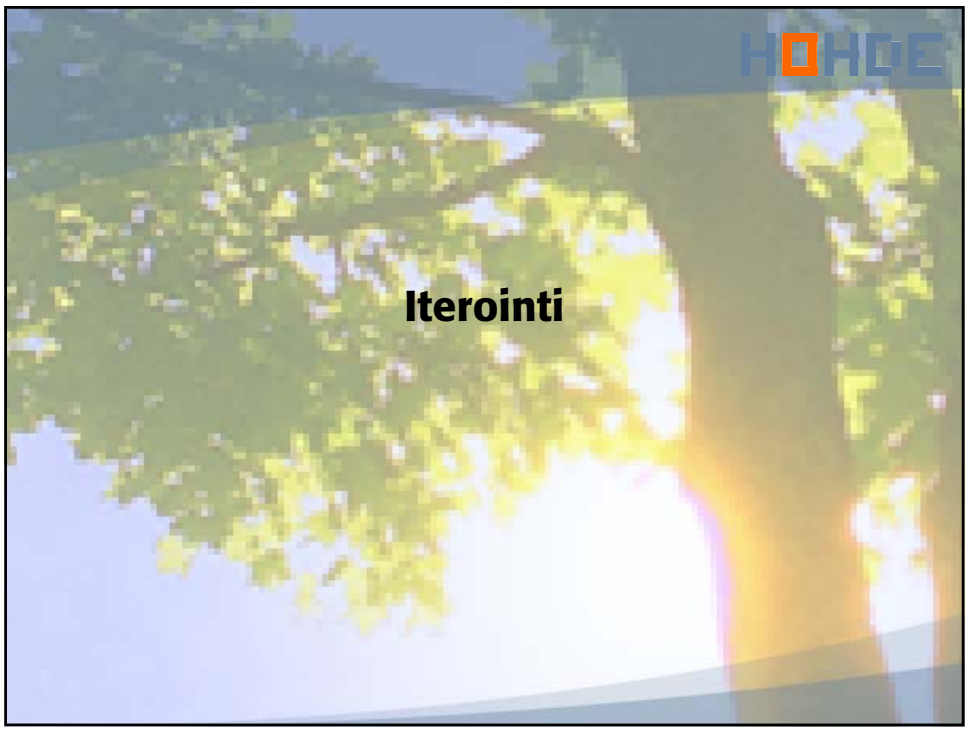

Muuttujat

- XQuery mahdollistaa muuttujien määrittelyn ja käytön
  - muuttujat ovat tyypitettyjä
    - sama tyyppisysteemi käytössä kuin XPathissa (boolean, number, string, node-set)
- Muuttujaan viitataan merkinnällä $nimi
- Muuttuja asetetaan lausekkeella let

```
let $s := "testi"
```

  - tämän jälkeen muuttuja on käytettävissä
  - muuttuja voi sisältää myös sekvenssin

```
let $d := doc('testi.xml')
```



Iterointi

- XQueryn yleisin käyttötapa on käydä tietty solmujoukko läpi
  - solmujoukon sisältä poimitaan arvoja, jotka palautetaan kyselyn vastauksessa
- Solmujoukko käydään läpi lausekkeella for:

```
for $p in doc('testi.xml')//para
```

  - yllä muuttujaan $p sijoitetaan dokumentin 'testi.xml' jokainen para-elementti, yksi kerrallaan

Iterointi useassa tasossa

- For-lauseke voi yhdistää useita iterointeja, esimerkiksi

```
for $i in (1,2),
    $j in (3,4)
```

  - tällöin iteroidaan seuraavasti:

```
$i = 1, $j =3
$i = 1, $j = 4
$i = 2, $j = 3
$i = 2, $j = 4
```


Iteroinnin järjestys

- Iterointi voidaan suorittaa joko järjestettynä (ordered) tai ei-järjestettynä (unordered)
- Järjestetyssä iteroinnissa alkiot käydään läpi aina dokumenttijärjestyksessä
  - ei-järjestetyssä iteroinnissa XQuery-sovellus voi säätää järjestyksen haluamakseen
    - mahdollistaa kyselyjen optimoinnin

```
unordered {
  for $p in doc('testi.xml')//para,
      $t in doc('reviews.xml')//review
  ...
}
```

Sekvenssin järjestys

- Myös sekvenssin järjestys voidaan vapauttaa
  - tällöin XQuery-prosessori saa päättää sekvenssin alkioiden käsittelyjärjestyksen

```
let $p := unordered(doc('testi.xml')//para)
```


Järjestysmuuttujat

- For-lausekkeessa voidaan määritellä ylimääräinen järjestysmuuttuja (positional variable), joka kertoo iteraatiokierrosten lukumäärän:

```
for $p at $i in doc('testi.xml')//para
```

  - tässä muuttuja $i sisältää kappaleen järjestysnumeron
- Järjestysmuuttujia on mahdollista käyttää jokaisessa iteraatiotasossa

Suodatus

- for- ja let-lauseiden toimintaa voidaan ohjata lisäämällä lauseeseen suodatus
  - tällöin käsiteltävästä joukosta poimitetaan vain ehdot täyttävät kohteet
  - suodatus tapahtuu where-lausekkeella:

```
for $p in doc('testi.xml')//para
where $p[@level='1']
```

    - yllä valitaan ainoastaan ne para-elementit, joiden title-attribuutin arvo on 1
    - sama olisi tosin saatu aikaan pelkällä XPath-lausekkeellakin

```
for $p in doc('testi.xml')//para[@level='1']
```


Unioni

- Suodatuksen perinteisimpiä käyttöjä on unionin muodostaminen
  - luetaan kaksi dokumenttia sisään ja poimitaan niistä tietyllä tavalla samanlaiset elementit
  - näistä elementeistä koostetaan vastaus, yhdistellen tietoa molemmista dokumenteista:

```
for $i in doc('testi.xml')//para,
    $j in doc('toimittajat.xml')//editor
where $i/@author=$j/@editor
```

Järjestäminen

- Iteroitavat kohteet voidaan järjestää yhdellä tai useammalla järjestysehdolla
  - järjestämistä säädetään lausekkeella order by:

```
for $p in doc('testi.xml')//para
order by $para/@title
```

  - järjestyksen suunta voidaan kääntää lisäämällä loppuun avainsana 'descending'
  - jos halutaan, että yhtä suurten kohteiden järjestys ei muutu, lausekkeen alkuun täytyy lisätä avainsana 'stable'

```
for $p in doc('testi.xml')//para
stable order by $para/author descending
```


Vastauksen palauttaminen

- Vastaus palautetaan return-lausekkeella
  - lauseke voi sisältää XML-koodia, joka liitetään mukaan vastaukseen
  - mahdolliset XQuery-lausekkeet täytyy kirjoittaa aaltosulkuihin { }

```
for $i in doc('testi.xml')//para
return
  <editor>
    {$i/@author}
  </editor>
```

Elementtien luonti

- Elementtejä voidaan luoda myös dynaamisesti:

```
element $author-el {
  element first { "Janne" }
  element last { "Kalliola" }
}
```

- Samoin muut XML:n rakennepalaset voidaan luoda dynaamisesti (`comment`, `text`, `processing-instruction`, yms.)

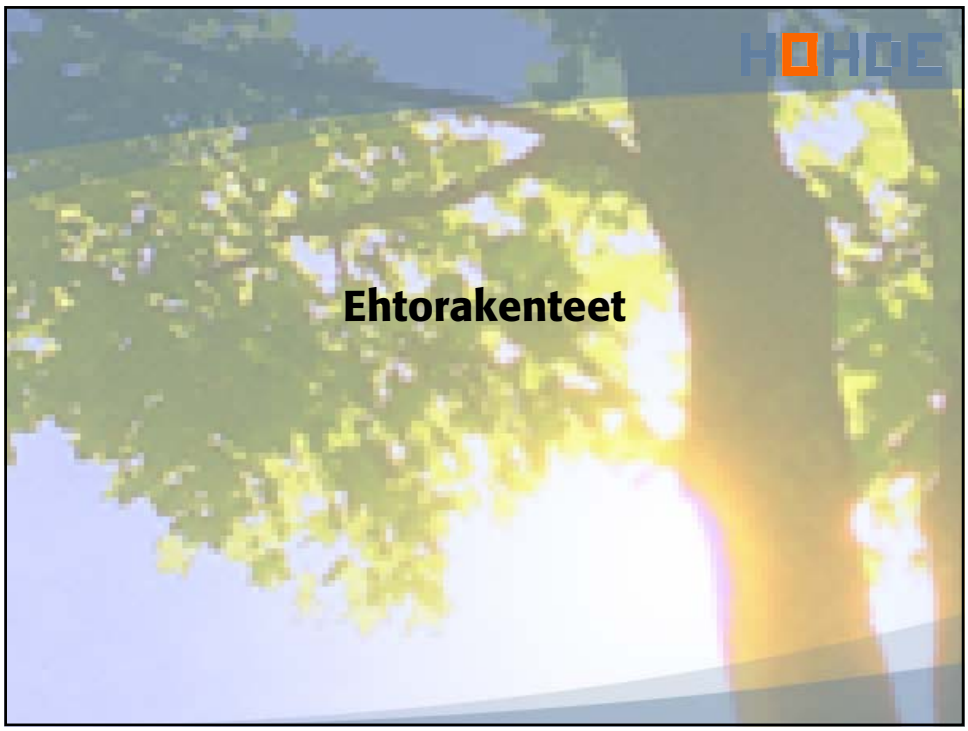
Attribuuttien luonti

- Vastaukseen voidaan myös luoda attribuutteja
  - käytetään element ja attribute-lausekkeita:

```
return
  {
    element editor {
      attribute name { $p/@author }
    }
  }
```

  - tai kirjoitetaan attribuutit suoraan XML-koodiin

```
return
  <editor name="{$p/@author}"/>
```



Ehtolauseet

- XQueryssa on if – then – else –rakenne:

```
if ($value)
  then $true
  else $false
```

  - kuten muissakin ohjelmointikielissä, vain toinen haaroista evaluoidaan
  - molemmat haarat ovat pakollisia
    - jos ei haluta tehdä toimenpiteitä, käytetään tyhjää sekvenssiä ()

Case-lauseke

- Myös case-lauseke on määritelty:

```
typeswitch($customer/billingaddress)
  case $a as element(*, USAddress)
    return $a/state
  case $a as element(*, CanadaAddress)
    return $a/province
  default return "unknown"
```

  - vertailu pohjautuu ainoastaan kohteiden tyyppeihin

Määreiset lausekkeet

- XQuery mahdollistaa testien suorittamisen solmujoukolle määreisillä lausekkeilla (quantified expression)
  - määreinen lauseke palauttaa aina totuusarvon:

```
every $p in //para satisfies $p/@author
```

    - palauttaa true, jos jokaisella para-elementillä on attribuutti author

```
some $p in //para satisfies $p/author='Kalliola'
```

    - palauttaa true, jos yksikin para-elementin author-attribuutin arvo on 'Kalliola'

Iterointi määreisillä lausekkeilla

- Määreinen lauseke voi saada käsiteltäväkseen useampia solmujoukkoja
  - ehto liittyy tällöin molempien joukkojen kohteisiin
  - kaikki kombinaatiot käydään läpi

```
some $x in (1, 2, 3), $y in (3, 2, 1)
satisfies $i + $j = 4
```
-> true

```
every $x in (1, 2, 3), $y in (3, 2, 1)
satisfies $i + $j = 4
```
-> false

Erillisyys

- Sekvenssiä luotaessa voidaan vaatia, että sekvenssissä ei saa olla duplikaatteja
  - käytetään funktiota distinct-values:

    for $i in distinct-value(doc('testi.xml')//para/@author)

Funktiot

- Käyttäjä voi määrittää omia funktioita
  - funktioiden argumenttien ja paluuarvon tyypit täytyy määrittää

```
declare function local:depth($e as node())
as xs:integer
{
  if(empty($e/*))
  then 1
  else max(for $c in $e return local:depth($c)) + 1
};
```

- Funktioita kutsutaan samalla tavalla kuin kielen omia funktioita

```
local:depth(doc('testi.xml'))
```

XQueryn liittäminen XML-dokumenttiin

- XQuery-lausekkeet voidaan sijoittaa XML-dokumentin sisään
  - tämä XML-dokumentti toimii syötteenä XQuery-sovellukselle
  - sovellus suorittaa XQuery-lausekkeet ja korvaa ne niiden paluuarvoilla
  - XQuery-lausekkeet sijoitetaan aaltosulkuihin { } (vrt. return):

```
<authors>
  { for $p in doc('testi.xml')//para
    return
      <author>
        { $p/@author
      </author>
  }
</authors>
```


Dokumenttien luonti

- XQuery-kysely voi myös luoda XML-dokumentin
  - tällöin kysely ei sijaitse XML-dokumentin sisässä

```
document
  {
    <paragraphs>
      { doc('testi.xml')//para }
    </paragraphs>
  }
```

  - huomaa, että esimerkkirakenteessa ei ole erityistä return-lausetta
    - mikään ei toki estä sellaisen käyttöä

Kommentit

- XQueryn kommentit kirjoitetaan merkkien (: ja :) väliin
  - kommentit jätetään huomiotta XQuery-lausekkeita suoritettaessa
  - XML-kommentit sen sijaan kopioidaan tuotettavaan dokumenttiin