

XQuery

XQuery

- standardni upitni jezik za XML
- W3C standard
XQuery 1.0: An XML Query Language
<http://www.w3.org/TR/xquery/>
- dugotrajan razvoj (1998-2007)
- ekvivalent SQL-a za XML dokumente

XQuery tipovi podataka

- svi ugrađeni XML Schema tipovi
- još 7 tipova vezanih za tipove čvorova u stablu dokumenta
- još 6 tipova specifičnih za XQuery

XQuery tipovi podataka

- svaka XQuery vrednost je sekvenca koja sadrži 0 ili više elemenata
- element sekvence je *singleton* sekvenca dužine 1, koja sadrži baš taj element
 - $(1) = 1$
- sekvenca može biti prazna
 - $()$
- ali ne može sadržati druge sekvence
 - sekvence se „poravnavaju“
 - $(0, (), (1, 2)) = (0, 1, 2)$

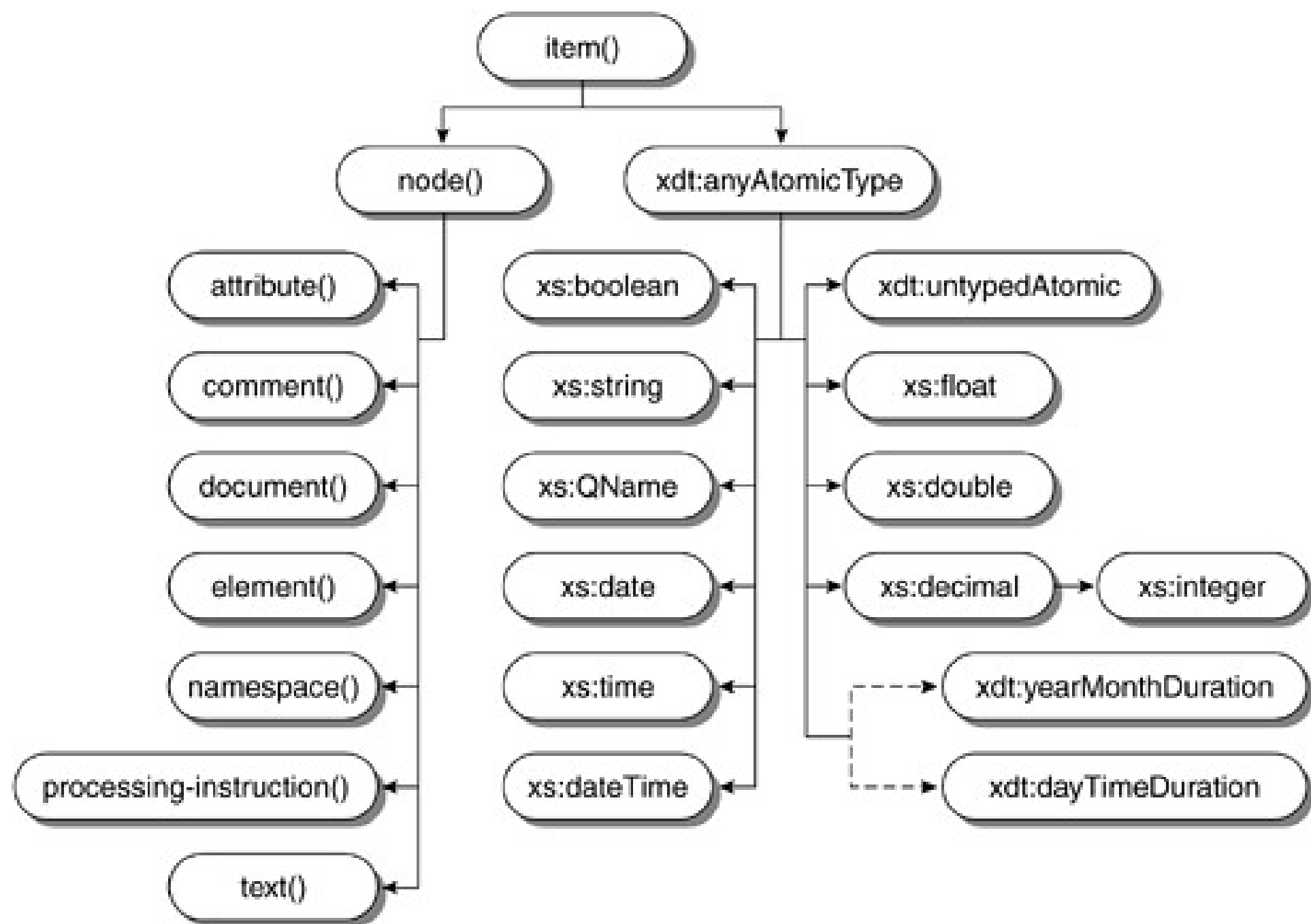
XQuery tipovi podataka

- svaki *singleton* ima svoj tip izveden iz `item()`
 - slično kao `java.lang.Object`, ali je apstraktan - ne može se instancirati
 - piše se sa zagradama da bi se razlikovao od korisničkih tipova istog imena
 - da bude nalik XPath testu čvora

XQuery tipovi podataka

- postoji dve vrste `item()`-a
 - XML čvorovi
 - nasleđuju `node()`
 - atomičke vrednosti
 - nasleđuju `xdt:anyAtomicType`

XQuery tipovi podataka



XQuery tipovi podataka

- sekvenca ima tip koji se sastoji od
 - imena tipa ili `empty()` - prazna sekvenca,
 - (opciono) indikatora ponavljanja
 - * - 0 ili više
 - + - 1 ili više
 - ? - 0 ili 1
- primeri
 - `item()` - sekvenca koja sadrži jedan element bilo kog tipa
 - `item()*` - sekvenca koja sadrži 0..* elemenata bilo kog tipa
 - `xsd:integer*` - sekvenca koja sadrži 0..* celih brojeva
 - `xsd:integer*` je ujedno i `item()*`

XQuery tipovi podataka

- svaki XQuery izraz ima
 - statički tip (*compile-time*)
 - dinamički tip (*run-time*)
 - tip dobijenog rezultata
 - vrednost rezultata je instanca tog tipa
- proveru se može vršiti *compile-time* i *run-time*
- primer dinamičke provere:
`12 + if ($foo) then 30 else "0"`

(: ako je \$foo=true izraz je ispravan,
inače nije ispravan :)

Whitespace i komentari

- whitespace znakovi su
 - U+0020 (space),
 - U+0009 (tab),
 - U+000D (carriage return),
 - U+000A (line feed)
- komentar može da se pojavi bilo gde na mestu whitespace znakova
- komentar se navodi između (: . . . :)

Whitespace i komentari

- primer

```
(: Komentar. :)  
let $i := 42 (: Komentar. :)  
return <x>(: Nije komentar. :)</x>
```
- rezultat

```
<x>(: Nije komentar. :)</x>
```

Konstante

- prazna sekvenca: `()`
- logičke (`xs:boolean`): `true()`, `false()`
- stringovi (`xs:string`): `"hello"`, `'world'`
- celi brojevi (`xs:integer`): `42`
- brojevi u fiksnom zarezu (`xs:decimal`): `42.`, `4.2`, `.42`
- brojevi u pokretnom zarezu (`xs:double`): `42E0`, `4.2e+0`, `42E-2`
- drugi tipovi: `xs:float("1.25")`, `xs:ID("X1")`

Primer XQuery upita

```
(: Primer 1 :)
declare namespace my = "urn:foo";
declare function my:fact($n) {
  if ($n < 2)
    then 1
    else $n * my:fact($n - 1)
};
declare variable $my:ten { my:fact(10) };

<table> {
  for $i in 1 to 10
  return
    <tr>
      <td>10!/{$i}! = {$my:ten div my:fact($i)}</td>
    </tr>
}</table>
```

The diagram illustrates the structure of the XQuery query. Callouts in blue boxes point to specific parts of the query:

- komentar** points to the comment `(: Primer 1 :)`.
- deklaracija namespace-a** points to `declare namespace my = "urn:foo";`.
- deklaracija funkcije** points to the function definition `declare function my:fact($n) { ... }`.
- globalna promenljiva** points to `declare variable $my:ten { my:fact(10) };`.
- konstruisani XML** points to the opening tag `<table> {`.
- FLWR izraz** points to the `for $i in 1 to 10` clause.
- ugrađeni izraz** points to the expression `my:fact($i)` inside the `<td>` tag.

Brackets on the right side group the query into two sections:

- prolog** groups the first four lines (comment, namespace declaration, function declaration, and variable declaration).
- body** groups the last three lines (XML opening tag, FLWR expression, and XML closing tag).

XQuery prolog

- nije obavezan
- definiše kontekst za upit (compile-time)
 - namespaces
 - korisničke funkcije
 - importovani šema tipovi
 - importovani moduli
 - promenljive

XQuery prolog

- deklaracije namespace-ova u prologu

```
declare namespace x = "http://www.foo.com";
```

```
<x:foo/>
```

Korisničke funkcije

- deklaracija korisničke funkcije

ime
funkcije

parametar

tip parametra
(opciono)

tip rezultata
(opciono)

```
declare function my:fact($n as xs:integer) as xs:integer
{
    if ($n < 2)
        then 1
        else $n * my:fact($n - 1)
};
```


XML sadržaj

- rezultat XQuery upita može biti
 - atomička vrednost
 - XML sadržaj
 - XQuery se često koristi za generisanje XML dokumenata
 - slično kao u SQL-u (rezultat upita je relacija)
- XML node constructor
`<hello>world</hello>`

XML sadržaj

- dinamičko generisanje sadržaja: XQuery izrazi unutar { . . . }

- primer

```
<x y="2*2 = {2*2}">
```

```
Velika istina: 2*2={2*2}.
```

```
</x>
```

- rezultat

```
<x y="2*2 = 4">
```

```
Velika istina: 2*2=4.
```

```
</x>
```

XML sadržaj

- navođenje vitičastih zagrada: $\{\{ i \}\}$
- primer
 $\langle \text{add} \rangle$
 $\{\{ 1 + 1 = \{ 1+1 \} \}\}$
 $\langle / \text{add} \rangle$
- rezultat
 $\langle \text{add} \rangle \{ 1 + 1 = 2 \} \langle / \text{add} \rangle$

XML sadržaj

- alternativni način za konstrukciju elemenata:
element {"ime"} {"sadržaj"} → <ime>sadržaj</ime>
- konstruktori za različite tipove čvorova
document {
 element foo {
 attribute bar { 1 + 1 }
 text { "trt" }
 <x xmlns='urn:x'>Može i da se meša</x>
 }
}
↓
<foo bar="2">trt<x xmlns='urn:x'>Može i da se meša</x></foo>

XML sadržaj

- sekvence se poravnaju pre ugrađivanja u XML

`<x y="{ () }">{ (1, 2) }</x>`

↓

`<x y="">1 2</x>`

Operatori

- zarez: definiše sekvencu
1, "trt"
 - zarez ima najniži prioritet, pa se sekvence često smeštaju u zagrade
- zagrade: mogu da grupišu izraze različitih tipova
 - $() \rightarrow ()$
 - $(1, 2) \rightarrow (1, 2)$
 - $1 + 2 * 3 \rightarrow 7$
 - $(1 + 2) * 3 \rightarrow 9$

Operatori

- logički operatori
 - and
 - or
 - not () ... piše se kao funkcija zbog kompatibilnosti sa XPath
- if-then-else operator
 - else je obavezan
 - if (true()) then "true" else "false"

Operatori

- aritmetički
 - binarni: +, -, *, `div`, `idiv`, `mod`
 - unarni: +, -
- ima i aritmetičkih funkcija
 - `min((2, 1, 3, -100))`
 - `round(9 div 2)`
 - `round-half-to-even(9 div 2)`

Operatori

- poređenje
 - poređenje vrednosti: za poređenje dva singletona
 - eq, ne, gt, ge, lt, le
 - generalno poređenje: za poređenje dve sekvence
 - vraćaju true() ako u obe sekvence postoji bar po jedan element za koje poređenje po vrednosti vraća true()
 - $(1, 2, 3) = 4 \rightarrow \text{false}$
 - $(1, 2, 3) = 3 \rightarrow \text{true}$
 - $(1, 2) = (3, 4) \rightarrow \text{false}$
 - $(1, 2) \neq (3, 4) \rightarrow \text{true}$
 - $(1, 2) = (2, 3) \rightarrow \text{true}$
 - $(1, 2) \neq (2, 3) \rightarrow \text{true}$
 - $(1, 2) \neq (1, 2) \rightarrow \text{true}$

Operatori

- poređenje
 - poređenje čvorova: operišu nad sekvencama čvorova
 - << - „before“: vraća true ako je levi čvor ispred/pre desnog u dokumentu
 - >> - „after“: vraća true ako je levi čvor iza/posle desnog u dokumentu
 - is - vraća true ako su čvorovi isti (po identitetu)
 - isnot - negacija od is
 - primer:

<code><a/> is </code>	<code>→ false</code>
<code><a/> isnot <a/></code>	<code>→ true</code>
<code>x/.. << x</code>	<code>→ true</code>

Operatori

- funkcije za poređenje
 - `compare()`: poredi dve atomičke vrednosti
 - `deep-equal()`: poredi cele sekvence, sa dubokim poređenjem svih elemenata

Ugrađene funkcije

- 110 ugrađenih funkcija
- razlikuju se po imenu i listi parametara
- namespace: <http://www.w3.org/2003/11/xpath-functions>
 - uobičajeni prefiks: fn
- primer 1: broj elemenata sekvence
 $\text{fn:count}(\text{"a"}, 2, \text{"c"}) \rightarrow 3$
- primer 2: podsekvenca
 $\text{fn:subsequence}((-5, 4, -3, 2, -1), 2, 3) \rightarrow (4, -3, 2)$

Ugrađene funkcije

- funkcije nad sekvencama
 - `count()`: dužina sekvence
 - `distinct-values()`: ukloni sve duplikate
 - `empty()`: da li je sekvenca prazna
 - `exists()`: da li sekvenca nije prazna
 - `index-of()`: položaj elementa u sekvenci
 - `insert-before()`: ubaci element u sekvencu
 - `remove()`: ukloni element iz sekvence
 - `reverse()`: obrni redosled sekvence
 - `subsequence()`: izdvoj podsekvencu
 - `unordered()`: naglasi da redosled nije važan

Ugrađene funkcije

- aritmetičke funkcije
 - floor()
 - ceiling()
 - abs()
 - min()
 - max()
 - avg()
 - sum()
 - round()
 - round-half-to-even()

Sintaksni biseri

- znak - je validan znak za ime
 - $a-1$ nije isto što i $a - 1$
- znak / je separator koraka u XPath putanji
 - a/b nije isto što i $a \text{ div } b$

Sintaksni biseri

- `not(=)` i `!=`
 - operatori `=` i `!=` proveravaju egzistenciju
 - operatori `eq` i `ne` ne proveravaju egzistenciju
 - `not(a=b)` nije isto što i `a!=b`
 - prva varijanta negira i test egzistencije i test jednakosti
 - primer

```
x[@y = 1]
  (: nalazi <x y="1"/> ali ne i <x/> i <x y="2"/> :)
```

```
x[not(@y=1)]
  (: nalazi <x y="2"/> i <x/> ali ne i <x y="1"/> :)
```

```
x[@y != 1]
  (: nalazi <x y="2"/> ali ne i <x/> i <x y="1"/> :)
```


Sintaksni biseri

- aritmetički operatori nisu (uvek) asocijativni

- primer dokumenta:

```
<x>  
  <y>2</y>  
  <y>3</y>  
</x>
```

- primer dva različita izraza

```
/x[1 + 2 = y]
```

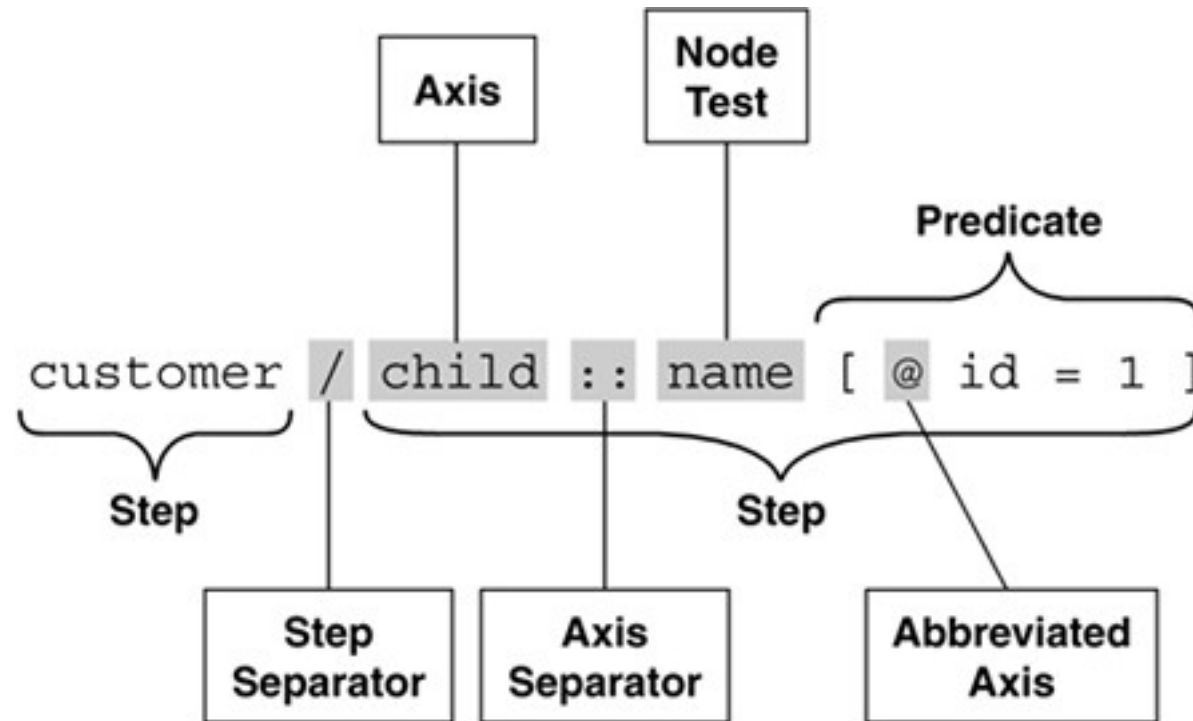
(: pronalazi x sa detetom y jednakim 3 :)

```
/x[ 2 = y - 1]
```

(: greška: y-1 nije atomička vrednost :)

Putanje

- koristi se neznatno izmenjen XPath
 - struktura elementa putanje je ista



Putanje

- funkcije za navigaciju
 - `collection()`: imenovana sekvenca
 - `doc()`: koren datog XML dokumenta
 - `id()`: element sa datim ID-jem
 - `idref()`: elementi koji pokazuju na dati ID
 - `root()`: koren tekućeg dokumenta
 - primeri:
`doc("team.xml")/Team/Player[Name/Last="Divac"]`
`collection("teams")//Player[Name/Last="Divac"]`

Promenljive

- navode se sa znakom \$ ispred imena
- ime može biti nekvalifikovano ili kvalifikovano
 - prefiks zamenjuje namespace
- promenljive nisu promenljive
tj. ne može im se promeniti vrednost

FLWOR izrazi

- centralni izraz u XQuery
- čita se „flower“
- po prvom slovu klauzula: for, let, where, order by, return
- više namena
 - za definisanje promenljivih
 - za iteraciju kroz sekvencu
 - za filtriranje rezultata
 - za sortiranje sekvenci
 - za spajanje različitih izvora podataka

FLWOR izrazi

- primer: koristi svih pet klauzula

```
for $i in doc("orders.xml")//Customer
let $name := concat($i/@FirstName, $i/@LastName)
where $i/@ZipCode = 91126
order by $i/@LastName
return
    <Customer Name="{ $name }">
        { $i//Order }
    </Customer>
```

FLWOR izrazi

- `for` i `let` mogu da se pojave u bilo kom redosledu,
- i u bilo kom broju,
- sve dok postoji bar jedna `for` ili `let` klauzula
- `for` iterira kroz sekvencu
- `let` dodeljuje promenljivoj vrednost datog izraza
 - promenljive se nalaze u opsegu do kraja FLWOR izraza
- `where` filtrira
- `order by` sortira
- `return` konstruiše rezultat

FLWOR primeri

```
let $variable := "any expression here"  
return concat("xx", $variable, "xx")
```

=>

```
"xxany expression herexx"
```


FLWOR primeri

```
for $i in (1, 2, 3, 4, 5)  
where $i > 3  
return $i
```

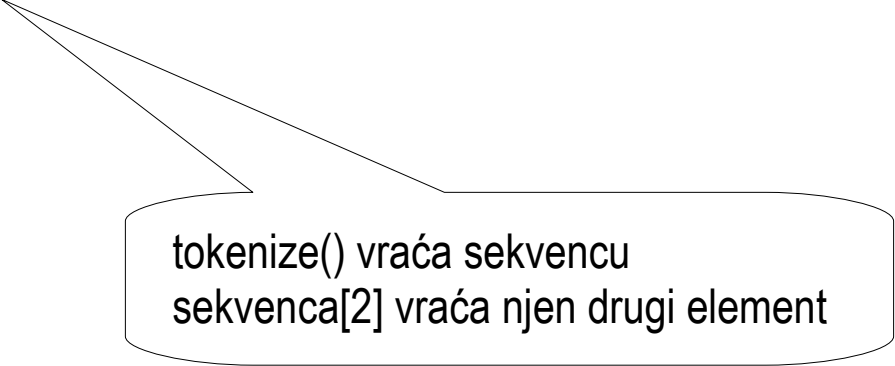
=>

(4, 5)

(: kraći zapis pomoću putanje :)
(1,2,3,4,5)[. > 3]

FLWOR primeri

```
for $e in doc("team.xml")//Employee  
let $name := $e/Name  
order by tokenize($name)[2] (: Prezime :)  
return $name
```



tokenize() vraća sekvencu
sekvenca[2] vraća njen drugi element

FLWOR primeri

```
for $i in doc("one.xml")//fish,  
    $j in doc("two.xml")//fish  
where $i/red = $j/blue  
return <fishes> { $i, $j } </fishes>
```

može i iz više dokumenata

Kvantifikacija

- operatori `some` i `every`
 - skraćeni FLWOR
 - vraćaju logičku vrednost
 - postoji
 - za svaki

```
some $emp in doc("team.xml")//Employee satisfies $emp/@years > 5  
every $emp in doc("temp.xml")//Employee satisfies $emp/@years > 5
```

FLWOR join

1. Dekartov proizvod

```
for $i in (1, 2, 3)
for $j in (3, 4, 5)
return ($i, $j)
```

=>

(1, 3, 1, 4, 1, 5, 2, 3, 2, 4, 2, 5, 3, 3, 3, 4, 3, 5)

FLWOR join

2. inner join

```
for $i in (1, 2, 3)
for $j in (3, 4, 5)
where $i = $j
return ($i, $j)
```

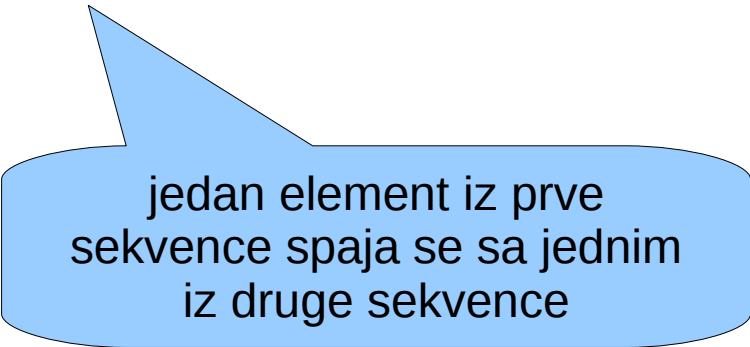
=>

(3, 3)

FLWOR join

2. inner join / one-to-one

```
for $proj in doc("projects.xml")/Projects/Project
for $emp in doc("team.xml")//Employee
where $proj/@owner = $emp/@id
return $proj/Name, $emp/Name
```




jedan element iz prve
sekvence spaja se sa jednim
iz druge sekvence

FLWOR join

2. inner join / one-to-one

```
for $proj in doc("projects.xml")/Projects/Project
for $emp in doc("team.xml")//Employee
where $proj/@owner = $emp/@id
return <Assignment>{$proj/Name,$emp/Name}</Assignment>
```

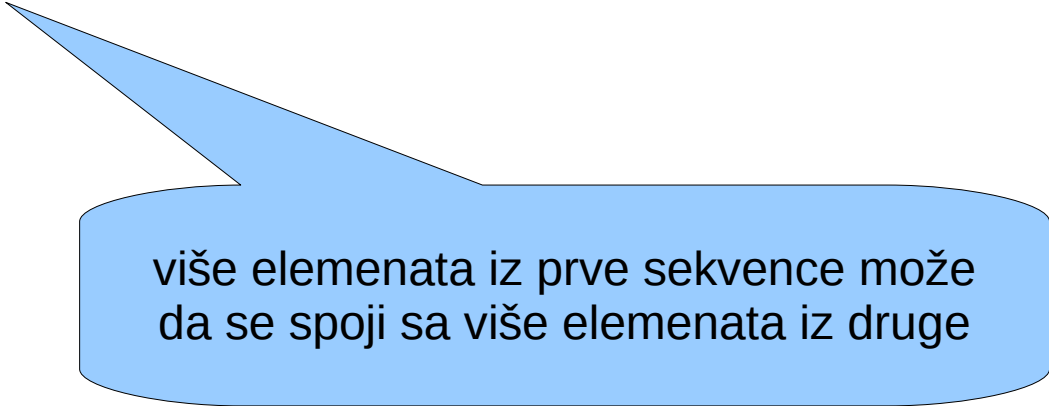


grupiřemo rezultat u novi
element

FLWOR join

3. inner join / many-to-many

```
for $proj in doc("projects.xml")/Projects/Project
for $emp in doc("team.xml")//Employee
where $proj/Category = $emp/Expertise
return
<Assignment proj="{ $proj/Name}" emp="{ $emp/Name}" />
```



više elemenata iz prve sekvence može
da se spoji sa više elemenata iz druge

FLWOR join

3. outer join / left outer join

```
for $proj in doc("projects.xml")/Projects/Project
return
  <Assignment proj="{ $proj/Name}">{
    for $emp in doc("team.xml")//Employee
    where $emp/Expertise = $proj/Category
    return $emp/Name
  }</Assignment>
```

imena
zaposlenih se
mogu ponavljati

rezultat sadrži po jednu stavku za svaki
element prve (leve) sekvence, čak i ako
nema odgovarajućeg elementa u drugoj
sekvenci

FLWOR join

3. outer join / left outer join

```
for $proj in doc("projects.xml")/Projects/Project
```

```
return
```

```
<Assignment proj="{ $proj/Name}" emps="{  
  for $emp in doc('team.xml')//Employee  
  where $emp/Expertise = $proj/Category  
  return $emp/@id
```

```
}" />
```

emps je tipa
IDREFS, nema
ponavljanja imena
zaposlenih!

rezultat sadrži po jednu stavku za svaki
element prve (leve) sekvence, čak i ako
nema odgovarajućeg elementa u drugoj
sekvenci

FLWOR join

3. outer join

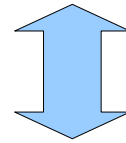
```
for $proj in doc("projects.xml")/Projects/Project
for $emp in doc("team.xml")//Employee
where $proj/@owner = $emp/@id
  and not($proj/Category = $emp/Expertise)
return <Mismatch>{$emp/Name, $proj/Name}</Mismatch>
```

svi projekti gde vlasnik nema ekspertizu u toj kategoriji

FLWOR join

4. self join

```
doc("team.xml")//Employee[Title =  
    doc("team.xml")//Employee[@id="E0"]/Title]/Name
```



```
let $emp := doc("team.xml")//Employee  
for $i in $emp, $j in $emp  
where $i/Title = $j/Title and $j/@id = "E0"  
return $i/Name
```

Poređenje sekvenci

- egzistencijalno poređenje
 - $\$seq1 > \$seq2$
 - da li postoji bar jedan element u $\$seq1$ veći od bilo kog elementa u $\$seq2$
 - `some $a in $seq1, $b in $seq2 satisfies $a > $b`
 - isto što i prethodni izraz
 - `some $a in $seq1, $b in $seq2 satisfies $a > $b+3`
 - da li postoji bar jedna element u $\$seq1$ veći od bilo kog elementa u $\$seq2$ uvećanog za 3
 - fleksibilnija varijanta
 - `exists($seq1[. > $seq2])`
 - XPath varijanta prvog izraza
 - `exists(for $a in $seq1, $b in $seq2 where $a > $b return $a)`
 - puni FLWOR izraz

Poređenje sekvenci

- poređenje član-po-član
 - `deep-equal()` funkcija - ponaša se rekurzivno
 - `shallow-equal()` ne postoji kao ugrađena funkcija

```
declare function shallow-equal($seq1 as node()*,
    $seq2 as node()*) as xs:boolean {
    if (count($seq1) != count($seq2))
        then false()
    else
        empty(
            for $i at $p1 in $seq1
            for $j at $p2 in $seq2
            where $p1 = $p2 and $i is not $j
            return 1
        )
};
```

Poređenje sekvenci

- univerzalno poređenje - uslov je zadovoljen za svaki element
 - primer: svaki element iz \$seq1 veći od svakog elementa iz \$seq2

```
every $a in $seq1, $b in $seq2 satisfies $a gt $b
```

```
empty(for $a in $seq1  
      where not($a > $seq2)  
      return $a )
```

```
empty($seq1[not(. > $seq2)])
```


FLWOR sortiranje

- `order by` klauzula u FLWOR izrazu
 - sortira se po datim ključevima

```
for $i in doc("team.xml")//Employee
where exists($i//Employee)
stable order by $i/@id descending
return $i/Name
```

stavke sa jednakim sort
ključem čuvaju originalni
redosled

u opadajućem redosledu

FLWOR sortiranje

- tretiranje prazne sekvence i NaN
 - `empty least`: prazna sekvenca je manja od svake neprazne; NaN je manji od svake ne-NaN vrednosti i neprazne sekvence
 - `empty greatest`: prazna sekvenca je veća od svake neprazne; NaN je veći od svake ne-NaN vrednosti i neprazne sekvence

```
for $i in (1E0, 2E0, 3E0, 0E0 div 0)
let $key := if ($i < 2.5) then $i else ()
order by $key empty least, $i descending
return $i
```

=>

(2E0, 1E0, NaN, 3E0)

FLWOR grupisanje

- nema posebnog group by operatora
- rezultujući XML predstavlja grupisanje
 - primer: pronaći zaposlene koji imaju svoje podređene i sortirati ga po broju podređenih u opadajućem redosledu

```
for $i in doc("team.xml")//Employee
let $reports := count($i/Employee)
where $reports > 0
order by $reports descending
return
  <Employee name="{ $i/Name}" reports="{ $reports}"/>
```

Obrada grešaka

- statička ili dinamička - zavisno od implementacije
 - npr. `xs:decimal("X")` - može biti prijavljeno prilikom kompajliranja ili izvršavanja
 - `$a + $b` - ako je `$a` ili `$b` sekvenca sa više od jednog elementa, ili ako se vrednosti ne mogu sabrati
 - `"1"+2` će izazvati grešku
 - u XPath-u je to validan izraz: "1" se konvertuje u ceo broj, pa se onda sabere sa 2
- funkcija `error()` - za programsko izazivanje greške
- funkcija `trace()` - za generisanje poruke o grešci bez prekidanja izvršavanja