Povpraševalni jezik XQuery

Iztok Savnik

2018/19

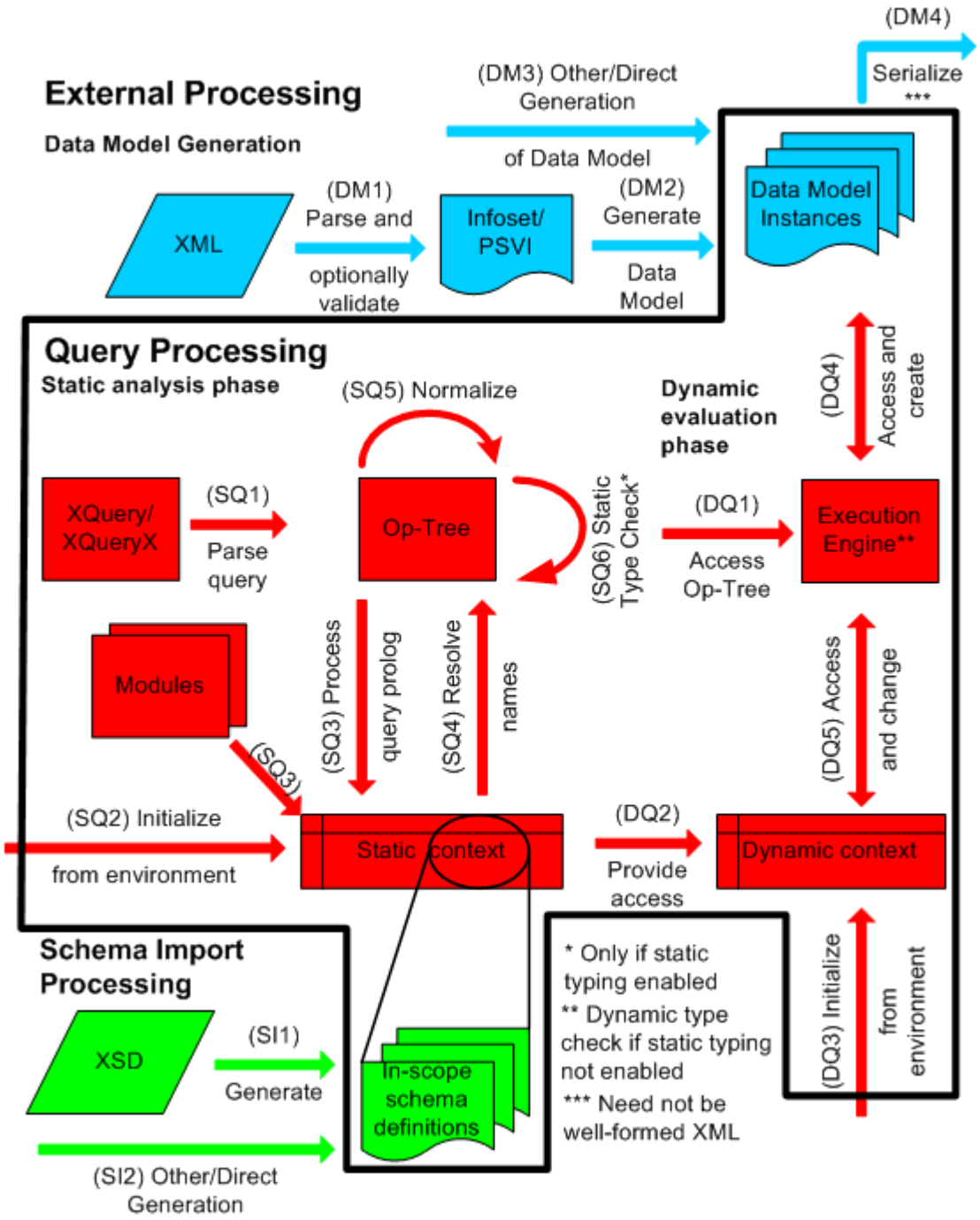
XQuery

- Začetni W3C predlogi:
  - XQL (1999, predlog W3C)
  - XML-QL (1998, predlog W3C)
- Povpraševalni jezik za XML !
- Delo z drevesi oz. usmerjenimi grafi
  - Podatkovni model = usmerjen označen graf

Potek

- Procesni model
- Tipi
- Izrazi (1)
- Izrazi (2)
- Primeri
- Primeri tipičnih SQL poizvedb

Procesni model



Statični in dinamični kontekst

- Statični kontekst:
  - statična analiza dokumenta
- Dinamični kontekst:
  - evaluacija dokumenta

Koncepti

- Urejenost dokumenta
- Atomatizacija
- Vhodni viri
- URI konstante

Urejenost dokumenta

- Urejenost dokumenta je definirana z urejenostjo vozlišč dokumentov med procesiranjem vprašanj, ki se lahko nanašajo na več dreves
- Urejenost dokumenta v drevesih:
  - Koren drevesa je prvo vozlišče
  - Vsako vozlišče je pred otroci in nasledniki
  - Atributi sledijo takoj za elementi s katerimi so povezani
  - Relativna urejenost bratov in sester sledi urejenosti po kateri so napisani v dokumentu
  - Otroci in nasledniki so pred ostalimi brati/sestrami
- Urejenost dokumenta je stabila

Atomatizacija

- Atomatizacija se aplicira nad vrednostmi, kjer se pričakuje sekvenca atomarnih vrednosti
  - Rezultat je sekvenca atomičnih vrednosti ali napaka
  - fn:data() -- aplicirana na sekvenci
- Atomatizacija se uporablja na:
  - Aritmetičnih izrazih
  - Primerjalnih izrazih
  - Klici funkcij
  - “Cast” izrazi
  - Konstruktorji
  - order by stavek

Vhodni viri

- Vhodni viri XQuery
- fn:doc(URI)
  - prebere dokument in ga pretvori v podatkovni model
- fn:collection(URI)
  - prebere kolekcijo in jo pretvori v podatkovni model
- fn:collection()
  - prebere privzeto kolekcijo

URI konstante

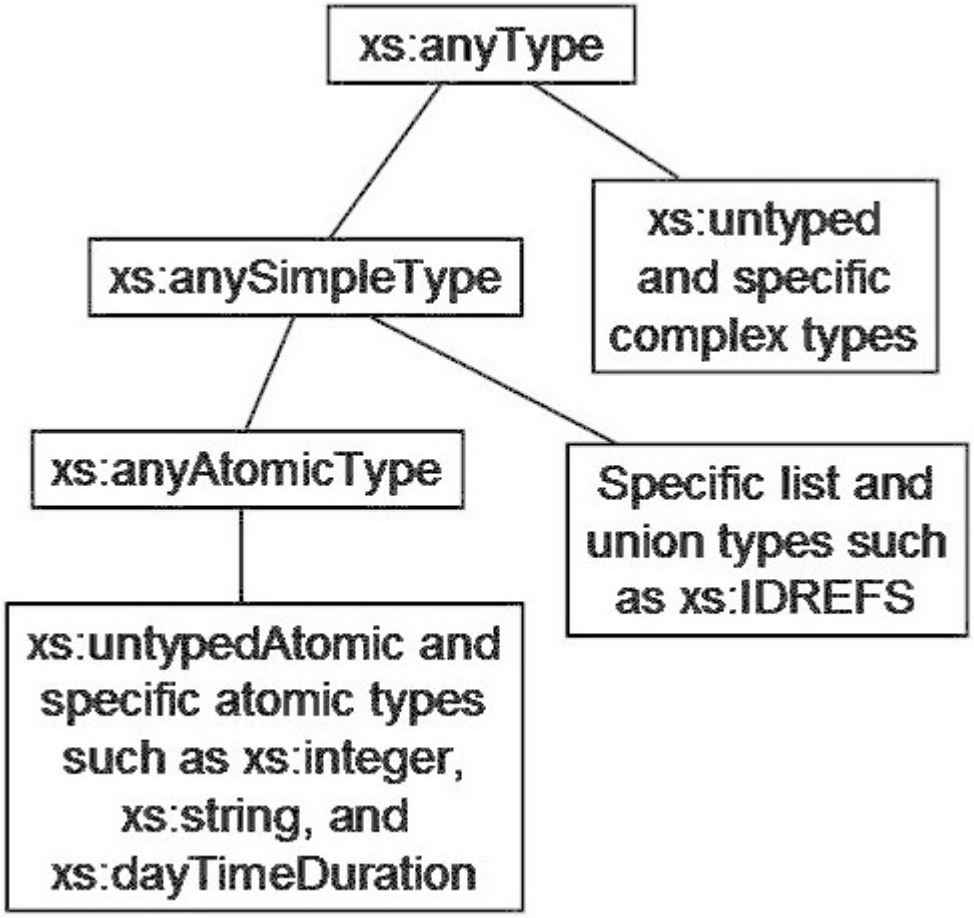
- Včasih je potrebno v vprašanjih uporabljati URI konstante
- Primer legalnega URI:
  - "http://www.w3.org/2005/xpath-functions/collation/codepoint"
- Tip xs:anyURI

Tipi

- Tipi XQuery so definirani na osnovi XMLSchema tipov
- Sekvence
  - Tipi sekvenc, tipi v XQuery izrazih
- Tipi sheme
  - Tipi definirani v XMLSchema
  - Simple, complex, list type, union, atomic type
- Atomični tipi
  - Presek med sekvencami in tipi sheme

Predefinirani tipi sheme

- XMLSchema tipi
- xs imenski prostor
  - xs:untyped
  - xs:untypedAtomic
  - xs:dayTimeDuration
  - xs:yearMonthDuration
  - xs:anyAtomicType



Vrednost vozlišča

- Vsako vozlišče ima
  - vrednost označeno s tipom in
  - tekstovno vrednost.
- Branje vrednosti
  - fn:data() -- vrednost s tipom
  - fn:string – tekstovna vrednost

Tip sekvenc

Tipi sekvenc se uporabljajo povsod, kjer se je potrebno referencirati na tip v XQuery izrazu.

Tip rezultata XQuery poizvedbe je vedno sekvenca.

```
[184]  SequenceType          ::=  ("empty-sequence" "(" ")")
                                  | (ItemType OccurrenceIndicator?)
[186]  ItemType              ::=  KindTest | ("item" "(" ")") |
                                  FunctionTest | MapTest | ArrayTest |
                                  AtomicOrUnionType |
                                  ParenthesizedItemType
[185]  OccurrenceIndicator   ::=  "?" | "*" | "+"                    /* xgc:
                                                                      occurrence-
                                                                      indicators */
[187]  AtomicOrUnionType     ::=  EQName
[188]  KindTest              ::=  DocumentTest
                                  | ElementTest
                                  | AttributeTest
                                  | SchemaElementTest
                                  | SchemaAttributeTest
                                  | PITest
                                  | CommentTest
                                  | TextTest
                                  | NamespaceNodeTest
                                  | AnyKindTest
[190]  DocumentTest          ::=  "document-node" "(" (ElementTest |
                                  SchemaElementTest)? ")"
[199]  ElementTest           ::=  "element" "(" (ElementNameOrWildcard
                                  ("," TypeName "?"?)?)? ")"
[201]  SchemaElementTest     ::=  "schema-element" "("
                                  ElementDeclaration ")"
[202]  ElementDeclaration    ::=  ElementName
[195]  AttributeTest         ::=  "attribute" "(" (AttribNameOrWildcard
                                  ("," TypeName)?)? ")"
[197]  SchemaAttributeTest   ::=  "schema-attribute" "("
                                  AttributeDeclaration ")"
[198]  AttributeDeclaration  ::=  AttributeName
[200]  ElementNameOrWildcard ::=  ElementName | "*"
[204]  ElementName           ::=  EQName
```

...

Primeri tipov sekvenc

- xs:date -- built-in atomic schema type named xs:date
- attribute()? -- refers to an optional attribute node
- element() -- refers to any element node
- element(po:shipto, po:address) -- refers to an element node that has the name po:shipto and has the type annotation po:address (or a schema type derived from po:address)
- element(*, po:address) -- refers to an element node of any name that has the type annotation po:address (or a type derived from po:address)
- element(customer) -- refers to an element node named customer with any type annotation
- node()* -- refers to a sequence of zero or more nodes of any kind
- item()+ refers to a sequence of one or more nodes or atomic values

Ujemanje tipov sekvenc

- Ujemanje tipov sekvenc primerja pričakovani tip (statični) z dinamičnim tipom sekvence.
  - Dinamični tip mora biti izpeljan iz pričakovanega statičnega tipa
  - Zamenljivost (substitutability), ali “subtype substitution”
- derives-from(AT, ET) – tip AT je izpeljan iz ET
  - ET je pričakovan tip in AT je dinamični tip sekvence
- Ujemanje tipa sekvence in vrednosti
  - Prazna sekvenca se ujema s tipom empty-sequence()
  - ItemType se ujema z vrednostjo (eno)
  - Sekvenca ItemType s številom ponovitev vrednosti se ujema z sekvenco, kjer se ItemType ujema z vsako vrednostjo
  - Število ponovitev: ?, *, +

Ujemanje tipov sekvenc

- Ujemanje ItemType in Item
  - derives-from(AT, AtomicOrUnionType) vrne true
- Testi ItemType
  - Testi preverjajo ujemanje tipa (npr. Node()) z vrednostjo
  - Test Item
    - item(), node(), text(), processing-instruction(), processing-instruction(N), comment(), document-node()
  - Test elementa
    - element(), element(ElementName), element(ElementName, TypeName), element(ElementName, TypeName ?), element(*, TypeName), element(*, TypeName ?)
  - Test atributa
    - attribute(), attribute(*), attribute(AttributeName), attribute(AttributeName, TypeName), attribute(*, TypeName)
  - Test funkcij, test polj, itd.

Komentarji

- Sintaksa

```
Comment         ::=     "(:" (CommentContents | Comment)* ":)"
CommentContents         ::=     (Char+ - (Char* ('(:' | ':)') Char*))
```

- Komentiranje vprašanj (: ... :)

Izrazi (1)

- Osnovni izrazi
- Izrazi poti
- Sekvence
- Aritmetični izrazi
- Primerjalni izrazi
- Logični izrazi

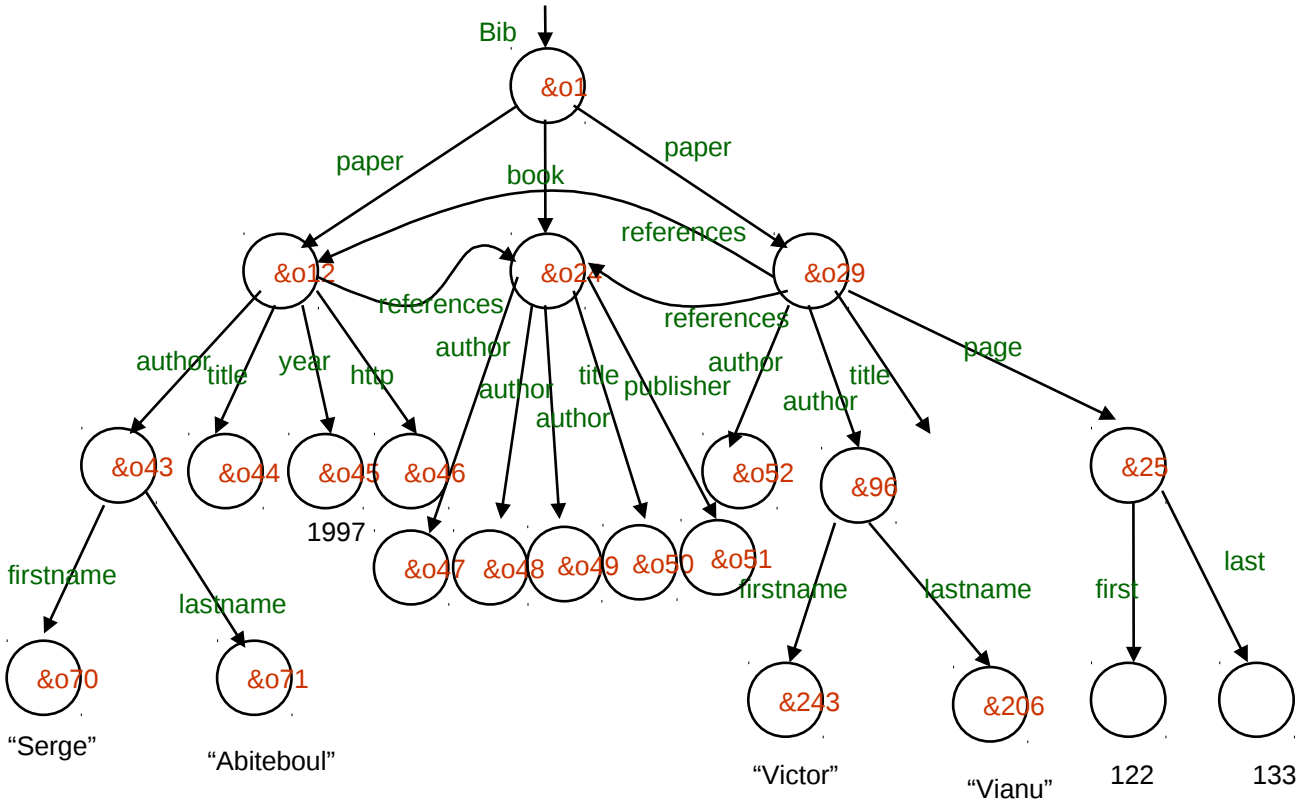
Osnovni izrazi (1)

- Literali
  - Sintaktična predstavitev atomične vrednosti.
  - Celoštevilski literal (12,101)
  - Decimalni literal (12.5,123.9)
  - Double literal (125E2
  - Entitete (<&")
  - Nizi ("12.5", "Ben & Jerry's")
- Spremenljivke
  - "$" VarName
  - $count,$value

Osnovni izrazi (2)

- Konstruktorji
  - xs:integer("12")
  - xs:date("2001-08-25")
  - xs:dayTimeDuration("PT5H")
- Function calls
  - my:three-argument-function(1, 2, 3)
  - my:two-argument-function((1, 2), 3)
  - my:zero-argument-function( )

Izrazi poti (1)




```
Bib/paper={&o12,&o29}
Bib/book/publisher={&o51}
Bib/paper/author/lastname={&o71,&206}
```

Izrazi poti (2)

- Koraki

StepExpr ::= FilterExpr | AxisStep

AxisStep ::= (ReverseStep | ForwardStep) PredicateList

ForwardStep ::= (ForwardAxis NodeTest) | AbbrevForwardStep

ReverseStep ::= (ReverseAxis NodeTest) | AbbrevReverseStep

PredicateList ::= Predicate*

- Osi

ForwardStep = child, descendant, attribute, self, descendant-or-self, following-sibling, following

BackwardStep = parent, ancestor, preceding-sibling, preceding, ancestor-or-self

Izrazi poti (3)

- Testi vozlišč

NodeTest ::= KindTest | NameTest
NameTest ::= QName | Wildcard
Wildcard ::= "*"
| (NCName ":" "*")
| ("*" ":" NCName)

- Primeri

node(), text(), comment(), element(), schema-element(person), element(person), element(*, surgeon), attribute(), attribute(price), attribute(*, xs:decimal), document-node(), document-node(element(book))

Izrazi poti (4)

- Predikati

Predicate ::= "[" Expr "]"

- Primeri

child::chapter[2]

descendant::toy[attribute::color = "red"]

child::employee[secretary][assistant]

- Polna in okrajšana sintaksa

Sekvence (1)

- Sekvenca je definirana z operacijo “,”
- Primeri:

(10, 1, 2, 3, 4)

(10, (1, 2), (), (3, 4)) --- = (10, 1, 2, 3, 4)

(salary, bonus) --- otroci elementa salary + otroci elementa bonus

($price, $price) --- vrednosti spremenljivk

Sekvence (2)

- Filtri

```
$products[price gt 100]
(1 to 100)[. mod 5 eq 0]
(21 to 29)[5]
$orders[fn:position() = (5 to 9)]
$book/(chapter | appendix)[fn:last()]
fn:doc("zoo.xml")/fn:id('tiger'])
```

Sekvence (3)

- Kombiniranje sekvenc

A,B in C so elementi.

$seq1 = (A, B)

$seq2 = (A, B)

$seq3 = (B, C)

$seq1 union $seq2 = (A, B).

$seq2 union $seq3 = (A, B, C).

$seq1 intersect $seq2 = (A, B).

$seq2 intersect $seq3 = B.

$seq1 except $seq2 = {}.

$seq2 except $seq3 = A.

Primerjalne operacije (1)

- Sintaksa

ComparisonExpr ::= RangeExpr ((ValueComp
| GeneralComp
| NodeComp) RangeExpr)?

ValueComp ::= "eq" | "ne" | "lt" | "le" | "gt" | "ge"

GeneralComp ::= "=" | "!=" | "<" | "<=" | ">" | ">="

NodeComp ::= "is" | "<<" | ">>"

- Primerjava vrednosti

$book1/author eq "Kennedy"

//product[weight gt 100]

<a>5 eq <a>5 = true

<a>5 eq 5 = true

my:hatsize(5) eq my:shoesize(5) = true (: if both are restriction of numeric :)

Primerjalne operacije (2)

- Splošne primerjave
  - Najprej se izvede atomatizacija
  - Potem se določijo tipi vrednosti
  - Izvede se pripadajoča vrednostna primerjava
- Primerjava seznamov z eksistenčnim kvantifikatorjem

```
$book1/author = "Kennedy"    // true if left exp evals to “Kennedy”
(1, 2) = (2, 3)   // true
(2, 3) = (3, 4)   // true
(1, 2) = (3, 4)   // false
(1, 2) != (2, 3)  // true
```

Primerjalne operacije (3)

- Primerjava vozlišč “is”
  - Identičnost vozlišč = enakost
  - Vrstni red vozlišč v dokumentu
- Primeri:

```
/books/book[isbn="1558604820"] is /books/book[call="QA76.9 C3845"]
(: true if both sides evaluate to same element :)
<a>5</a> is <a>5</a>  (: false :)
/transactions/purchase[parcel="28-451"]
   << /transactions/sale[parcel="33-870"]   (: levo vozlišče pred desnim => true :)
```

Izrazi (2)

- Konstruktorji
- FLWOR izrazi
- Urejeni in neurejeni izrazi
- Pogojni izrazi
- Kvantificirani izrazi
- Izrazi na tipih sekvenc
- Validacija
- Ekstenzija

Konstruktorji elementov

- S konstruktorji kreiramo XML strukture
- Direktni in izračunani konstruktorji
- Direktni konstruktorji

```
<book isbn="isbn-0060229357">
    <title>Harold and the Purple Crayon</title>
    <author>
        <first>Crockett</first>
        <last>Johnson</last>
    </author>
</book>
```

Konstruktorji elementov (2)

- Vgnezdeni Xquery izrazi z {}

```
<example>
   <p> Here is a query. </p>
   <eg> $b/title </eg>
   <p> Here is the result of the query. </p>
   <eg>{ $b/title }</eg>
</example>
```

Result:

```
<example>
  <p> Here is a query. </p>
  <eg> $b/title </eg>
  <p> Here is the result of the query. </p>
  <eg><title>Harold and the Purple Crayon</title></eg>
</example>
```

Konstruktorji elementov (3)

- Konstruktorji atributov

<shoe size="{7}"/> --> <shoe size="7"/>

<chapter ref="[{1, 5 to 7, 9}]"/> --> <chapter ref="[1, 5, 6, 7, 9]"/>

<shoe size="As big as {$hat/@size}"/> --> $hat/@size dobi vrednost

{1, "2", "3"} --> 1 2 3

```
<box xmlns:metric = "http://example.org/metric/units"
        xmlns:english = "http://example.org/english/units">
  <height> <metric:meters>3</metric:meters> </height>
  <width> <english:feet>6</english:feet> </width>
  <depth> <english:inches>18</english:inches> </depth>
</box>
```

Atributi za kreiranje imenskih prostorov

Konstruktorji elementov (4)

- Izračunani konstruktorji
  - Alternativni način za konstrukcijo XML
  - Konstruktorji: element, attribute, document, text, processing-instruction, ali comment.

```
element book {
   attribute isbn {"isbn-0060229357" },
   element title { "Harold and the Purple Crayon"},
   element author {
      element first { "Crockett" },
      element last {"Johnson" }
   }
}
```

Konstruktorji elementov (5)

- Imena elementov laho izračnamo!
  - $dict ima vrednost slovarja <dictionary>, ki vsebuje zapise <entry>

```
<entry word="address">
   <variant xml:lang="de">Adresse</variant>
   <variant xml:lang="it">indirizzo</variant>
</entry>
```

  - $e ima vrednost

```
<address>123 Roosevelt Ave. Flushing, NY 11368</address>
```

  - Vprašanje:

```
element
   {$dict/entry[@word=name($e)]/variant[@xml:lang="it"]}
   {$e/@*, $e/node()}
```

Rezultat:

```
<indirizzo>123 Roosevelt Ave. Flushing, NY 11368</indirizzo>
```

Konstruktorji (6)

- Konstruktorji atributov

```
attribute
   { if ($sex = "M") then "husband" else "wife" }
   { <a>Hello</a>, 1 to 3, <b>Goodbye</b> }
```

- Konstruktor dokumenta

```
document
   { <author-list>  {fn:doc("bib.xml")/bib/book/author} </author-list> }
```

- Konstruktor teksta
- Konstruktor komentarja

```
let $homebase := "Houston"
return comment {fn:concat($homebase, ", we have a problem.")}
```

FLWOR (1)

- Deklarativno povpraševanje
- FLWOR: for, let, where, order by, in return
- Primer vsebuje vse gradnike:

```
for $d in fn:doc("depts.xml")/depts/deptno
let $e := fn:doc("emps.xml")/emps/emp[deptno = $d]
where fn:count($e) >= 10
order by fn:avg($e/salary) descending
return
   <big-dept>
      {
      $d,
      <headcount>{fn:count($e)}</headcount>,
      <avgsal>{fn:avg($e/salary)}</avgsal>
      }
   </big-dept>
```

FLWOR (2)

- for stavek definira iteracijo po kolekciji

```
for $s in (<one/>, <two/>, <three/>)
return <out>{$s}</out>
```

```
<out>
  <one/>
</out>
<out>
  <two/>
</out>
<out>
  <three/>
</out>
```

- Vsaka povezava s spremenljivko ima indeks:

```
for $car at $i in ("Ford", "Chevy"),
    $pet at $j in ("Cat", "Dog")
```

```
($i = 1, $car = "Ford", $j = 1, $pet = "Cat")
($i = 1, $car = "Ford", $j = 2, $pet = "Dog")
($i = 2, $car = "Chevy", $j = 1, $pet = "Cat")
($i = 2, $car = "Chevy", $j = 2, $pet = "Dog")
```

FLWOR (3)

- let stavek definira prirejanje vrednosti

```
let $s := (<one/>, <two/>, <three/>)
return <out>{$s}</out>
```

- Rezultat:

```
<out>
   <one/>
   <two/>
   <three/>
</out>
```

FLWOR (4)

- Stavek where

```
fn:avg(for $x at $i in $inputvalues
    where $i mod 100 = 0
    return $x)
```

- Stavka order by in return

```
for $e in $employees
order by $e/salary descending
return $e/name
```

```
for $b in $books/book[price < 100]
order by $b/title
return $b
```

FLWOR (5)

- Primer
  - Vgnezdeni FLWOR
  - Vhod $bib

```
<bib>
  <book>
    <title>TCP/IP Illustrated</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
  </book>
  <book>
    <title>Advanced Programming
           in the Unix Environment</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
  </book>
  <book>
    <title>Data on the Web</title>
    <author>Abiteboul</author>
    <author>Buneman</author>
    <author>Suciu</author>
  </book>
</bib>
```

FLWOR (6)

- Vhod spremeni tako, da bo izpisan avtor in seznam del avtorja

```
<authlist>
 {
  for $a in fn:distinct-values($bib/book/author)
  order by $a
  return
    <author>
      <name> {$a} </name>
      <books>
       {
        for $b in $bib/book[author = $a]
        order by $b/title
        return $b/title
       }
      </books>
    </author>
 }
</authlist>
```

```
<authlist>
  <author>
    <name>Abiteboul</name>
    <books>
      <title>Data on the Web</title>
    </books>
  </author>
  <author>
    <name>Buneman</name>
    <books>
      <title>Data on the Web</title>
    </books>
  </author>
  <author>
    <name>Stevens</name>
    <books>
      <title>Advanced Programming
            in the Unix Environment</title>
      <title>TCP/IP Illustrated</title>
    </books>
  </author>
  <author>
    <name>Suciu</name>
    <books>
      <title>Data on the Web</title>
    </books>
  </author>
</authlist>
```

Urejeni in neurejeni izrazi

- Imamo dva načina (mode)
  - ordered, unordered
- Sintaksa:

```
OrderedExpr      ::=   "ordered" "{" Expr "}"
UnorderedExpr    ::=   "unordered" "{" Expr "}"
```

- Primer:

```
unordered {
  for $p in fn:doc("parts.xml")/parts/part[color = "Red"],
      $s in fn:doc("suppliers.xml")/suppliers/supplier
  where $p/suppno = $s/suppno
  return
    <ps>
      { $p/partno, $s/suppno }
    </ps>
}
```

Pogojni izrazi

- Pogojni stavek if-then-else
- Sintaksa:

IfExpr ::= "if" "(" Expr ")" "then" ExprSingle "else" ExprSingle

- Primer:

Uporaba primerjalnih operacij

```
if ($widget1/unit-cost < $widget2/unit-cost)
  then $widget1
  else $widget2
```

Preverimo obstoj atributa discounted

```
if ($part/@discounted)
  then $part/wholesale
  else $part/retail
```

Kvantificirani izrazi

- Univerzalna in eksistenčna kvantifikacija izrazov
- Sintaksa:

```
QuantifiedExpr ::= ("some" | "every") "$" VarName TypeDeclaration? "in"
                   ExprSingle ("," "$" VarName TypeDeclaration? "in" ExprSingle)*
                   "satisfies" ExprSingle
```

- Primeri:

Vsi pari so označeni z discounted

```
every $part in /parts/part satisfies $part/@discounted
```

Kartezijski produkt: some --> true, Every --> false.

```
some $x in (1, 2, 3), $y in (2, 3, 4)
    satisfies $x + $y = 4
every $x in (1, 2, 3), $y in (2, 3, 4)
    satisfies $x + $y = 4
```

Preostali gradniki

- Preverjanje tipov (instanceOf)
- Case stavek na vrstah tipov
- Prirejanje tipov (type cast)
- Validiranje izrazov
- Moduli in prologi
- Polja in slovarji

Primeri

- Primeri, ki vsebujejo vse predstavljene gradnike
- Bibliografija – seznam knjig, ki vsebuje avtorje, založbe, naslove knjig, leto izdaje, itd.

Konceptualna shema

- Bibliografija
- Zapisi predstavljajo knjige

```
<!ELEMENT bib  (book* )>
<!ELEMENT book  (title,  (author+ | editor+ ),
publisher, price )>
<!ATTLIST book  year CDATA  #REQUIRED >
<!ELEMENT author  (last, first )>
<!ELEMENT editor  (last, first, affiliation )>
<!ELEMENT title  (#PCDATA )>
<!ELEMENT last  (#PCDATA )>
<!ELEMENT first  (#PCDATA )>
<!ELEMENT affiliation  (#PCDATA )>
<!ELEMENT publisher  (#PCDATA )>
<!ELEMENT price  (#PCDATA )>
```

- http://bstore1.example.com/bib.xml

Osnovni podatki

```
<bib>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
      <author><last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price>65.95</price>
   </book>
   <book year="1992">
      <title>Advanced Programming in the Unix environment</title>
      <author><last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price>65.95</price>
   </book>
   <book year="2000">
      <title>Data on the Web</title>
      <author><last>Abiteboul</last><first>Serge</first></author>
      <author><last>Buneman</last><first>Peter</first></author>
      <author><last>Suciu</last><first>Dan</first></author>
      <publisher>Morgan Kaufmann Publishers</publisher>
      <price>39.95</price>
   </book>
   <book year="1999">
      <title>Economics of Technology and Content for Digital TV</title>
      <editor>
            <last>Gerbarg</last><first>Darcy</first>
             <affiliation>CITI</affiliation>
      </editor>
         <publisher>Kluwer Academic Publishers</publisher>
      <price>129.95</price>
   </book>
</bib>
```

Primer 1

Izpiši leto in naslov knjig založnika Addison-Wesley po 1991.

```
<bib>
 {
  for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
     { $b/title }
    </book>
 }
</bib>
```

Rezultat:

```
<bib>
    <book year="1994">
        <title>TCP/IP Illustrated</title>
    </book>
    <book year="1992">
        <title>Advanced Programming in the Unix environment</title>
    </book>
</bib>
```

Primer 2

Izpiši vse pare naslov-avtor, kjer je vsak par obdan z elementom <result>.

```
<results>
  {
    for $b in doc("http://bstore1.example.com/bib.xml")/bib/book,
        $t in $b/title,
        $a in $b/author
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

Primer 2

Rezultat:

```
<results>
    <result>
        <title>TCP/IP Illustrated</title>
        <author>
            <last>Stevens</last>
            <first>W.</first>
        </author>
    </result>
    <result>
        <title>Advanced Programming in Unix environment</title>
        <author>
            <last>Stevens</last>
            <first>W.</first>
        </author>
    </result>
    <result>
        <title>Data on the Web</title>
        <author>
            <last>Abiteboul</last>
            <first>Serge</first>
        </author>
    </result>
    <result>
        <title>Data on the Web</title>
        <author>
            <last>Buneman</last>
            <first>Peter</first>
        </author>
    </result>
    <result>
        <title>Data on the Web</title>
        <author>
            <last>Suciu</last>
            <first>Dan</first>
        </author>
    </result>
</results>
```

Primer 3

Za vsako knjigo bibligrafije izpiši naslov in avtorje. Vsak rezultat naj bo obdan z <result>.

```
<results>
{
    for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
    return
        <result>
            { $b/title }
            { $b/author  }
        </result>
}
</results>
```

Primer 3

Rezultat:

```
<results>
   <result>
      <title>TCP/IP Illustrated</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </result>
   <result>
      <title>Advanced Programming in the Unix environment</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </result>
   <result>
      <title>Data on the Web</title>
      <author>
         <last>Abiteboul</last>
         <first>Serge</first>
      </author>
      <author>
         <last>Buneman</last>
         <first>Peter</first>
      </author>
      <author>
         <last>Suciu</last>
         <first>Dan</first>
      </author>
   </result>
   <result>
      <title>The Economics of Technology and Content for Digital TV</title>
   </result>
</results>
```

Primer 4

Za vsakega avtorja v bibliografiji izpiši ime avtorja in seznam naslovov knjig grupirano z <result>.

```
<results>
 {
   let $a := doc("http://bstore1.example.com/bib/bib.xml")//author
   for $last in distinct-values($a/last),
       $first in distinct-values($a[last=$last]/first)
   order by $last, $first
   return
       <result>
           <author>
              <last>{ $last }</last>
              <first>{ $first }</first>
           </author>
           {
               for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
               where some $ba in $b/author
                     satisfies ($ba/last = $last and $ba/first=$first)
               return $b/title
           }
       </result>
 }
</results>
```

Primer 4

Rezultat:

```
<results>
   <result>
      <author>
         <last>Abiteboul</last>
         <first>Serge</first>
      </author>
      <title>Data on the Web</title>
   </result>
   <result>
      <author>
         <last>Buneman</last>
         <first>Peter</first>
      </author>
      <title>Data on the Web</title>
   </result>
   <result>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
      <title>TCP/IP Illustrated</title>
      <title>Advanced Programming in the Unix environment</title>
   </result>
   <result>
      <author>
         <last>Suciu</last>
         <first>Dan</first>
      </author>
      <title>Data on the Web</title>
   </result>
</results>
```

Primer 5

Podatki za Primer 5

```
<!ELEMENT reviews (entry*)>
<!ELEMENT entry   (title, price, review)>
<!ELEMENT title   (#PCDATA)>
<!ELEMENT price   (#PCDATA)>
<!ELEMENT review  (#PCDATA)>
```

```
<reviews>
   <entry>
      <title>Data on the Web</title>
      <price>34.95</price>
      <review>
            A very good discussion of semi-structured database
            systems and XML.
      </review>
   </entry>
   <entry>
      <title>Advanced Programming in the Unix environment</title>
      <price>65.95</price>
      <review>
            A clear and detailed discussion of UNIX programming.
      </review>
   </entry>
   <entry>
      <title>TCP/IP Illustrated</title>
      <price>65.95</price>
      <review>
            One of the best books on TCP/IP.
      </review>
   </entry>
</reviews>
```

Primer 5

Za vsako knjigo, ki je v obeh katalogih bstore1.example.com in bstore2.example.com izpiši ceno iz obeh katalogov.

```
<books-with-prices>
 {
   for $b in doc("http://bstore1.example.com/bib.xml")//book,
       $a in doc("http://bstore2.example.com/reviews.xml")//entry
   where $b/title = $a/title
   return
       <book-with-prices>
           { $b/title }
           <price-bstore2>{ $a/price/text() }</price-bstore2>
           <price-bstore1>{ $b/price/text() }</price-bstore1>
       </book-with-prices>
 }
</books-with-prices>
```

Primer 5

Rezultat:

```
<books-with-prices>
   <book-with-prices>
      <title>TCP/IP Illustrated</title>
      <price-bstore2>65.95</price-bstore2>
      <price-bstore1>65.95</price-bstore1>
   </book-with-prices>
   <book-with-prices>
      <title>Advanced Programming in the Unix environment</title>
      <price-bstore2>65.95</price-bstore2>
      <price-bstore1>65.95</price-bstore1>
   </book-with-prices>
   <book-with-prices>
      <title>Data on the Web</title>
      <price-bstore2>34.95</price-bstore2>
      <price-bstore1>39.95</price-bstore1>
   </book-with-prices>
</books-with-prices>
```

Primer 6

Za vsako knjigo, ki ima vsaj enega avtorja, izpiši naslov in prva dva avtorja ter prazen element <et-al>, če ima knjiga več avtorjev.

```
<bib>
 {
   for $b in doc("http://bstore1.example.com/bib.xml")//book
   where count($b/author) > 0
   return
       <book>
           { $b/title }
           {
               for $a in $b/author[position()<=2]
               return $a
           }
           {
               if (count($b/author) > 2)
                then <et-al/>
                else ()
           }
       </book>
 }
</bib>
```

Primer 6

Rezultat:

```
<bib>
   <book>
      <title>TCP/IP Illustrated</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </book>
   <book>
      <title>Advanced Programming in the Unix environment</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </book>
   <book>
      <title>Data on the Web</title>
      <author>
         <last>Abiteboul</last>
         <first>Serge</first>
      </author>
      <author>
         <last>Buneman</last>
         <first>Peter</first>
      </author>
      <et-al/>
   </book>
</bib>
```

Primer 7

Izpiši naslove in leta objave knjig založnika Addison-Wesley po letu 1991, po abecednem vrstnem redu.

```
<bib>
 {
   for $b in doc("http://bstore1.example.com/bib.xml")//book
   where $b/publisher = "Addison-Wesley" and $b/@year > 1991
   order by $b/title
   return
       <book>
          { $b/@year }
          { $b/title }
       </book>
 }
</bib>
```

Primer 7

Rezultat:

```
<bib>
   <book year="1992">
      <title>Advanced Programming in the Unix environment</title>
   </book>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
   </book>
</bib>
```

Primer 8

```
<!ELEMENT chapter (title, section*)>
<!ELEMENT section (title, section*)>
<!ELEMENT title (#PCDATA)>
```

Podatki za Primer 8

```
<chapter>
   <title>Data Model</title>
   <section>
      <title>Syntax For Data Model</title>
   </section>
   <section>
      <title>XML</title>
      <section>
         <title>Basic Syntax</title>
      </section>
      <section>
         <title>XML and Semistructured Data</title>
      </section>
   </section>
</chapter>
```

Primer 8

V dokumentu "books.xml" poišči vse sekcije ali poglavja, ki vsebujejo besedo "XML", ne glede na nivo gnezdenja.

```
<results>
  {
    for $t in doc("books.xml")//(chapter | section)/title
    where contains($t/text(), "XML")
    return $t
  }
</results>
```

```
<results>
    <title>XML</title>
    <title>XML and Semistructured Data</title>
</results>
```

Primer 9

- Podatki in shema za Primer 9

```
<!ELEMENT prices (book*)>
<!ELEMENT book (title, source, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

```
<prices>
  <book>
    <title>Advanced Programming in the Unix
environment</title>
    <source>bstore2.example.com</source>
    <price>65.95</price>
  </book>
  <book>
    <title>Advanced Programming in the Unix
environment</title>
    <source>bstore1.example.com</source>
    <price>65.95</price>
  </book>
  <book>
    <title>TCP/IP Illustrated</title>
    <source>bstore2.example.com</source>
    <price>65.95</price>
  </book>
  <book>
    <title>TCP/IP Illustrated</title>
    <source>bstore1.example.com</source>
    <price>65.95</price>
  </book>
  <book>
    <title>Data on the Web</title>
    <source>bstore2.example.com</source>
    <price>34.95</price>
  </book>
  <book>
    <title>Data on the Web</title>
    <source>bstore1.example.com</source>
    <price>39.95</price>
  </book>
</prices>
```

Primer 9

V dokumentu "prices.xml" poišči minimalno ceno za vsako knjigo v obliki elementa <minprice> z atributom, ki vsebuje naslov knjige.

```
<results>
 {
   let $doc := doc("prices.xml")
   for $t in distinct-values($doc//book/title)
   let $p := $doc//book[title = $t]/price
   return
     <minprice title="{ $t }">
       <price>{ min($p) }</price>
     </minprice>
 }
</results>
```

Primer 9

Rezultat:

```
<results>
    <minprice title="Advanced Programming in the Unix environment">
        <price>65.95</price>
    </minprice>
    <minprice title="TCP/IP Illustrated">
        <price>65.95</price>
    </minprice>
    <minprice title="Data on the Web">
        <price>34.95</price>
    </minprice>
</results>
```

Primeri tipičnih SQL poizvedb

- ... z XQuery
- parts.xml
  - elementi <part>: <partno>, <description>
- suppliers.xml
  - <supplier>: <suppno>, <suppname>
- catalog.xml
  - Relacija med <suppliers> in <parts>:
  - <item>: <partno>, <suppno>, in <price>

Stik

Izpiši opis dela, ime dobavitelja in ceno za vse dele.

Inner join

```
<descriptive-catalog>
  {
    for $i in fn:doc("catalog.xml")/items/item,
        $p in fn:doc("parts.xml")/parts/part[partno = $i/partno],
        $s in fn:doc("suppliers.xml")/suppliers
                /supplier[suppno = $i/suppno]
    order by $p/description, $s/suppname
    return
      <item>
        {
        $p/description,
        $s/suppname,
        $i/price
        }
      </item>
  }
</descriptive-catalog>
```

Stik

Izpiši dobavitelje in opise delov, ki jih dobavljajo, tudi v primeru, da ne dobavljajo nobenega dela.

Left outer join

```
for $s in fn:doc("suppliers.xml")/suppliers/supplier
order by $s/suppname
return
   <supplier>
     {
      $s/suppname,
      for $i in fn:doc("catalog.xml")/items/item
               [suppno = $s/suppno],
          $p in fn:doc("parts.xml")/parts/part
               [partno = $i/pno]
      order by $p/description
      return $p/description
     }
   </supplier>
```

Grupiranje

Poišči številko dela in povprečno ceno delov, ki imajo Vsaj 3 dobavitelje.

```
for $pn in fn:distinct-values(fn:doc("catalog.xml")/items/item/partno)
let $i := fn:doc("catalog.xml")/items/item[partno = $pn]
where fn:count($i) >= 3
order by $pn
return
   <well-supplied-item>
      <partno> {$pn} </partno>
      <avgprice> {fn:avg($i/price)} </avgprice>
   </well-supplied-item>
```

Viri

- XQuery 3.1: An XML Query Language, https://www.w3.org/TR/xquery-31/, 2017.
- XQuery 1.0: An XML Query Language (Second Edition), http://www.w3.org/TR/xquery/, 2011.
- XML Query Use Cases, http://www.w3.org/TR/xquery-use-cases/, 2007.
- R. Ramakrishnan, J. Gehrke, Database Management Systems, 3rd Edition, McGraw Hill, 2004.