FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

dbai

Semi-structured Data

8 - XQuery

Mantas Šimkus

Outline

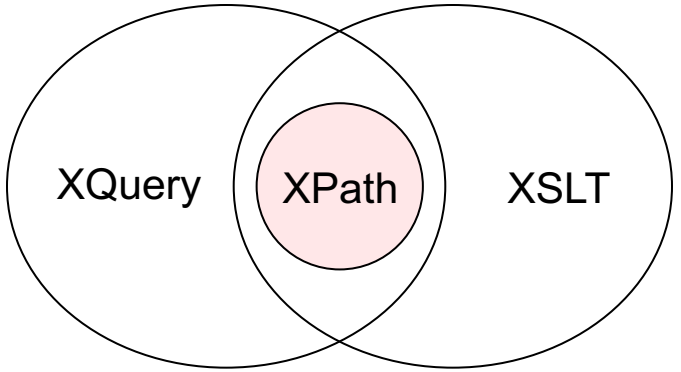
- What is XQuery?
- XQuery at First Glance
- FLWOR Expressions
- Element Constructors
- List, Conditional and Quantified Expressions
- Joins
- Aggregating Values

What is XQuery?

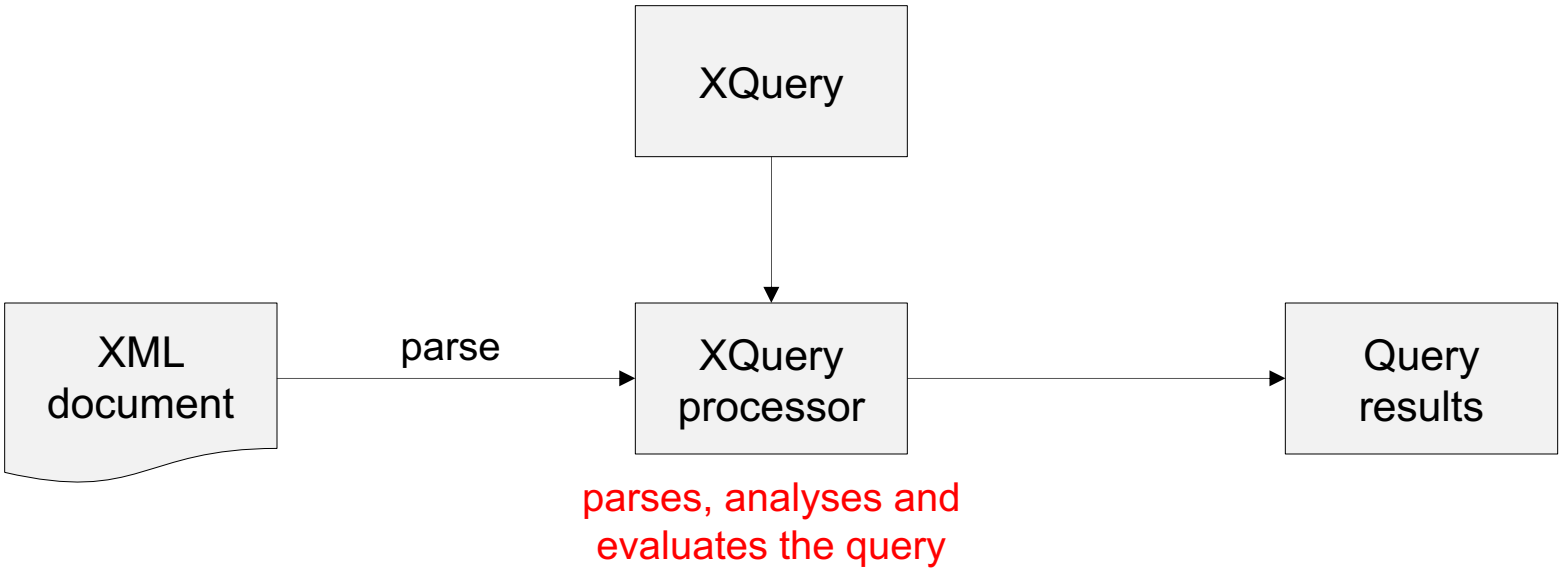
- XQuery is a language for querying XML data
- XQuery for XML is like SQL for relational databases
- XQuery is built on XPath expressions
- As expected, XQuery is a W3C standard

XQuery vs. XPath

- XPath is essentially a subset of XQuery
- XQuery has a number of features not supported by XPath
- XQuery can structure or sort query results (not just select elements and attributes)



Processing XQueries



- Analysis phase: finds syntax errors and other static errors that do not depend on the input document
- Evaluation phase: may raise dynamic errors (e.g., missing input document or division by zero)
- A number of implementations available - http://www.w3.org/XML/Query

XQuery at First Glance

```
<courses>
    <course semester="Summer">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
doc("courses.xml")/courses/course/title
```

```
<title> Semi-structured Data </title>
<title> Databases </title>
```

XQuery at First Glance

```
<courses>
    <course semester="Summer">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
doc("courses.xml")/
        courses/course[@semester="Winter"]
```

```
<course semester="Winter">
    <title> Databases </title>
    <day> Tuesday </day>
    <time> 09:15 </time>
    <location> HS8 </location>
</course>
```

XQuery at First Glance

```
<courses>
    <course semester="Summer">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
doc("courses.xml")/
        courses/course[@semester="Winter"]/title
```

```
<title> Databases </title>
```

XQuery at First Glance

```
<courses>
    <course semester="Summer">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
for $x in doc("courses.xml")/courses/course
where $x/@semester="Winter"
return $x/title
```

```
<title> Databases </title>
```

Equivalent to the query

```
doc("courses.xml")/
        courses/course[@semester="Winter"]/title
```

XQuery at First Glance

```
<courses>
    <course semester="Summer">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
for $x in doc("courses.xml")/courses/course
where $x/@semester="Winter"
order by $x/title
return $x/title
```

```
<title> Databases </title>
```

XQuery at First Glance

```
<courses>
    <course semester="Winter">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
for $x in doc("courses.xml")/courses/course
where $x/@semester="Winter"
order by $x/title
return $x/title
```

```
<title> Databases </title>
<title> SSD </title>
```

XQuery at First Glance

```
<courses>
    <course semester="Winter">
        <title> SSD </title>
        <day> Thursday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
    <course semester="Winter">
        <title> Databases </title>
        <day> Tuesday </day>
        <time> 09:15 </time>
        <location> HS8 </location>
    </course>
</courses>
```

```
for $x in doc("courses.xml")/courses/course
where $x/@semester="Winter"
order by $x/title descending
return $x/title
```

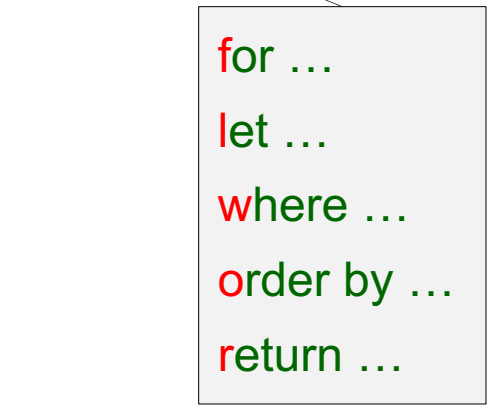
```
<title> SSD </title>
<title> Databases </title>
```

Up to Now

- **What is XQuery?**
- **XQuery at First Glance**
- FLWOR Expressions
- Element Constructors
- List, Conditional and Quantified Expressions
- Joins
- Aggregating Values

FLWOR Expressions

- The main engine of XQuery is FLWOR expressions



- Pronounced "Flower Expressions"

- Generalize Select-From-Having-Where in SQL

FLWOR Expressions: A Complete Example

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
    <large_dept>
        {$d}
        <size> {count($e)} </size>
        <avg_salary> {avg($e/salary)} </avg_salary>
    </large_dept>
```

a list of departments with at least ten employees, sorted by average salary

FLWOR Expressions: Semantics

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
      <large_dept>
            {$d}
            <size> {count($e)} </size>
            <avg_salary> {avg($e/salary)} </avg_salary>
      </large_dept>
```

- for generates bindings of dept_no values to $d
- let associates to each binding a further binding of the list of employee elements with that dept_no to $e
- where filters that list to keep only the desired pairs
- order by sorts that lists by the given criteria
- return constructs for each pair a resulting value

FLWOR Expressions: General Rules

- for and let may be used many times in any order
- Only one where is allowed
- More than one sorting criteria can be specified

order by <expression> ascending, <expression> descending, …

Difference Between for and let

```
for $x in (1,2,3)
let $y := ("a", "b")
return ($x, $y)
```

1 a b 2 a b 3 a b

```
let $x := (1,2,3)
for $y in ("a", "b")
return ($x, $y)
```

1 2 3 a 1 2 3 b

```
for $x in (1,2,3)
for $y in ("a", "b")
return ($x, $y)
```

1 a 1 b 2 a 2 b 3 a 3 b

```
let $x := (1,2,3)
let $y := ("a", "b")
return ($x, $y)
```

1 2 3 a b

Element Constructors

- An XQuery expression may construct a new XML element

- XML constructs can be used to create elements and attributes that appear in the query result
  - Wrapping results in a new element
  - Adding attributes to results

- A key difference compared to XPath

Element Constructors

Wrapping results in a new element

```
<sorted_departments> {
  for $d in doc("departments.xml")//dept_no
  let $e := doc("employees.xml")//employee[dept_no = $d]
  where count($e) >= 10
  order by avg($e/salary) descending
  return
    <large_dept>
        {$d}
        <size> {count($e)} </size>
        <avg_salary> {avg($e/salary)} </avg_salary>
    </large_dept>                    }  </sorted_departments>
```

Element Constructors

Adding attributes to results

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
    <large_dept name = "{$d}">
        <size> {count($e)} </size>
        <avg_salary> {avg($e/salary)} </avg_salary>
    </large_dept>
```

List Expressions

- XQuery expressions manipulate lists of items
  - Value lists: (1,2,3), ("a", "b")
  - Results of XPath expressions

- Many operators are supported
  - Range expressions (e.g., ”3 *to* 10”)
  - Concatenation using “**,**”
  - Set operators (union, intersect, except)

- Many functions are supported
  - count, avg, max, min, sum, distinct-values, …

List Expressions: Example

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
    <large_dept>
        {$d}
        <size> {count($e)} </size>
        <avg_salary> {avg($e/salary)} </avg_salary>
    </large_dept>
```

List Expressions: Example

```
<coauthors>
{
    for $coauthor in
    distinct-values(
        doc(”SimkusMantas.xml")//author[not(./text()=//dblpperson/@name)]
        )
return
     <coauthor>
          {$coauthor}
     </coauthor>
}
</coauthors>
```

Conditional Expressions

XQuery supports general if-then-else expressions

```
for $b in doc("books.xml")/bookstore/book
return
    if ($b/@category = "children")
    then <child> {$b} </child>
    else <adult> {$b} </adult>
```

ATTENTION: else is required, but it can be just else ()

Quantified Expressions

XQuery allows quantified expressions (exist, forall)

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where some $s in $e/salary satisfies $s > 1000
return $d
```

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where every $s in $e/salary satisfies $s > 1000
return $d
```

Joins

Using FLWOR expressions we can join data from multiple sources

```
for $d in doc("departments.xml")//dept_no
let $e := doc("employees.xml")//employee[dept_no = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
    <large_dept>
        {$d}
        <size> {count($e)} </size>
        <avg_salary> {avg($e/salary)} </avg_salary>
    </large_dept>
```

Joins

```
for $i in doc("order.xml")//item
let $n := doc("catalog.xml")//product[number = $i/@num]/name
return
    <item  num = "{$i/@num}"
           name = "{$n}"
           quantity = "{$i/@quantity}"/>
```

```
<catalog>
    <product dept="D1">
        <number> 130 </number>
        <name> N1 </name>
    </product>
    <product dept="D2">
        <number> 230 </number>
        <name> N2 </name>
    </product>
</catalog>
```

```
<order>
    <item dept="D1"  num="130"  quantity="5"/>
    <item dept="D2"  num="230"  quantity="10"/>
</order>
```

```
<item num="130" name="N1" quantity="5"/>

<item num="230" name="N2" quantity="10"/>
```

Aggregating Values

```
for $d in distinct-values(doc("order.xml")//item/@dept)
let $i := doc("order.xml")//item[@dept = $d]
order by $d descending
return <department  name = "{$d}"  totalQuantity = "{sum($i/@quantity)}"/>
```

```
<order>
    <item dept="D1"  num="130"  quantity="5"/>
    <item dept="D2"  num="230"  quantity="7"/>
    <item dept="D1"  num="100"  quantity="6"/>
    <item dept="D2"  num="330"  quantity="10"/>
</order>
```

```
<department name="D2"  totalQuantity="17"/>

<department name="D1"  totalQuantity="11"/>
```

Sum Up

- What is XQuery?
- XQuery at First Glance
- FLWOR Expressions
- Element Constructors
- List, Conditional and Quantified Expressions
- Joins
- Aggregating Values