B4M36DS2, BE4M36DS2: **Database Systems 2**

http://www.ksi.mff.cuni.cz/~svoboda/courses/201-B4M36DS2/

Lecture 3

XML Databases: XPath

Martin Svoboda
martin.svoboda@fel.cvut.cz

12. 10. 2020

Charles University, Faculty of Mathematics and Physics
Czech Technical University in Prague, Faculty of Electrical Engineering

Lecture Outline

XQuery and XPath

- Data model
- Query expressions
  - Paths
  - Sequence constructor
  - For expressions
  - Let expressions
  - Comparisons

XPath

XML Path Language

Introduction

XPath = *XML Path Language*

- **Navigation in an XML tree,**
  selection of nodes by a variety of criteria
- Versions: 1.0 (1999), 2.0 (2010), 3.0 (2014), **3.1** (March 2017)
- W3C recommendation
  - https://www.w3.org/TR/xpath-31/

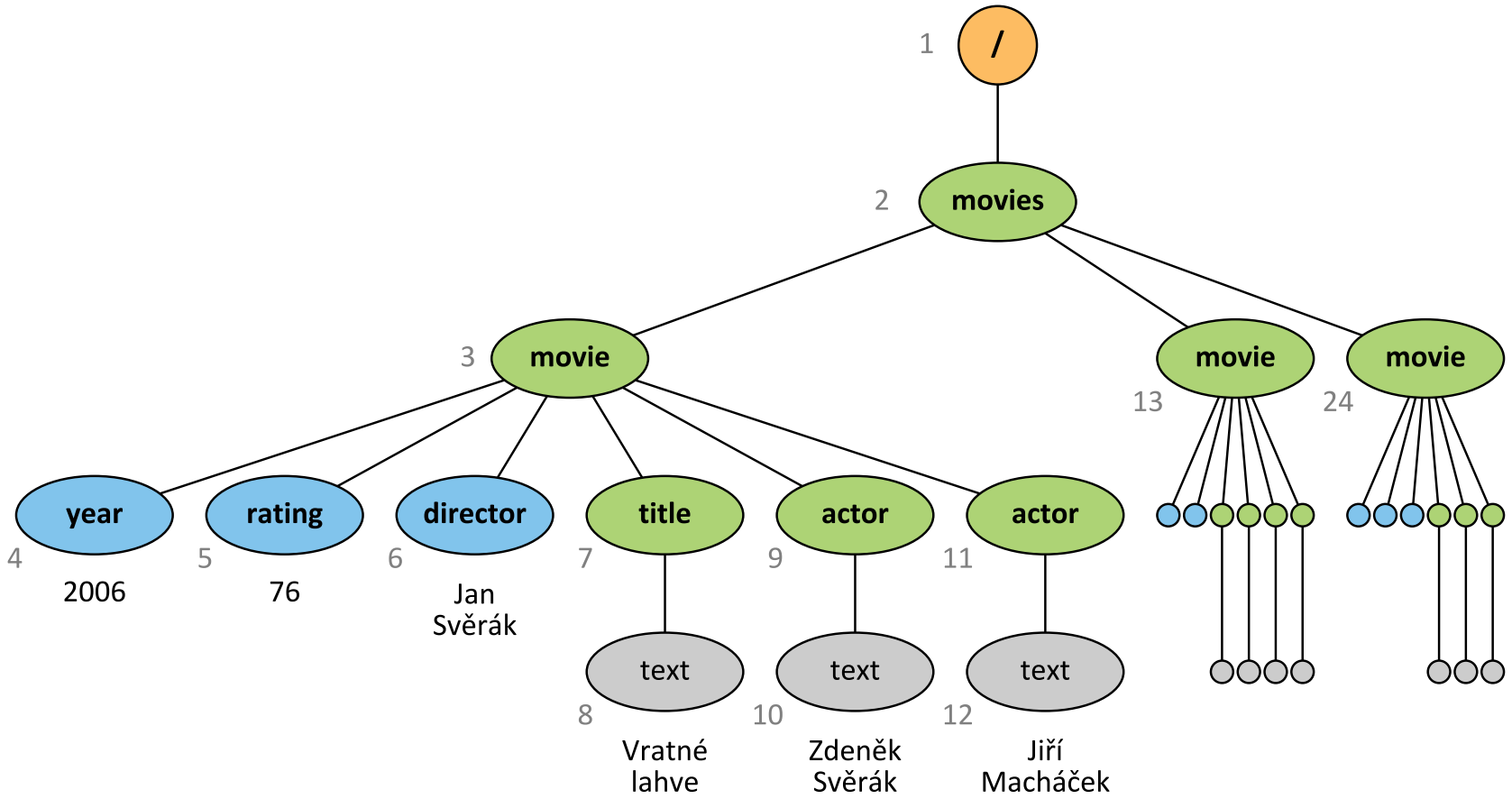
XQuery = *XML Query Language*

- **Complex functional query language**
- Contains XPath
- Versions: 1.0 (2007), 3.0 (2014), **3.1** (March 2017)
- W3C recommendation
  - https://www.w3.org/TR/xquery-31/

Sample Data

```
<?xml version="1.1" encoding="UTF-8"?>
<movies>
  <movie year="2006" rating="76" director="Jan Svěrák">
    <title>Vratné lahve</title>
    <actor>Zdeněk Svěrák</actor>
    <actor>Jiří Macháček</actor>
  </movie>
  <movie year="2000" rating="84">
    <title>Samotáři</title>
    <actor>Jitka Schneiderová</actor>
    <actor>Ivan Trojan</actor>
    <actor>Jiří Macháček</actor>
  </movie>
  <movie year="2007" rating="53" director="Jan Hřebejk">
    <title>Medvídek</title>
    <actor>Jiří Macháček</actor>
    <actor>Ivan Trojan</actor>
  </movie>
</movies>
```

Sample Data


1
/
2
movies
3
movie
13
movie
24
movie
year
4
2006
rating
5
76
director
6
Jan
Svěrák
title
7
actor
9
actor
11
text
8
Vratné
lahve
text
10
Zdeněk
Svěrák
text
12
Jiří
Macháček

Data Model

XDM = *XQuery and XPath Data Model*

- **XML tree** consisting of **nodes** of different kinds
  - Document, element, attribute, text, …
- **Document order** / reverse document order
  - The order in which nodes appear in the XML file
    - I.e. nodes are numbered using a **pre-order depth-first traversal**

Query result

- Each query expression is evaluated to a sequence

Data Model

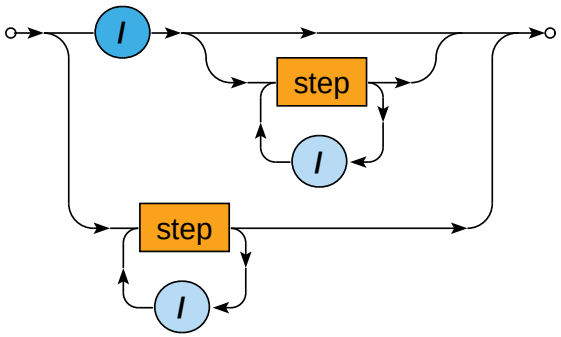
Sequence = **ordered collection of nodes and/or atomic values**

- Can be **empty**
  - E.g.: `()`
- Automatically **flattened**
  - E.g.: `(1, (), (2, 3), (4))` ⇔ `(1, 2, 3, 4)`
- Standalone items are treated as singleton sequences
  - E.g.: `1` ⇔ `(1)`
- Can be **mixed**
  - But usually just nodes, or just atomic values
- **Duplicate items** are allowed
  - More precisely…
    - Duplicate nodes are removed
    - Duplicate atomic values are preserved

Path Expressions

Path expression

- Describes navigation within an XML tree
- Consists of individual navigational steps



- **Absolute** paths = path expressions starting with /
  - Navigation starts at the document node
- **Relative** paths
  - Navigation starts at an explicitly specified node / nodes

Path Expressions

Examples

Absolute paths

```
/
```

```
/movies
```

```
/movies/movie
```

```
/movies/movie/title/text()
```

```
/movies/movie/@year
```

Relative paths

```
actor/text()
```

```
@director
```

Path Expressions

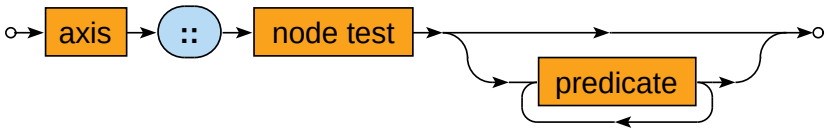
Evaluation of path expressions

- Let P be a **path expression**
- Let C be an initial **context set**
  - If P is **absolute**, then C contains just the document node
  - Otherwise (i.e. P is **relative**) C is given by the user or context
- If P does not contain any step
  - Then C is the **final result**
- Otherwise (i.e when P contains **at least one step**)
  - Let S be the **first step**, P' the **remaining steps** (if any)
  - Let $C' = \{\}$
  - For each node $u \in C$:
    evaluate S with respect to u and add the result to C'
  - Evaluate P' with respect to C'

Path Expressions

Step

- Each step consists of (up to) 3 components



- **Axis**
  - Specifies the **relation of nodes** to be selected for a given node u
- **Node test**
  - **Basic condition** the selected nodes must further satisfy
- **Predicates**
  - **Advanced conditions** the selected nodes must further satisfy

Path Expressions: Axes

Axis

- Specifies the relation of nodes to be selected for a given node

Forward axes

- `self`, `child`, `descendant(-or-self)`, `following(-sibling)`
- The order of the nodes corresponds to the document order

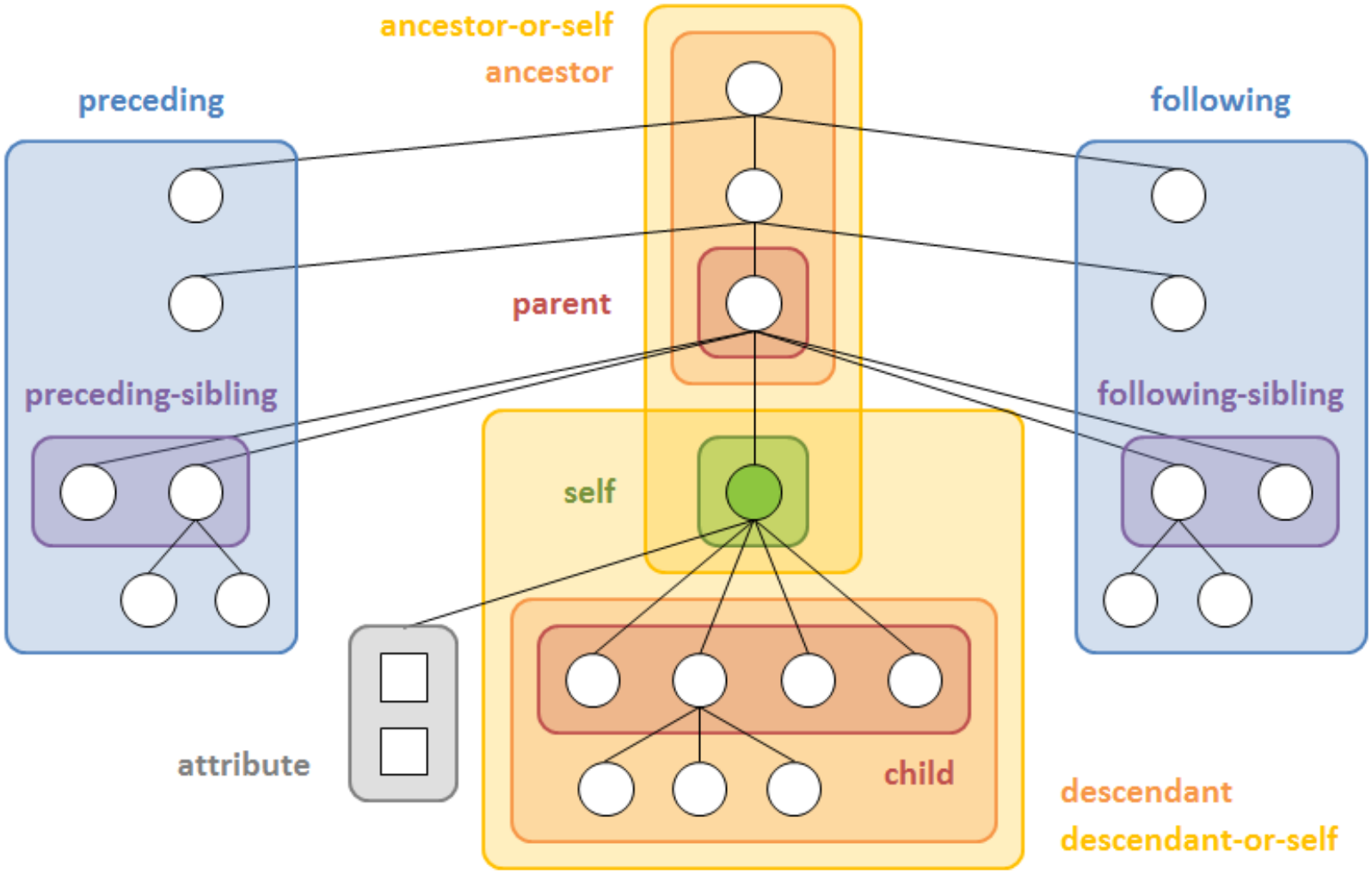
Reverse axes

- `parent`, `ancestor(-or-self)`, `preceding(-sibling)`
- The order of the nodes is reversed

Attribute axis

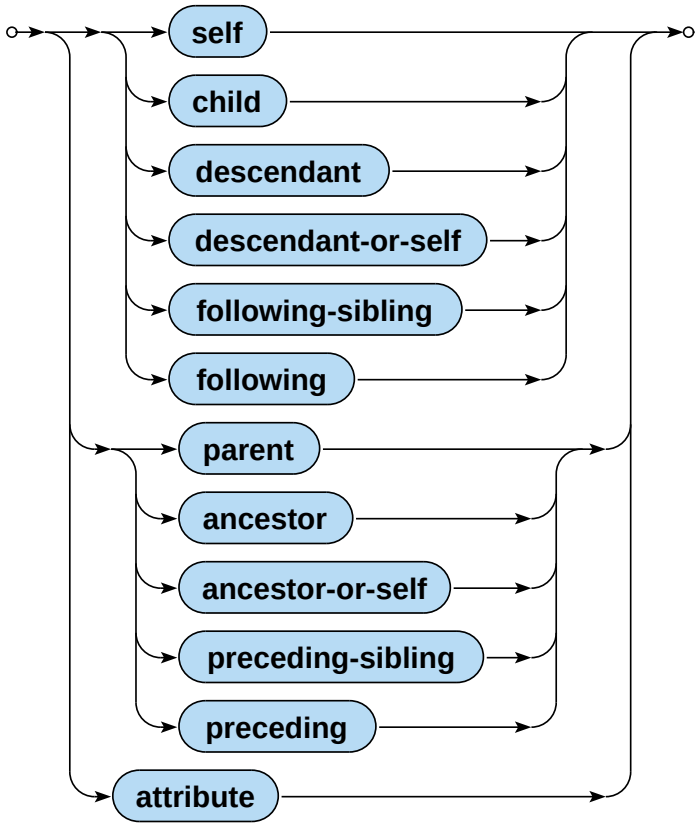
- `attribute` – the only axis that selects attributes

Path Expressions: Axes



Path Expressions: Axes

Available **axes**



Path Expressions

Examples

Axes

```
/child::movies
```

```
/child::movies/child::movie/child::title/child::text()
```

```
/child::movies/child::movie/attribute::year
```

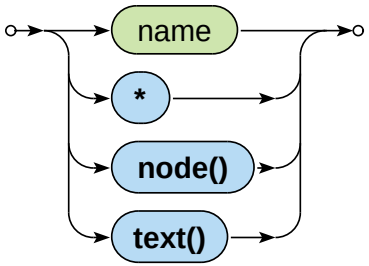
```
/descendant::movie/child::title
```

```
/descendant::movie/child::title/following-sibling::actor
```

Path Expressions: Node Tests

Node test

- Filters the nodes selected by the axis using basic tests



Available node tests

- `name` – all elements / attributes with a given name
- `*` – all elements / attributes
- `node()` – all nodes (i.e. no filtering takes place)
- `text()` – all text nodes

Path Expressions

Examples

Node tests

```
/movies
```

```
/child::movies
```

```
/descendant::movie/title/text()
```

```
/movies/*
```

```
/movies/movie/attribute::*
```

Path Expressions: Predicates

Predicate

- Further filters the nodes based on advanced conditions

- When **more predicates** are provided, they must all be satisfied

Commonly used conditions

- **Comparisons**
- **Path expressions**
  - Treated as `true` when evaluated to a non-empty sequence
- **Position tests**
  - Based on the order as defined by the axis, starting with 1
- **Logical expressions**: `and`, `or`, `not` connectives

Path Expressions

Examples

Predicates

```
/movies/movie[actor]
```

```
/movies/movie[actor]/title/text()
```

```
/descendant::movie[count(actor) >= 3]/title
```

```
/descendant::movie[@year > 2000 and @director]
```

```
/descendant::movie[@director][@year > 2000]
```

```
/descendant::movie/actor[position() = last()]
```

Path Expressions: Abbreviations

Multiple (mostly syntax) **abbreviations** are provided

- …/… (i.e. no axis is specified) ⇔ …/`child::`…
- …/`@`… ⇔ …/`attribute::`…
- …/`.`… ⇔ …/`self::node()`…
- …/`..`… ⇔ …/`parent::node()`…
- …`//`… ⇔ …`/descendant-or-self::node()/`…
- …/…`[number]`… ⇔ …/…`[position() = number]`…

Path Expressions

Examples

Abbreviations

```
/movie/title
```

```
/child::movie/child::title
```

```
/movie/@year
```

```
/child::movie/attribute::year
```

```
/movie/actor[2]
```

```
/child::movie/child::actor[position() = 2]
```

```
//actor
```

```
/descendant-or-self::node()/child::actor
```

Path Expressions: Conclusion

Path expressions

- Absolute / relative

Step components

- Axis
- Node test
- Predicates

Path expression result

- Evaluated **from left to right, step by step**
- **Result of the entire path expression** is the result of its **last step**
- Nodes are ordered in the **document order**
- **Duplicate nodes** are removed (based on the identity of nodes)

Comparison Expressions

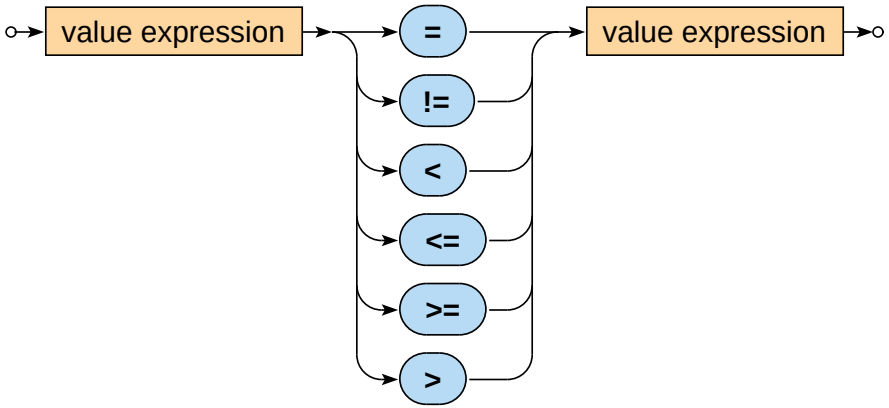
Comparisons

- **General** comparisons
  - Two sequences of values are expected to be compared
  - =, !=, <, <=, >=, >
  - E.g.: `(0,1) = (1,2)`
- **Value** comparisons
  - Two standalone values (singleton sequences) are compared
  - `eq`, `ne`, `lt`, `le`, `ge`, `gt`
  - E.g.: `1 lt 3`
- **Node** comparisons
  - `is` – tests identity of nodes
  - <<, >> – test positions of nodes (preceding, following)
  - Similar behavior as in case of value comparisons

Comparison Expressions

General comparison (**existentially quantified** comparisons)

- **Both the operands can be evaluated to sequences of values of any length**
- The result is `true` if and only if there exists at least one pair of individual values satisfying the given relationship



Comparison Expressions

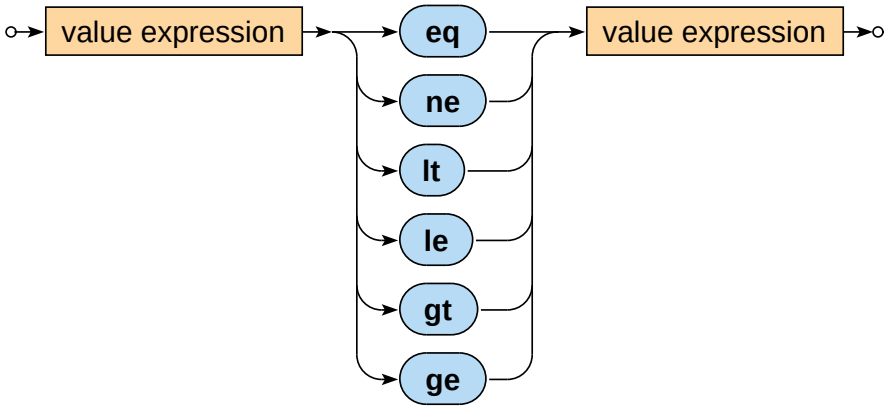
General comparison: examples

- $[\![\texttt{(1) < (2)}]\!] = \texttt{true}$
- $[\![\texttt{1 < (2)}]\!] = \texttt{true}$
- $[\![\texttt{(1) < (1,2)}]\!] = \texttt{true}$
- $[\![\texttt{(1) < ()}]\!] = \texttt{false}$
- $[\![\texttt{(0,1) = (1,2)}]\!] = \texttt{true}$
- $[\![\texttt{(0,1) != (1,2)}]\!] = \texttt{true}$

Comparison Expressions

Value comparison

- **Both the operands are expected to be evaluated to singleton sequences**
  - Then these values are mutually compared in a standard way
- Empty sequence () is returned…
  - when at least one operand is evaluated to an empty sequence
- Type error is raised…
  - when at least one operand is evaluated to a longer sequence



Comparison Expressions

Value comparison: examples

- $[\![$ `(1) le (2)` $]\!] =$ `true`
- $[\![$ `1 le (2)` $]\!] =$ `true`
- $[\![$ `(1) le ()` $]\!] =$ `()`
- $[\![$ `(1) le (1,2)` $]\!] \Rightarrow$ `error`
- $[\![$ `() le (1,2)` $]\!] =$ `()`

Comparison Expressions

Value and general comparisons

- Atomization of values – takes place automatically
  - Atomic values are preserved untouched
  - Nodes are transformed to atomic values
- In particular…
  - **Element node is transformed to a string with concatenated text values it contains** (even indirectly)
    - E.g.: `<movie year="2006">Vratné lahve</movie>` is atomized to a string `Vratné lahve`
    - Note that attribute values are not included!
  - **Attribute node is transformed to its value**
  - **Text node is transformed to its value**

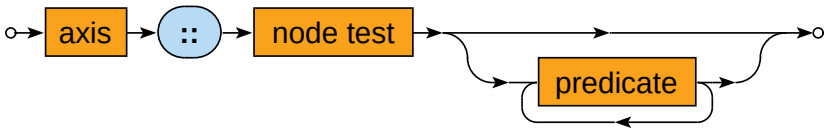
Comparison Expressions

Value and general comparisons: examples

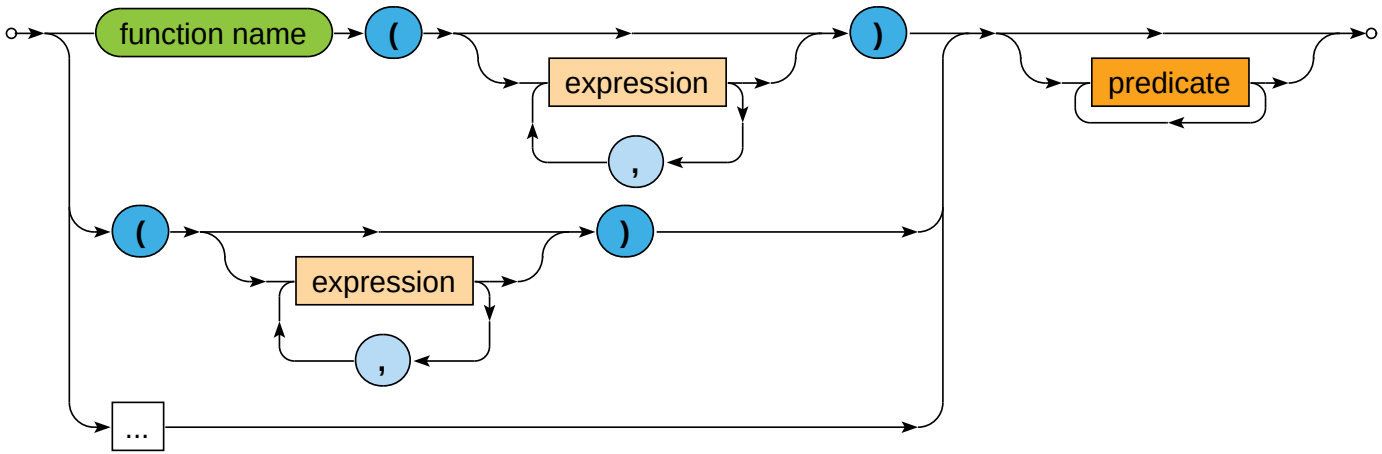
- $[\![$ `<a>5</a> eq <b>5</b>` $]\!] =$ `true`
- $[\![$ `<a>12</a> = <a><b>1</b>2</a>` $]\!] =$ `true`
- $[\![$ `<a t="1">5</a> lt 3` $]\!] =$ `false`

Path Operator

Navigational **steps** in path expressions



- Extended functionality (XPath 2.0)
  - **Function calls**, **sequence constructors**, …
  - Must not yield mixed sequences (nodes and atomic values)



Path Operator

Examples

Numbers of actors who appeared in individual movies

```
/movies/movie/count(actor)
```

```
2
3
2
```

Flat sequence of interleaved movie titles and years of filming

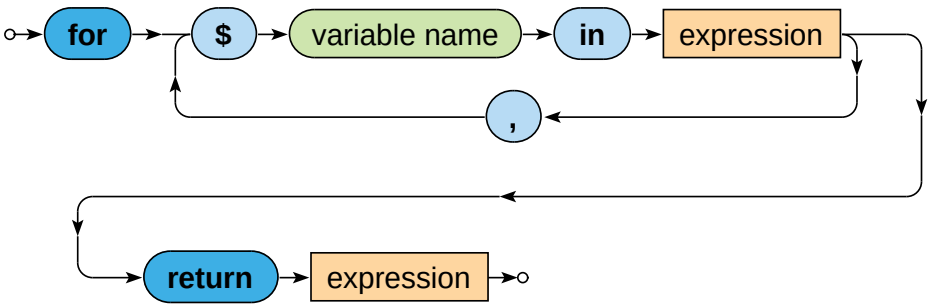
```
/movies/movie/(title, @year)/data(.)
```

```
2006
Vratné lahve
2000
Samotáři
2007
Medvídek
```

For Expressions

For expression (XPath 2.0, XQuery 1.0 FLWOR)

- **Allows for the iteration over items of an input sequence** / tuples of items of more input sequences



- **Returns a sequence containing results of the `return` clause** evaluated for each input item / tuple of items

For Expressions

Example

Numbers of actors who appeared in movies filmed in *2000* or later

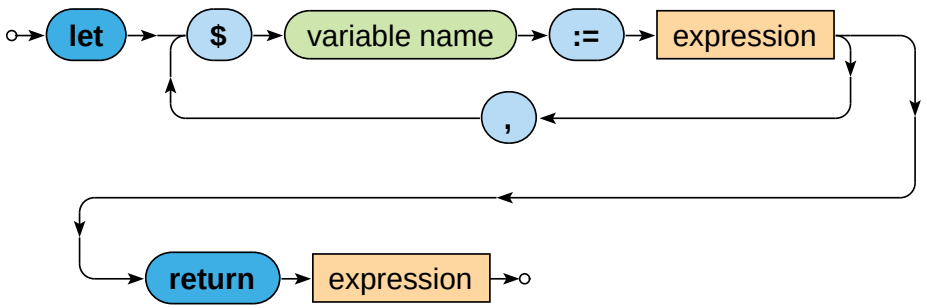
```
for $m in //movie[@year >= 2000]
return count($m/actor)
```

```
2
3
2
```

Let Expressions

Let expression (XPath 2.0, XQuery 1.0 FLWOR)

- **Allows to associate one or more variables with values**
  - $ symbol is used to denote variables
- These variables can then be accessed



- Returns the result of the evaluated `return` clause

Let Expressions

Example

Find titles of movies with at least average rating

```
let $a := avg(//movie/@rating)
return //movie[@rating >= $a]/title/text()
```

```
Vratné lahve
Samotáři
```

Range Expressions

Range expression (XPath 2.0)

- Allows to construct a **sequence of consecutive integers**

Example

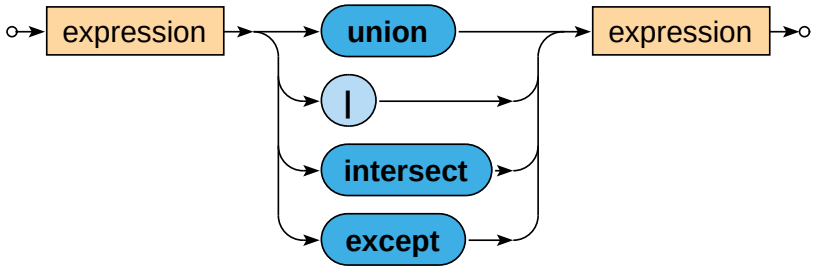
```
1 to count(//movie)
```

```
1
2
3
```

Set Operations

Traditional **set operations** (XPath 2.0)

- Only applicable on sequences of nodes (not atomic values)!
- Duplicate nodes are removed



Union

- Nodes that occur in either of the operands

Intersection

- Nodes that occur in both the operands

Difference

- Nodes that occur in the first operand but not in the second one

Additional Expressions

Further XPath expressions

- **Conditional expressions**
- **Quantified expressions**
- Primary expressions
  - Function calls, variable references, sequence constructors, …

Lecture Conclusion

XPath

- **Path expressions**
  - Absolute / relative paths
  - Axes, node tests, predicates
- **Comparison expressions**
  - General / value / node comparisons
- For expressions
- Let expressions
- Range expressions
- Set operations
  - Union, intersection, difference