MIE-PDB: **Advanced Database Systems**

Practical Class 3:

XPath and XQuery

31. 3. 2017

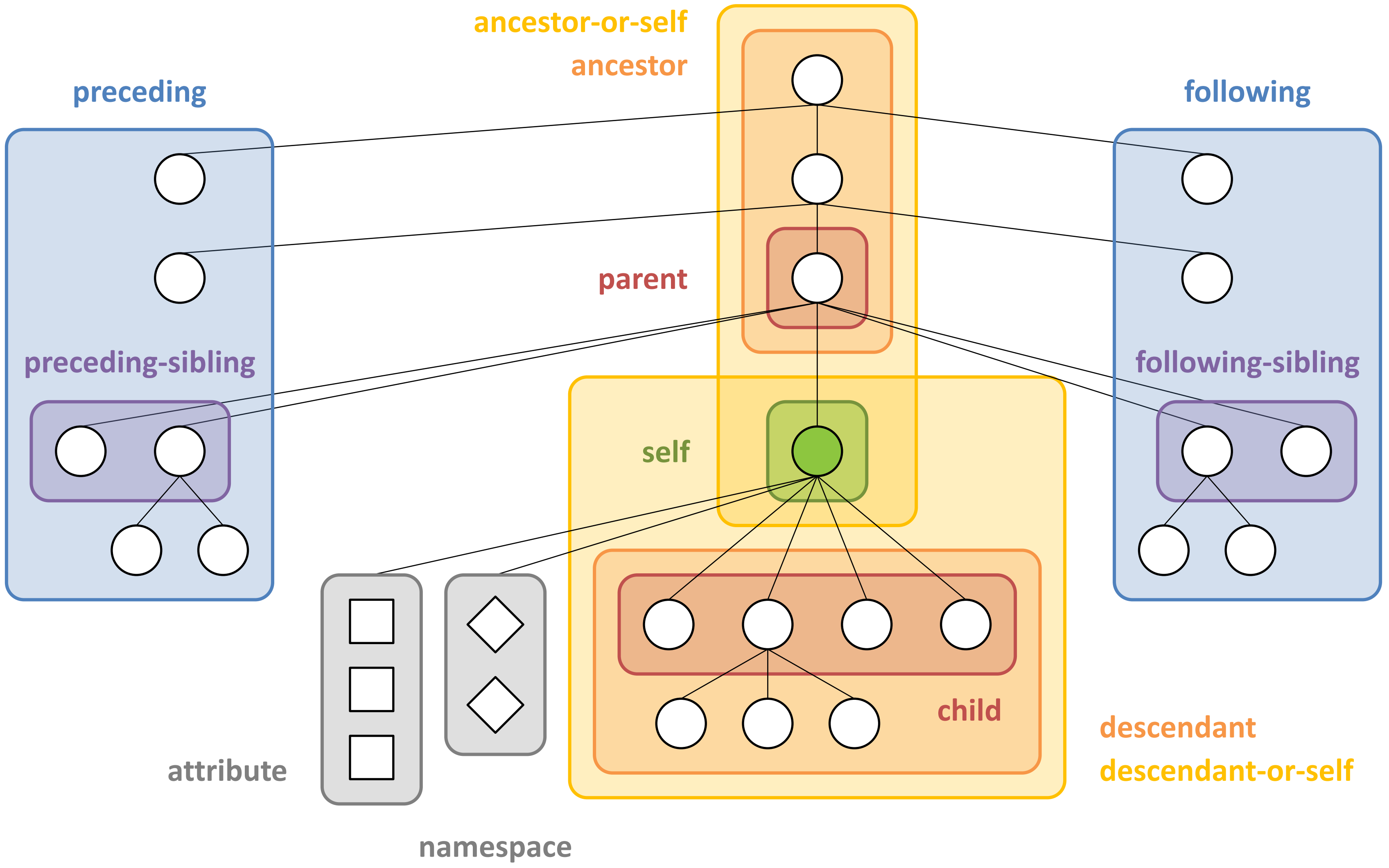
Martin Svoboda
svoboda@ksi.mff.cuni.cz

http://www.ksi.mff.cuni.cz/~svoboda/courses/2016-2-MIE-PDB/

Path Expressions

- **Paths**
  - Absolute
    - `/Step`$_1$`/Step`$_2$`/…/Step`$_N$
  - Relative
    - `Step`$_1$`/Step`$_2$`/…/Step`$_N$
- **Steps**
  - `axis::test predicate`$_1$ `predicate`$_2$ `…`

Axes


ancestor-or-self
ancestor
preceding
following
parent
preceding-sibling
following-sibling
self
child
descendant
descendant-or-self
attribute
namespace

Axes

- **Forward axes**
  - self, child, descendant(-or-self), following(-sibling)
- **Reverse axes**
  - parent, ancestor(-or-self), preceding(-sibling)
- **Attributes**
  - attribute
- **Namespace declarations**
  - namespace

Node Tests

- Tests
  - `node()` – all nodes selected by the axis
  - `text()` – all texts nodes
  - `*` – all elements / attributes selected by the axis
  - *name* – elements / attributes of the given *name*

Abbreviations

- **Abbreviations**
  - …/… <=> …/child::…
  - …/@… <=> …/attribute::…
  - …/.… <=> …/self::node()…
  - …/..… <=> …/parent::node()…
  - …//… <=> …/descendant-or-self::node()/…

Predicates

- **Predicates**
  - Path expressions: both relative and absolute
  - Comparisons: $= \neq < \leq \geq >$
  - Positions

Functions

- **A few useful functions**
  - `position(), last()`
  - `count(), sum()`
  - `avg(), min(), max()`
  - `data()`
  - `distinct-values()`
  - …

Assignment 1

- Express the following XPath queries
    - Use *employees.xml*
  - Return all employees (with their entire subtrees)
  - Return surnames of all employees (just text content)
  - Return these surnames without duplicate values
  - Return salaries of all employees with surname *Smith*

Assignment 2

- Express the following XPath query
  - Use *employees.xml*
  - Return e-mail addresses of all employees with salaries above the average

Assignment 3

- Express the following XPath query
  - Use *departments.xml*
  - Return identifiers of all departments with no directly subordinated employees

Assignment 4

- Express the following XPath query
  - Use *departments.xml*
  - Return the name of the very last department in the whole input document

Assignment 5

- Express the following XPath query
  - Use *departments.xml*
  - Return identifiers of all departments that have at least two subdepartments (even recursively)

Assignment 6

- Express the following XPath query
    - Use *departments.xml*
  - Return identifier of the top level department that involves a given particular department (e.g. *D1.2.1*)

XQuery

Expressions

- XQuery expressions
  - **XPath** path expressions
  - Computed and direct **constructors**
  - **FLWOR** expressions
  - **Conditional expressions**
  - Universal and existential **quantifiers**

Constructors

- **Direct constructors**

```
<element>
  <element attribute="value">
    text
    {  nested XQuery expression  }
  </element>
  <!-- comment -->
  <?target data?>
</element>
```

Constructors

- **Computed constructors**
  - `document` { *content* }
  - **`element`** *name* { *content* }
  - **`attribute`** *name* { *value* }
  - **`text`** { *text* }
  - `comment` { *text* }
  - `processing-instruction` { *target* } { *content* }

FLWOR Expressions

- FLWOR clauses
    - (ForClause | LetClause)+ WhereClause? OrderByClause? ReturnClause
  - **FOR** clause: items selection
  - **LET** clause: auxiliary assignments
  - **WHERE** clause: filtering conditions
  - **ORDER BY** clause: result ordering
  - **RETURN** clause: result construction

FLWOR Expressions

- General FLWOR pattern
  - `for $item in` *sequence*, …
  - `let $variable :=` *expression*, …
  - `where` *condition*
  - `order by` *criterion*, …
  - `return` *result*

Other Constructs

- **Conditional expressions**
  - **`if`** `(`*condition*`)` **`then`** *expression* **`else`** *expression*
- **Quantified expressions**
  - Existential quantifier
    - **`some`** `$item` **`in`** *sequence* **`satisfies`** *condition*
  - Universal quantifier
    - **`every`** `$item` **`in`** *sequence* **`satisfies`** *condition*

Comparisons

- **Value comparison**
  - eq, ne, lt, le, ge, gt
- **General comparison**
  - =, !=, <, <=, >=, >
- **Node comparison**
  - is, <<, >>

Assignment 7

- Express the following XQuery query
  - Use *employees.xml*
  - Return a list of employees with their identifiers (transformed from attributes to subelements), and both first and last names
  - Exclude employees having last name *Smith*

```
<employee>
  <number>E4</number>
  <firstName>Peter</firstName>
  <lastName>Brown</lastName>
</employee>
…
```

Assignment 8

- Express the following XQuery query
    - Use *employees.xml*
  - Return a sequence of full names (concatenated first and last names) of all the employees

```
<employee>John Smith</employee>
…
```

Assignment 9

- Express the following XQuery query
  - Use *employees.xml*
  - Return a sequence of e-mail addresses of all employees with salaries greater than *2300*
  - Ignore employees that work directly in *D1.1*
  - Sort the output with respect to salaries and then surnames in reverse order

```
taylor@co.org
brown@co.org
```

Assignment 10

- Express the following XQuery query
  - Use *departments.xml*
  - Return a flat list of names of all departments
  - Add attributes with department identifiers and overall numbers of all their employees (including indirect)
  - Sort the output according to these numbers

```
<department employees="0" id="D1.2.1"/>
<department employees="0" id="D2">
  Accounting
</department>
…
```

Assignment 11

- Express the following XQuery query
  - Use *departments.xml*
  - Return full names of managers with the maximal overall number of employees (including indirect) they are responsible for

```
<manager>John Smith</manager>
```

Assignment 12

- Express the following XQuery query
  - Use *departments.xml*
  - Return an XHTML table with a list of identifiers, names, and managers (even indirect when known) of all the departments (including nested ones)
  - Sort the list according to department identifiers

```
<table>
   <tr><td>D1</td><td>Production</td><td>John Smith</td></tr>
   …
   <tr><td>D1.2.1</td><td></td><td>Arthur Taylor</td></tr>
   <tr><td>D2</td><td>Accounting</td><td><i>Unknown</i></td></tr>
</table>
```