MI-PDB, MIE-PDB: **Advanced Database Systems**

Practical class 4:

XML Query Language (XQuery)

5. 4. 2016

Martin Svoboda
svoboda@ksi.mff.cuni.cz

http://www.ksi.mff.cuni.cz/~svoboda/courses/2015-2-MIE-PDB/

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 1

- 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 2

- 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 3

- 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 4

- 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 5

- 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 6

- 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>
```

Assignment 7

- Express the following XQuery query
  - Use *departments.xml*
  - Create a test for an integrity constraint ensuring that all departments with at least *2* employees (including indirect) have a manager (even indirect)
    - Return <ok/> when no issues detected
    - Return a list of invalid departments otherwise