# 2. Unit: Querying with XPath

*Information about the XML course can be found at*
`http://www.stud.informatik.uni-goettingen.de/xml-lecture`.
*These exercises are supposed to be solved using XPath, not with XQuery*

**Exercise 2.1 (XPath: Mondial)**

a) Find out which countries are neighbors of Russia and have more than 10 million inhabitants.

```
(: country -> check if it is a neighbor of Russia
   (using a subquery in a condition) -> name      :)
//country
   [border/id(@country)/name='Russia' and .//population > 10000000]
  /population

(: Russia -> neighbors -> names :)
//country[name='Russia']/border/id(@country)[population>10000000]/name
```

b) Which countries are members of the NATO? Return the countries' names.

```
(: NATO -> members -> name :)
//organization[abbrev="NATO"]/members/id(@country)/name/text()

(: country -> check NATO membership in subquery -> name :)
//country[id(@memberships)/abbrev='NATO']/name/text()
```

c) Give the names of countries with a neighbor country with a mountain of 4000 m and higher.

```
//mountain[height>=4000]/located/id(@country)/border/id(@country)/name
```

**Exercise 2.2 (XPath: Hamlet)**

a) List all scenes with less than 10 persons speaking by their titles (duplicates allowed).

```
(: note that it counts the SPEAKER nodes in the scene, not the
   different speakers! :)
//SCENE[count(.//SPEAKER)<10]/TITLE
```

b) What is the title of the third scene of the act with a scene called 'The Queen's closet'?

```
(: Interessant, weil wegen Sonderzeichen
//ACT[SCENE[contains(TITLE, "The Queen's closet")]]/SCENE[3]/TITLE
(: or :)
 //SCENE[contains(TITLE,"The Queen's closet")]/parent::ACT/SCENE[3]/TITLE
```

c) Who are the persons speaking in both the first and the last act?

```
#interessanter waere nur im letzten nicht im ersten, aber not laeuft nicht)
//ACT[position()=last()]//SPEECH[SPEAKER=//ACT[1]//SPEAKER]/SPEAKER
```

d) What happens (stage directive) directly before King Claudius says: "Part them; they are incensed."?

```
(: note: preceding-sibling is a backward axis! :)
//SPEECH[SPEAKER="KING CLAUDIUS"and LINE="Part them; they are incensed."]
  //preceding-sibling::STAGEDIR[1]
```

**Exercise 2.3 (XPath: Mondial (2))**

a) Which (country) capitals are located at a river, sea or lake? Give their names.

```
//country/id(@capital)[located_at/@watertype]/name
```

b) What are the names of those cities located next to a lake?

```
//country/id(@capital)[located_at/@watertype="lake"]/name
```

c) What are the names of all lakes with no city located next to it?

```
//lake[not (@id = //city/located_at/@lake)]
```

d) What are the names of all rivers flowing through (at least) one capital?

```
//country/id(@capital)/located_at/id(@river)/name
```

e) Find all "german leaf-nodes", which means all element nodes that are sub-nodes of the country-element of Germany and have no children.

```
//country[name="Germany"]//*[count(./*) = 0]
```

f) In Mondial, there exist city elements as sub-elements of province elements, and city elements as sub-elements of country elements. Are there any other city elements?

```
(: liefert nur country- und province-Elemente. :)
/mondial//*[(./city)]/name()
```

**Exercise 2.4 (XML → RDB)** A possible model for storing (or indexing) XML data is based on relational tables (we ignore namespaces here).

(1) a table for storing element and text nodes:
   - first column: node identifier in Dewey Notation (e.g., 1.2.6.3 for the third child of the sixth child of the second child of the root node),
   - second column: element type (or "text"),
   - third column: text content (or NULL),
   - forth column: number of the node when enumerated in *preorder*,
   - fifth column: number of the node when enumerated in *postorder*.

(2) a table for storing attribute nodes:
   - first column: dewey identifier of the node where the attribute belongs to,
   - second column: attribute name,
   - third column: value.

a) Discuss whether the above information is sufficient for storing an XML document. Give the tables for a small example document.

b) Discuss what must be done when an update (modification, insertion, deletion) is executed.

c) Given a "current" element somewhere in the tree, characterize the following sets of nodes (i.e., the nodes that result from navigating along the different axes) by their dewey notation and, if possible, by their preorder/postorder information:
   - the parent
   - all children

- all descendants
- all ancestors
- all siblings
- all predecessors according to document order
- all successors according to document order
- all attributes

---

Consider the following XML tree (also available on the Web page):

---

```
<mondial>
 <country car_code="F" area="547030" capital="cty-France-Paris">
  <name>France</name>
  <population>58317450</population>
  <population_growth>0.34</population_growth>
  <languages percentage="100">French</languages>
  <province capital="cty-France-Strasbourg">
   <name>Alsace</name>
   <city id="cty-France-Strasbourg">
    <name>Strasbourg</name>
    <population year="90">252338</population>
   </city>
   <city>
    <name>Mulhouse</name>
    <population year="90">108357</population>
   </city>
  </province>
  <province capital="cty-France-Paris">
   <name>Ile de France</name>
   <city id="cty-France-Paris">
    <name>Paris</name>
    <population year="90">2152423</population>
   </city>
   :
  </province>
  :
 </country>
 <country car_code="D" area="356910" capital="cty-Germany-Berlin">
  <name>Germany</name>
  <population>83536115</population>
  <population_growth>0.67</population_growth>
  <languages percentage="100">German</languages>
  <province>
   <name>Baden Wurttemberg</name>
   <city>
    <name>Stuttgart</name>
    <population year="95">588482</population>
   </city>
   <city>
    <name>Karlsruhe</name>
    <population year="95">277011</population>
   </city>
```

```
  </province>
  :
  <province>
   <name>Berlin</name>
   <city id="cty-Germany-Berlin">
    <name>Berlin</name>
    <population year="95">3472009</population>
   </city>
  </province>
  :
 </country>
 <country car_code="H" area="93030" capital="cty-Hungary-Budapest">...</country>
 :
</mondial>
```

The resulting table is as follows:

```
Elements:
Dewey Nr     Element type| text contents |preorder|postorder

1            mondial                         1
1.1          country                         2    150
1.1.1        name                            3      2
1.1.1.1                     "France"         4      1
1.1.2        population                      5      4
1.1.2.1                     58317450         6      3
1.1.3        population_growth               7      6
1.1.3.1                     0.34             8      5
1.1.4        languages                       9      8
1.1.3.1                     "French"        10      7
1.1.5        province                       11     21
1.1.5.1      name                           12     10
1.1.5.1.1                   "Alsace"        13      9
1.1.5.2      city                           14     15
1.1.5.2.1    name                           15     12
1.1.5.2.1.1                 "Strasbourg"    16     11
1.1.5.2.2    population                     17     14
1.1.5.2.2.1                 252338          18     13
1.1.5.3      city                           19     20
1.1.5.3.1    name                           20     17
1.1.5.3.1.1                 "Mulhouse"      21     16
1.1.5.3.2    population                     22     19
1.1.5.3.2.1                 108357          23     18
1.1.6        province                       24     29
1.1.6.1      name                           25     23
1.1.6.1.1                   "Ile de France" 26     22
1.1.6.2      city                           27     28
1.1.6.2.1    name                           28     25
1.1.6.2.1.1                 "Paris"         29     24
1.1.6.2.2    population                     30     27
1.1.6.2.2.1                 2152423         31     26
  :
```

```
Note: we assume that Germany is node No 152 in preorder
enumeration.
That means, that Node no.1 is ‘‘mondial’’ and France
consists of 150 nodes (including the country node for it).
Thus, in postorder enumeration, the country node for
France has number 150.

1.2          country                       152
1.2.1        name                          153    152
1.2.1.1                    "Germany"       154    151
1.2.2        population                    155    154
1.2.2.1                    83536115        156    153
1.2.3        population_growth             157    156
1.2.3.1                    0.67            158    155
1.2.4        languages                     159    158
1.2.4.1                    "German"        160    157
1.2.5        province                      161    171
1.2.5.1      name                          162    160
1.2.5.1.1                  "Baden Wurttemberg" 163 159
1.2.5.2      city                          164    170
1.2.5.2.1    name                          165    162
1.2.5.2.1.1                "Stuttgart"     166    161
1.2.5.2.2    population                    167    164
1.2.5.2.2.1                588482          168    163
1.2.5.3      city                          169    169
1.2.5.3.1    name                          170    166
1.2.5.3.1.1                "Karlsruhe"     171    165
1.2.5.3.2    population                    172    168
1.2.5.3.2.1                277011          173    167
1.2.6        province                      174      ?
  :
1.2.7        province                      210
1.2.7.1      name                          211
1.2.7.1.1                  "Berlin"        212
1.2.7.2      city                          213
1.2.7.2.1    name                          214
1.2.7.2.1.1                "Berlin"        215
1.2.7.2.2    population                    216
1.2.7.2.2.1                3472009         217
  :
1.3          country                       389      ?
1.3.1        name                          390    389
1.2.1.1                    "Hungary"       391    388
 :

Attributes:

Parent(Dewey) AttrName Attr Value
1.1        car_code    "F"
1.1        area        "547030"
1.1        capital     "cty-France-Paris"
1.1.4      percentage  "100"
```

```
1.1.5      capital    "cty-France-Strasbourg"
1.1.5.2    id         "cty-France-Strasbourg"
1.1.5.2.2  year       "90"
1.1.6      capital    "cty-France-Paris"
1.1.6.2    id         "cty-France-Paris"
1.1.6.2.2  year       "90"
1.2        car_code   "D"
1.2        area       "356910"
1.2        capital    "cty-Germany-Berlin"
etc.
```

The information is more than sufficient: The *preorder* and *postorder* numbers are not necessary. But they will provide useful search indexes.

Note that there is no reasonable notion for *inorder* traversal (this would be "leftchild-self-rightchild" an is thus only applicable to *binary* trees).

Updates:

- update of text contents: only one update of the first table
- modification, insertion, or deletion of an attribute node: only one update to the second table
- insertion or deletion of an element:
  - change dewey number of all following siblings
  - change preorder and postorder numbers of all nodes with higher numbers

Use of the indexes:

- parent, following-sibling, preceding-sibling: by Dewey Number arithmetics (note that `CREATE TYPE DEWEY` with suitable methods `parent()`, `preceding-sibling()`, `following-sibling()` and an `ORDER` method makes this even easier [note that there cannot be a `MAP` method if the number of children of a node is not restricted]). Use also an index on this column.
- descendants: all nodes $x$ with `self.preorder <` $x$`.preorder and self.postorder >` $x$`.postorder`
- children: descendants+Dewey comparison, or add a `depth` column or `depth` function to the Dewey type.
- ancestors: all nodes $x$ with `self.preorder >` $x$`.preorder and self.postorder <` $x$`.postorder`.
- following: all nodes $x$ with `self.preorder <` $x$`.preorder`
- preceding: all nodes $x$ with `self.postorder >` $x$`.preorder` (note: following and preceding do not include the ancestors, but only nodes that are the roots of trees that *completely* follow/precede `self`!)

Optimizations:

- "gaps" in the preorder or postorder numbering reduce update efforts (since both are only used for comparisons, that does not matter in most cases)
- use relative numbers wrt. the previous sibling or the parent (amortized analysis!). Note that $post(x) = pre(x) - depth(x) + number - of - descendants(x)$
  Proof: when a node is enumerated in postorder, the following nodes have been enumerated before: all "preceding"nodes in preorder except the ancestors on the way back to the root, additionally, all nodes in the subtree rooted in $x$.
- Thus, if for each node, the size of the subtree rooted in it is known, $pre(x)$ and $post(x)$ can be computed as follows:
  - $pre(x)$ = sum of sizes of all subtrees that are rooted in preceding siblings of $x$'s ancestors + #(ancestors), and

 - $post(x)$ = sum of sizes of all subtrees that are rooted in preceding siblings of $x$'s ancestors + sum of sizes of the tree rooted in $x$ - 1.

Further exercise (solutions to be sent to us):

- create suitable tables in SQL, including a Dewey Object Type,
- implement an XSLT stylesheet or a recursive XQuery function or a recursive OraXML PL/SQL function (see later) that traverses an XML tree and creates suitable input statements,
- experiment with SQL queries for the axes.