# BaseX – Processing and Visualizing large XML Instances

http://www.inf.uni-konstanz.de/dbis/research/basex


University of Konstanz
Databases & Information Systems Group
Prof. Dr. Marc H Scholl

Christian Grün
Alexander Holupirek


## 1. Storing XML

```
<addressbook>
 <address id="address0">
  <name>Johannes Schmid</name>
  <street>Badstraße 13</street>
  <code>80327</code>
  <city>Munich</city>
  <country>Germany</country>
 </address>
 <address id="address1">
  <name>Giovanni Stanuti</name>
  <street>Via Pellaro 27</street>
  <code>80142</code>
  <city>Napoli</city>
  <country>Italy</country>
 </address>
 <address id="address2">
  <name>Jean-Claude Risset</name>
  <street>44 Rue St Maximin</street>
  <code>69007</code>
  <city>Lyon</city>
  <country>France</country>
 </address>
 ...
```

**XML Document**

| ID | PAR | SIZE | ATTS | TYPE | TAG/ATTNAME | TEXT/ATTVALUE |
|---|---|---|---|---|---|---|
| 1 | 0 | 724 | 0 | element | addressbook | |
| 2 | 1 | 11 | 1 | element | address | |
| 3 | 2 | | | attribute | id | address0 |
| 4 | 2 | 1 | 0 | element | name | |
| 5 | 4 | | | text | | Johannes Schmid |
| 6 | 2 | 1 | 0 | element | street | |
| 7 | 6 | | | text | | Badstraße 13 |
| 8 | 2 | 1 | 0 | element | code | |
| 9 | 8 | | | text | | 80327 |
| 10 | 2 | 1 | 0 | element | city | |
| 11 | 10 | | | text | | Munich |
| 12 | 2 | 1 | 0 | element | country | |
| 13 | 12 | | | text | | Germany |
| 14 | 1 | 11 | 1 | element | address | |
| 15 | 14 | | | attribute | id | address1 |
| 16 | 14 | 1 | 0 | element | name | |
| 17 | 16 | | | text | | Giovanni Ciampa |
| 18 | 14 | 1 | 0 | element | street | |
| 19 | 18 | | | text | | Via Pellaro 27 |
| 20 | 14 | 1 | 0 | element | code | |
| 21 | 20 | | | text | | 80142 |
| 22 | 14 | 1 | 0 | element | city | |
| 23 | 22 | | | text | | Napoli |
| ... | | | | | | |

**Table Representation**

| ID | PAR | SIZE | ATTS | TYPE | TAG | TEXT |
|---|---|---|---|---|---|---|
| 1 | 0 | 724 | 0 | 0 | 0 | |
| 2 | 1 | 11 | 1 | 0 | 1 | |
| 3 | 2 | | | 2 | 2 | 0 |
| 4 | 2 | 1 | 0 | 0 | 3 | |
| 5 | 4 | | | 1 | | 0 |
| 6 | 2 | 1 | 0 | 0 | 4 | |
| | | | ..... | | | |

Tags/Attribute Names

| | |
|---|---|
| 0 | addressbook |
| 1 | address |
| 2 | id |
| 3 | name |
| 4 | street |
| 5 | code |
| 6 | city |
| 7 | country |
| ... | |

Attribute Values

| | |
|---|---|
| 0 | address0 |
| 1 | address1 |
| ... | |

Texts

| | |
|---|---|
| 0 | Johannes Schmid |
| 1 | Badstraße 13 |
| 2 | 80327 |
| 3 | Munich |
| 4 | Germany |
| ... | |

Fixed-length node storage (128 bits):

| | |
|---|---|
| ID (unique node id) | 40 bits |
| PAR (distance to parent node) | 32 bits |
| ATTPAR (distance to attribute parent) | 8 bits |
| SIZE (#descendants) | 32 bits |
| ATTS (#attributes) | 8 bits |
| TYPE (node kind) | 3 bits |
| TAG (tag/attribute name) | 10 bits |
| TEXT (text/attribute value offset) | 40 bits |

element: ID,PAR,SIZE,ATTS,TYPE,TAG — 125 bits

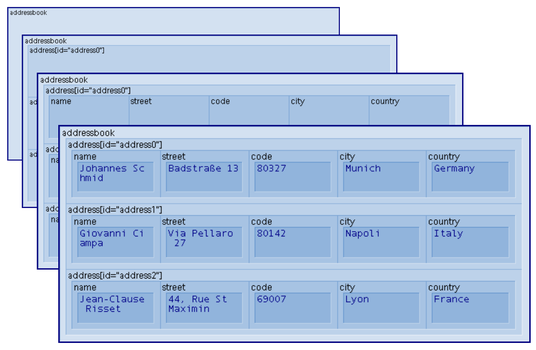
text: ID,PAR,TYPE,TEXT — 115 bits

attribute: ID,ATTPAR,TYPE,TAG,TEXT — 101 bits

**BaseX Storage**

## 2. Creating Visualizations

```
paintMap(root node, view rectangle):


paintMap(nodes, bounds):

   if #nodes == 1:
      paint nodes[0] inside bounds
      newBounds := bounds + fixed offset
      paintMap(xpath(nodes, "child::node()"), newBounds)

   else:
      split := calc. split point in nodes, dependant on
         number of descendants (size attribute)
      orientation := bounds.width < or > bounds.height

      firstBounds := left/upper half of bounds,
         dependant on split and orientation
      paintMap(nodes[0 - split], firstBounds)

      secondBounds := right/lower half of bounds,
         dependant on split and orientation
      paintMap(nodes[split - #nodes], secondBounds)


xpath(nodes, query):

   return query results, starting with nodes
```




**TreeMap Visualization**

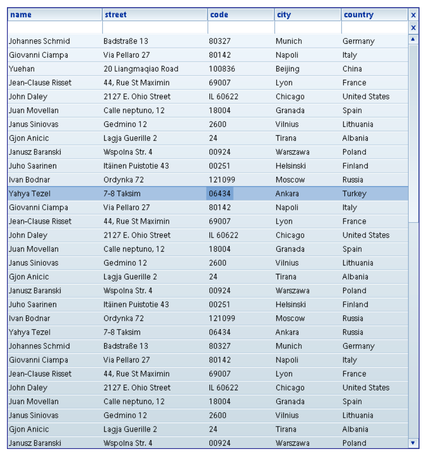



**Table Visualization**

```
paintTable(root node, table root tag):


paintTable(root, tag):

   nodes := xpath(root, "descendant::" + tag)
   columns := getTags(xpath(nodes, "child::*"))

   foreach node of nodes:
      foreach col of columns:
         text := xpath(node, "child::" + col)
         paint text in current row and column


getTags(nodes):

   return distinct tags from nodes


xpath(nodes, query):

   return query results, starting with nodes
```

## 3. Querying

### Support of Query Languages:
- XPath 1.0
- XPath/XQuery Full-Text (partial)
- XQuery Update (partial)

### Query Optimizations:
- Numerous query rewritings
- Text/Attribute value indexes
- Full-Text index

### Visual Interaction:
- XPath reformulation of simple keyword searches
- Querying by each key click
- Result highlighting in visualizations

### Sample Query

Query:

```
//country[.//religions ftcontains 'Muslim']
```

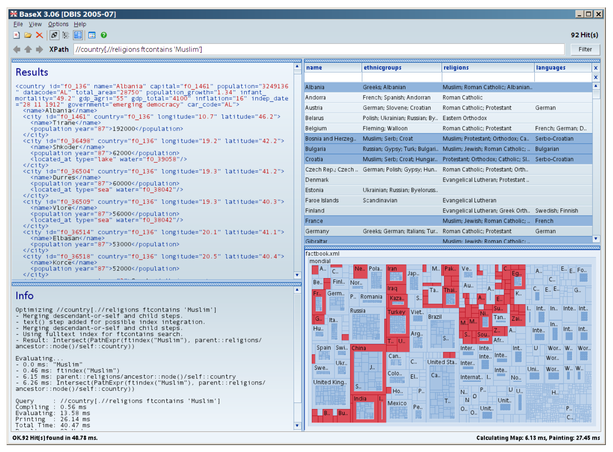
Optimizing:

```
- Merging descendant-or-self and child steps.
- text() step added for possible index integration.
- Merging descendant-or-selfand  child steps.
- Using fulltext index for ftcontains search.
```

Evaluating:

```
- Evaluate String         : 0.00 ms: "Muslim"
- Access Full-Text index  : 0.22 ms: ftindex("Muslim")
- Evaluate backward path  : 2.80 ms: parent::religions/ancestor::node()/self::country
- Create final result set : 2.86 ms: Intersect(PathExpr(ftindex("Muslim"),
                                       parent::religions/ancestor::node()/self::country))

Compiling  : 0.30 ms
Evaluating : 2.99 ms
Total Time : 3.49 ms
Results    : 92 Nodes
```

### Result Visualization




## 4. Mapping the File System

### Approach

Storing the file system in BaseX:
- missing query capability is added to file systems
- XPath support for directories, files & contents
- path traversal into files
- arbitrary metadata is embedded (EXIF, ID3, MPEG7, ...)
- textual data is embedded or indexed (ASCII files, XML documents, ...)

Integrating the operating system:
- file system operations are catched and processed
- file system updates and BaseX storage keep synchronized

### Command Line Queries

Conventional path traversal:

```
> cd /home/user
-> /dir[@name = 'home']/dir[@name = 'user']
```

Calculate size of all PNGs:

```
> du //*.png
-> sum(//file[@suffix = "png"]/@size)
```

Finding 'BTW' in all text files:

```
> grep 'BTW' //*.txt
-> //file[@suffix = "txt"][contains(content, "BTW")]
```

Show all 'bin' sub-directories:

```
> ls .//bin
-> fs_ls(.//dir[@name = "bin"])
```

### Visual Representation & Interaction