Handling Interlinked XML Instances on the Web

Erik Behrends, Oliver Fritzen, Wolfgang May
Institut für Informatik
Universität Göttingen
Germany
{behrends|fritzen|may}@informatik.uni-goettingen.de

EDBT, München, 29.3.2006

Situation

- focus on database aspect of XML
- autonomous XML sources on the Web
- provide data + external schema
- links between sources

Perspectives

- own sources reference other sources
  - what is the external schema?
  - queries against the own document must be “forwarded”
- other sources use my data
  - data restructuring, distribute over several instances
  - keep with the same external schema
  - transparent for the users

W3C XLink & XPointer

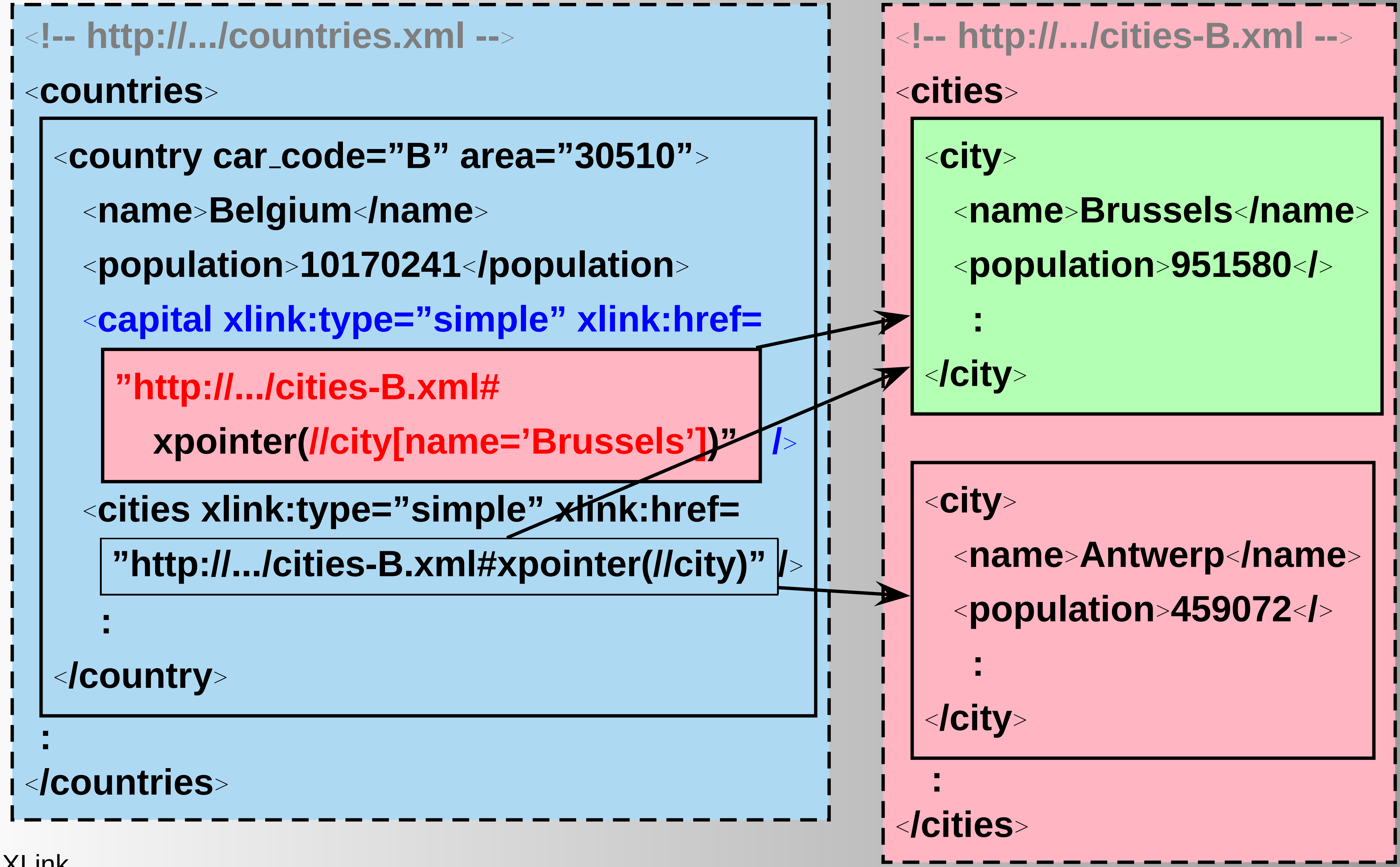
- XLink: language for defining links between XML documents
- To where?

  XPointer: *url* #xpointer(*xpath-expr*)

  <*linkelement* xlink:type="simple"

  xlink:href="*url* #xpointer(*xpath-expr*)"/>
- Data model?
- How to process?
- XLink and browsing: predefined xlink-attributes
- XLink and databases/queries ??

Simple Links – Example



```
<!-- http://.../countries.xml -->
<countries>
  <country car_code="B" area="30510">
    <name>Belgium</name>
    <population>10170241</population>
    <capital xlink:type="simple" xlink:href=
      "http://.../cities-B.xml#
        xpointer(//city[name='Brussels'])"  />
    <cities xlink:type="simple" xlink:href=
      "http://.../cities-B.xml#xpointer(//city)" />
    :
  </country>
  :
</countries>
```

```
<!-- http://.../cities-B.xml -->
<cities>
  <city>
    <name>Brussels</name>
    <population>951580</>
      :
  </city>
  <city>
    <name>Antwerp</name>
    <population>459072</>
      :
  </city>
  :
</cities>
```

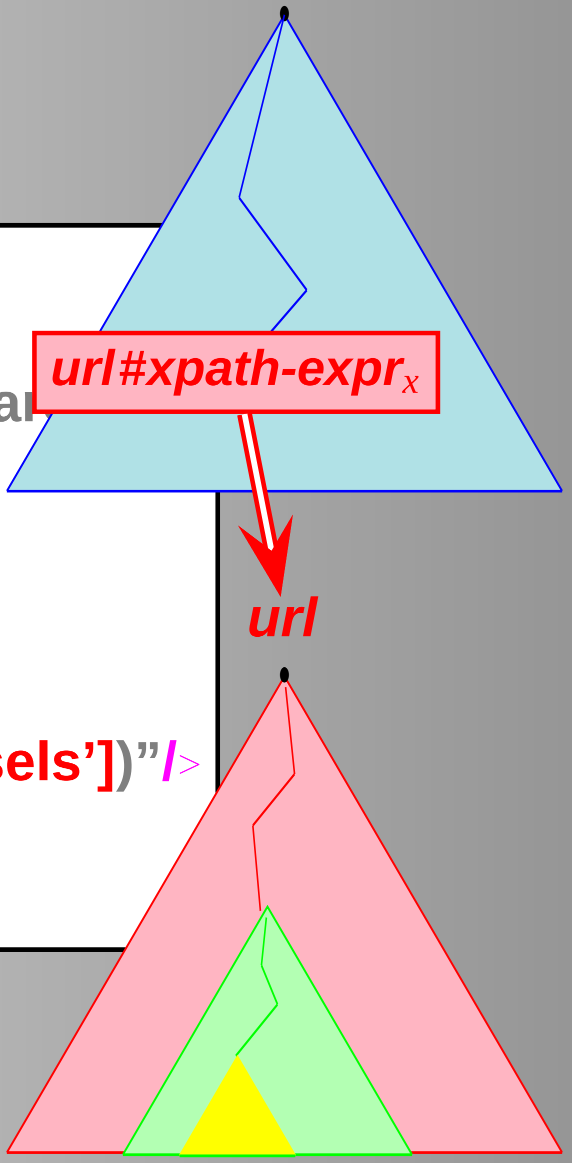
Simple Links

- similar to the HTML <A href="..."> construct.

Capitals of countries:

```
<!ELEMENT country (. . . capital . . . )>
<!ELEMENT capital EMPTY>
<!ATTLIST capital xlink:type (simple|extended|locator|ar
                                   #FIXED "simple"
                      xlink:href  CDATA #REQUIRED >

<country code="B">
  <capital xlink:href=
    "http://.../cities-B.xml#xpointer(//city[name='Brussels'])"/>
  :
</country>
```



query:

```
//country[@code="B"]/capital/??/population
```

Querying along Links

W3C *XML Query (XQuery) Requirements* (2001):

“the XML Query Data Model MUST include support for references, including both references within an XML document and references from one XML document to another”.

XPointer and XLink:

- specify how to *express* inter-document links in XML
- XLink specification tailored to browsing, not to querying

There is not yet an official proposal

- how to add link semantics to the actual data model (e.g., the XML Query Data Model)
- how to express/process queries through links
  Note: no way in XQuery, even not with user-defined functions!
- for evaluation strategies.

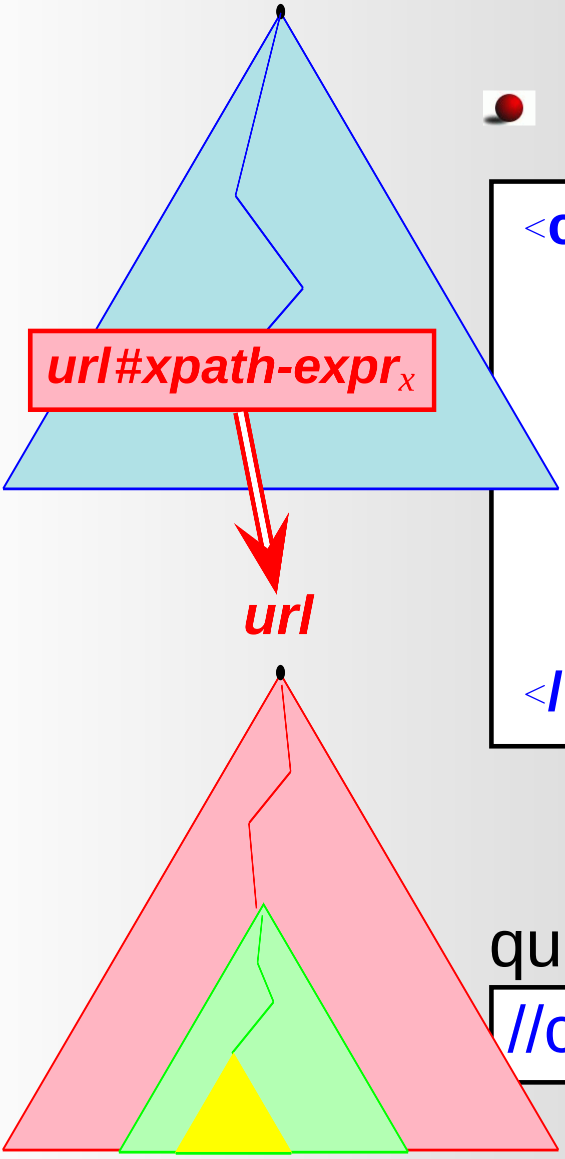
Data Models for Linked Documents

- extend the abstract data model with a linking construct
  - additional explicit navigation operator required (e.g. like dereferencing IDREF attributes with the id() function)

Abstract data model makes the links transparent

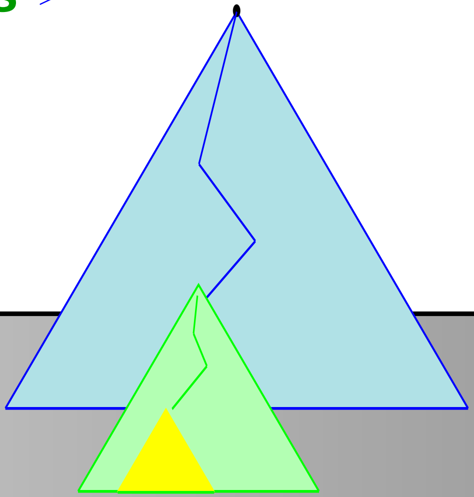
- each link can be seen as a view definition
  - integrates external schemata
  - embedded views implicitly become subtrees
  - queried by standard XPath expressions

Transparent Links



- regard link elements to be *transparent*

```
<country code="B">
   <capital xlink:type="simple"
            href="file:cities-B.xml#//city[name='Brussels']"
            attributes of Brussels >
      contents of Brussels
   </capital>
   :
</country>
```



query:

```
//country[@code="B"]/capital/population
```

Modeling

Applications:

- Data integration: building (virtual) XML documents by combining autonomous sources.
  - Sometimes: given target DTD/XML Schema
- Splitting an original XML document into a distributed database: Keep the external schema unchanged:
  - *virtual* model of the linked documents should be valid wrt. the *original* DTD,
  - all queries against the root document still yield the same answers as before.

$\Rightarrow$ cutting not only at (sub)elements, but also at attributes.

Modeling Switches = Integration Mapping

(a) Mapping of the target

- evaluate XPointer (sequence of nodes)
- take nodes ...

```
<city> <name>Brussels</> <population>951580</> </>
```

- .... or only their contents

```
<name>Brussels</> <population>951580</>
```

(b) Mapping of the XLink element and adding the result

- insert (a) into the link element

~~<capital> <city> <name>Brussels</> <population>951580</> </> </>~~

```
<capital> <name>Brussels</> <population>951580</> </>
```

Modeling Switches = Integration Mapping

(a) Mapping of the target

- evaluate XPointer (sequence of nodes)
- take nodes ...

```
<city> <name>Brussels</> <population>951580</> </>
```

- .... or only their contents

```
<name>Brussels</> <population>951580</>
```

(b) Mapping of the XLink element and adding the result

- add (a) instead of the link element

```
<city> <name>Brussels</> <population>951580</> </>
```

(... not suitable for capital, but for "cities")

~~<name>Brussels</> <population>951580</>~~

Modeling Switches = Integration Mapping

(a) Mapping of the target

- evaluate XPointer (sequence of nodes)
- take nodes ...

```
<city> <name>Brussels</> <population>951580</> </>
```

- .... or only their contents

```
<name>Brussels</> <population>951580</>
```

(b) Mapping of the XLink element and adding the result

- transform the link element name into a reference attribute that references (a):

```
<country capital= "•"> . . . </>
```

```
<city> <name>Brussels</> <population>951580</> </>
```

Modeling Switches = Integration Mapping

(a) Mapping of the target

(b) Mapping of the XLink element and adding the result

$$\left\{\begin{array}{c}\text{keep-element}\\ \text{drop-element}\\ \text{make-attribute}\end{array}\right\} \times \left\{\begin{array}{c}\text{insert-nodes}\\ \text{insert-bodies}\end{array}\right\}$$

<*linkelement* xlink:href="*xpointer*"

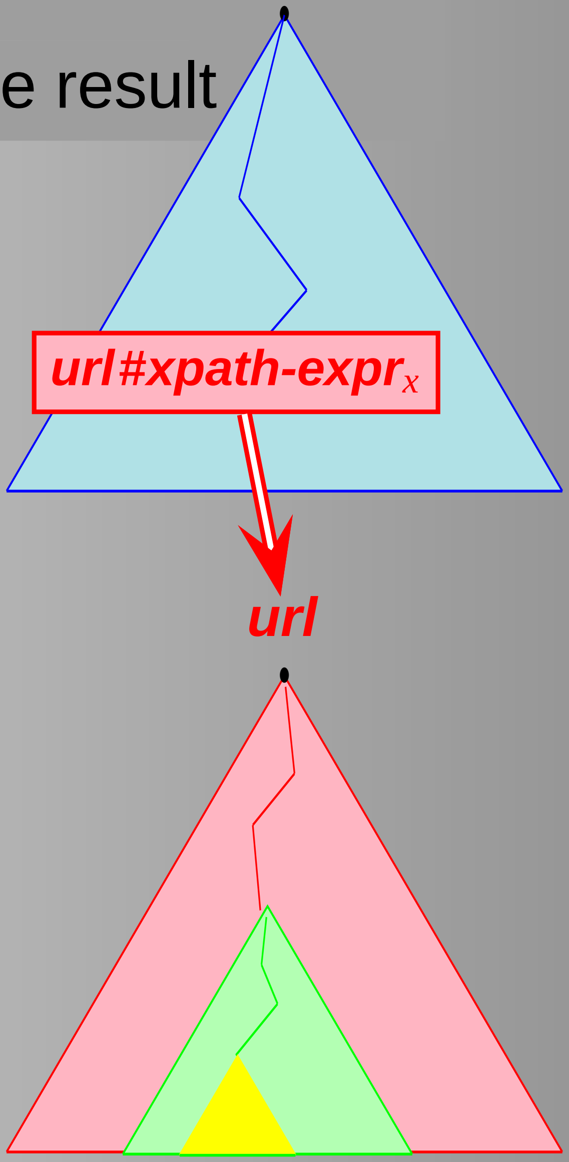
dbxlink:transparent=

"*left-hand-directive* *right-hand-directive*"

attributes>

content

</*linkelement*>



Formal Definition of the Virtual Model

Expand all XLink elements (recursively)

$\phi :$ NODE $\longrightarrow$ NODELIST

$\phi :$ for non-XLink elements:

element$(\textit{name}, \textit{attrs} \circ \textit{subelems}) \mapsto$

element$(\textit{name}, \textit{attrs} \circ \phi^*(\textit{subelems}))$

result can contain attribute nodes

$\phi^* :$ NODELIST $\longrightarrow$ NODELIST

$\phi^* :$ $[e_1, \ldots, e_k] \mapsto \phi(e_1) \circ \ldots \circ \phi(e_k)$

$\phi :$ for XLink elements:

$\phi(\textit{xlink-element}) = \phi^*(\gamma(\textit{xlink-element}))$

Formal Definition of the Virtual Model

$\phi:$ for XLink elements:

$$\phi(\textit{xlink-element}) = \phi^*(\gamma(\textit{xlink-element}))$$

$\gamma:$ XLinkElement $\longrightarrow$ NODELIST

$$\gamma(\textit{xlink-element}) =$$

$$= \gamma_L(\textit{left-hand-directive}, \gamma_R(\text{eval}(\textit{xpointer}), \textit{right-hand-directive}))$$

- virtual model can be infinite
- actual evaluation does not materialize it, but uses iterators & stepwise XPath strategy

Evaluation of XPath Expressions

Where? – dbxlink:eval

data shipping:

transfer contents of *url* and evaluate γ^{-1}(*xpath-expr*$_x$/*xpath-expr*$_2$) at the client.

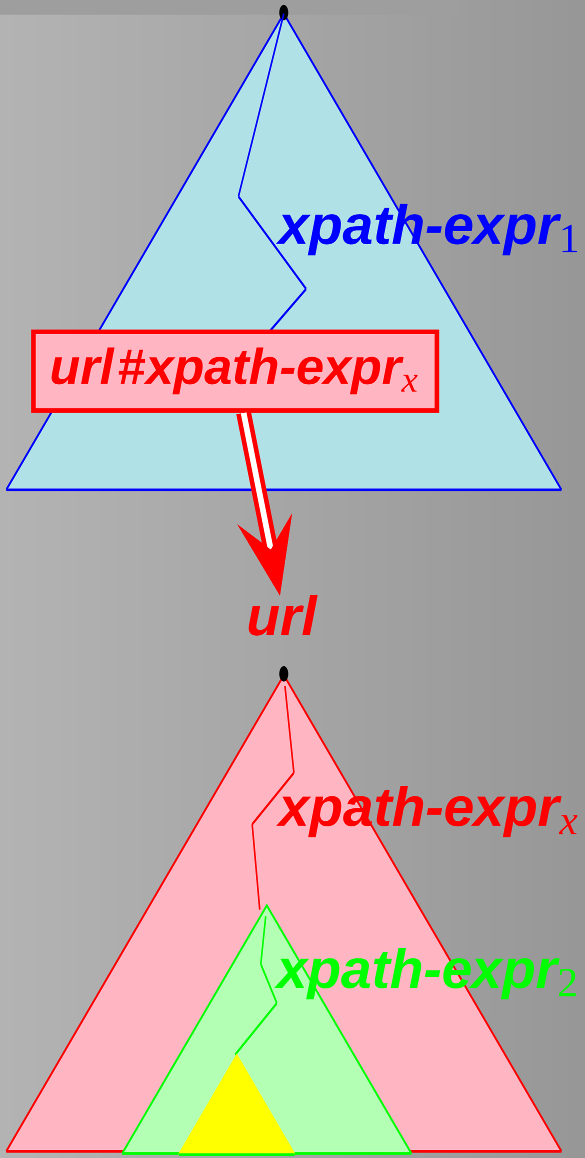
hybrid shipping:

evaluate *xpath-expr*$_x$ at the remote server, transfer the result and apply γ. Then, evaluate *xpath-expr*$_2$ at the client.

query shipping:

evaluate γ^{-1}(*xpath-expr*$_x$/*xpath-expr*$_2$) at the remote server, transfer the result.

Note: non-downward axes in *xpath-expr*$_2$ must be evaluated wrt. virtual model $\rightarrow$ restricted query shipping.



Answer Caching

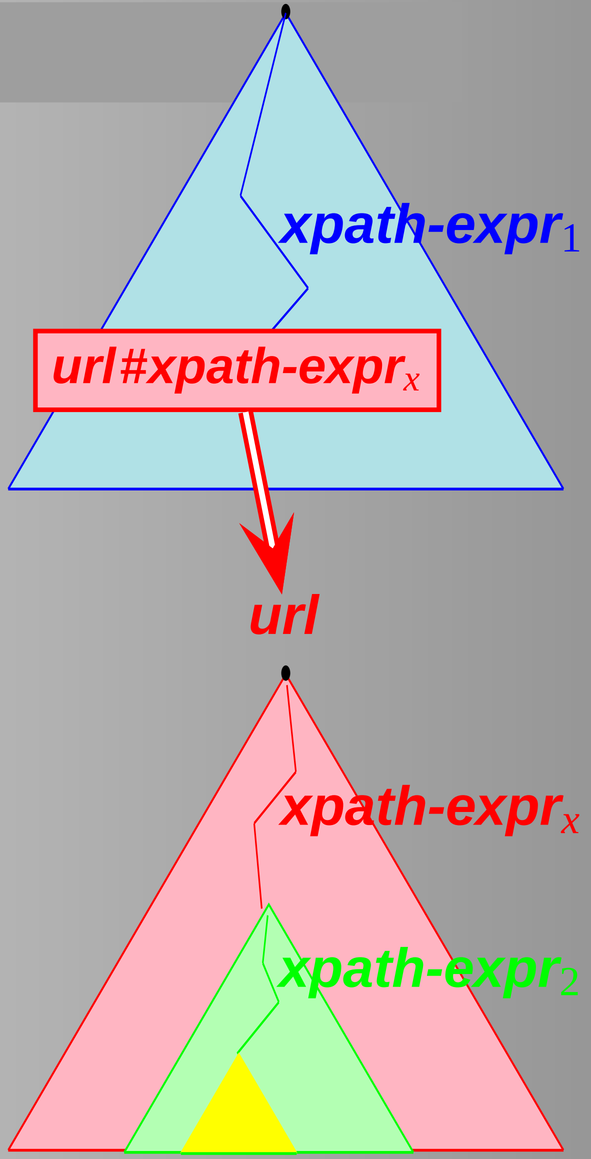
Caching – dbxlink:cache

complete: parses the *whole* target *document* and stores it (centrally) in the local XML database.

pointer: replaces the link by the result set of document(*url*)/$xpath\text{-}expr_x$

answer: stores the result of document(*url*)/$xpath\text{-}expr_x$/$xpath\text{-}expr_2$,

none: default



Implementation

Extension to the eXist [http://www.exist-db.org] XML database system.

- Adaptation of the XPath evaluation:
  materialization of computation of the new context in each step (resolve XPointer, apply γ),
- cyclic references between XPointers:
  - only dangerous for descendant axis $\rightarrow$ cycle detection (shipping history, bookkeeping),
  - infinite evaluation can be avoided when using “make-attribute”,
- caching.

Conclusion

- XLink elements:
  - seamless integration of *view definitions* into the database,
  - transparent semantics,
- new aspects for XLink coming from query languages.
- Optimizations and Perspectives:
  - Resource descriptions (Path index, XML schema): check a priori *whether* the actual query *xpath-expr*$_2$ on the result of *xpath-expr*$_x$ can be successful.
  - Hybrid shipping: use the “Projected Document” [Marian, Simeon VLDB03] of the result of *xpath-expr*$_x$ wrt. *xpath-expr*$_2$ (do the projection when the result comes in, or even request only the projected document).
  - Apply results from ActiveXML (e.g. algebraic optimization).

Thank You

Questions ??