Querying along XLinks in XPath/XQuery: Situation, Applications, Perspectives

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

QLQP- Query Languages and Query Processing
München, 31.3.2006

Situation

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

Different Scenarios

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

W3C XLink & XPointer

- XLink: language for defining links between XML documents
  - arbitrary elements can serve as links
  - arbitrary many sources can be connected (extended links)
  - arbitrary parts (XML fragments) can be referenced by XPointers: *url* #xpointer(*xpath-expr*)
- A simple XLink (we do not consider extended links here):

```
<linkelement   xlink:type="simple"
               xlink:href="url#xpointer(xpath-expr)">
    contents
</linkelement>
```

- XLink and browsing: predefined xlink-attributes ✓
- Data model for documents connected by XLinks?
- How to process XLinks during querying?

Simple Links – Example (MONDIAL Database)

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

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

  <city>
    <name>Antwerp</name>
    <population>459072</>
      :
  </city>
  :
</cities>
```

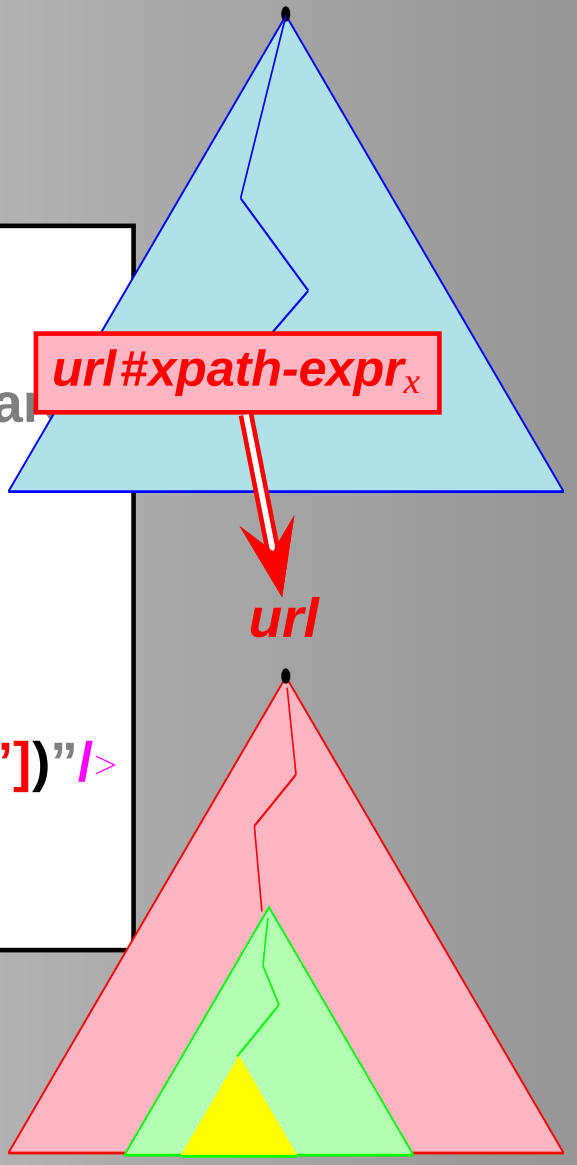
Simple Links

- 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=
        "file: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* versatile inter-document links in XML
- XLink specification is tailored to browsing, not to querying

There is not yet an official W3C 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.

Querying along Links

- Restricted “shorthand” pointers based on ID attributes (similar to anchors in HTML: href=”http://www.foo.com#id”)

```
(: adapted from [Lehner, Schoening: XQuery] :)
(: XPointer of the form  http://.../country.xml#xpointer(id('D')):)

declare namespace fu = "http://www.example.org/functions";
declare function fu:follow-xlink($href as xs:string) as item()*
{  let $docValue := fn:substring-before($href,"#")
   let $x := fn:substring-after($href,"#xpointer( id('")
   let $idValue := fn:substring-before($x,"')")"
   return fn:doc($docValue)/fn:id($idValue)
   (: evaluates fn:doc(http://.../country.xml)/fn:id('D') :)
};
```

- reduces XPointer evaluation to ID evaluation
- does not cover general case

Querying along Links

- General case (XPointer contains any XPath expression):

```
(: XPointer of the form http://.../country.xml#xpointer(//country[@code='D']) :)
declare namespace fu = "http://www.example.org/functions";
declare function fu:follow-xlink($href as xs:string) as item()*
{  let $docValue := fn:substring-before($href,"#")
   let $x := fn:substring-after($href,"#xpointer(")
   let $path := fn:substring-before($x,")")
   return fn:doc($docValue)/ $path
   (: should evaluate fn:doc(http://.../country.xml)//country[@code='D'] :)
};
```

- Syntax not allowed

Querying along Links

- With saxon extension function saxon:evaluate():

```
(: XPointer of the form http://.../country.xml#xpointer(//country[@code='D']) :)
declare namespace fu = "http://www.example.org/functions";
declare function fu:follow-xlink($href as xs:string) as item()*
{  let $docValue := fn:substring-before($href,"#")
   let $x := fn:substring-after($href,"#xpointer(")
   let $path := fn:substring-before($x,")")
   return fn:doc($docValue)/saxon:evaluate($path)
   (: evaluates fn:doc(http://.../country.xml)/saxon:evaluate('//country[...]') :)
};
```

- Query:

```
doc("http://.../countries.xml")//country[@car_code="B"]/
        capital/ fu:follow-xlink(@xlink:href) /population .
```

Querying Linked Documents

One possible solution:

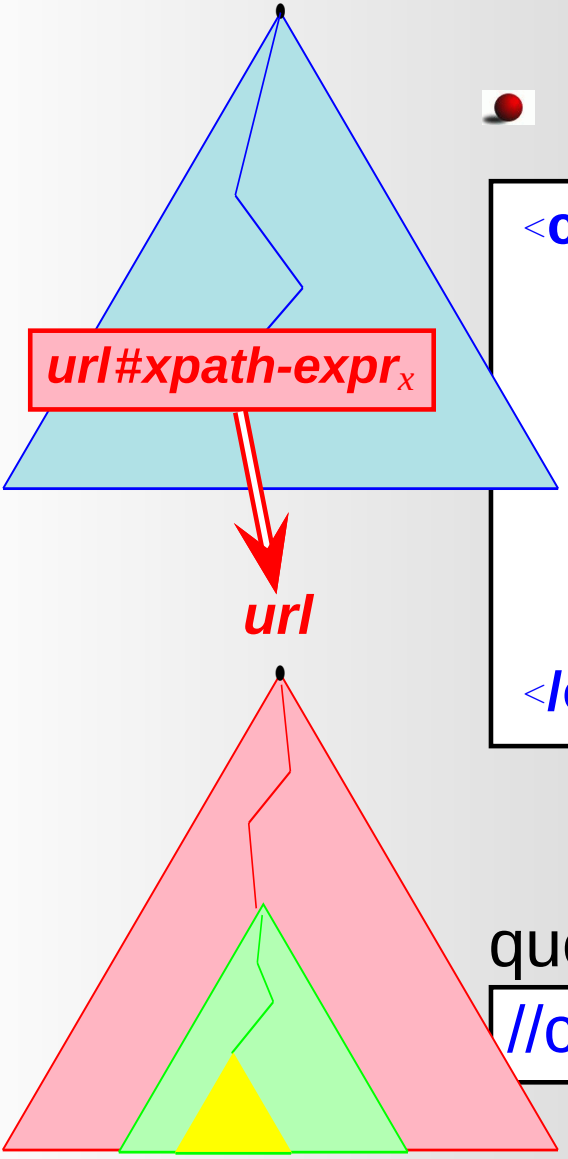
- extend the abstract XML data model with a linking construct
  - additional explicit navigation operator required
    (like dereferencing IDREF attributes with the id() function)

```
doc("http://.../countries.xml")//country[@car_code="B"]/
        capital/ fu:follow-xlink(@xlink:href) /population .
```

⇒ Here, the user has to be aware of the XLinks:

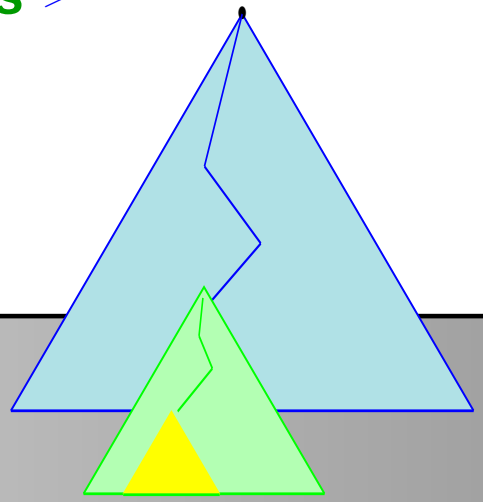
- queries have to respect XLinks
- XLinks are not part of the application data

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
```

Transparent Links

Our abstract data model makes the links transparent:

- each link can be seen as a view definition
  - integrates remote schemata
  - embedded views implicitly become subtrees, while XLinks are resolved silently
  - can be queried by standard XPath expressions:

```
doc("http://.../countries.xml")//country[@car_code="B"]/
        capital/population .
```

⇒ We introduced a namespace dbxlink with additional directives:

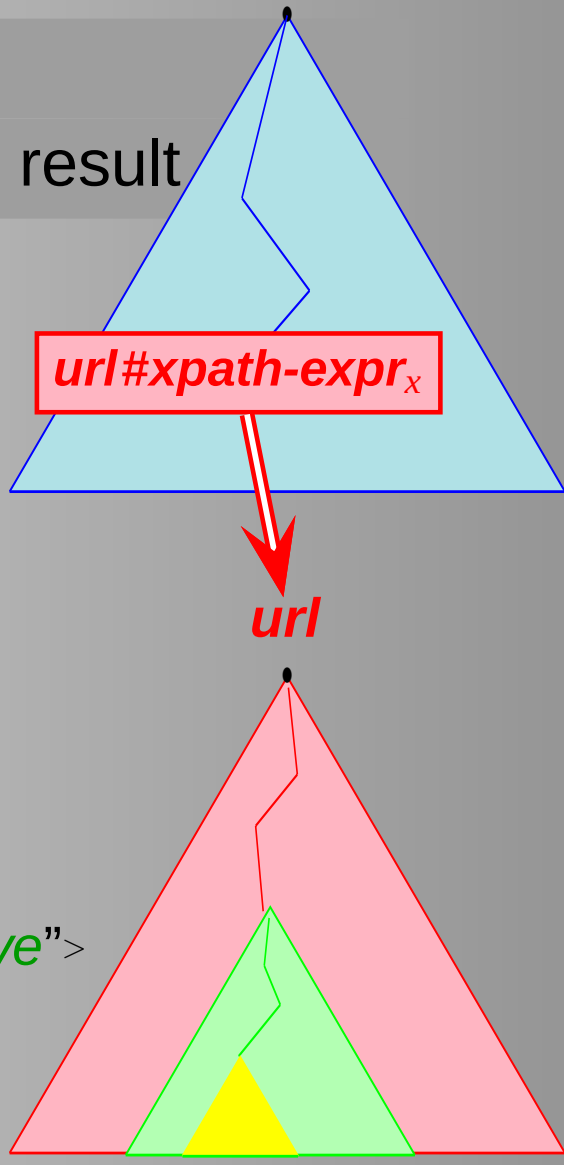
- specification of modeling, evaluation and caching modes
- no changes needed in XPath/XQuery and XLink

Modeling Switches = Integration Mapping

(a) Mapping of the target

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

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



<linkelement xlink:href="*xpointer*" *attributes*
dbxlink:transparent="*left-hand-directive right-hand-directive*">
content
</linkelement>

Modeling – Example (MONDIAL Database)

```
<!-- http://foo.com/countries.xml -->
<countries>
  <country car_code="B" area="30510">
    <name>Belgium</name>
    <capital xlink:type="simple" xlink:href=
      "http://bar.org/cities-B.xml#
        xpointer(//city[name='Brussels'])"
      dbxlink:transparent="keep-body insert-bodies"
    />
    <cities xlink:type="simple" xlink:href=
      "http://bar.org/cities-B.xml#xpointer(//city)"
      dbxlink:transparent="drop-element insert-nodes"
    />
  </country>
</countries>
```

```
<!-- bar.org/cities-B.xml -->
<cities>
  <city>
    <name>Brussels</>
    <population>951580</>
    :
  </city>

  <city>
    <name>Antwerp</>
    <population>459072</>
    :
  </city>
  :
</cities>
```

Modeling – Example (MONDIAL Database)

```
<country car_code="B">
  <name>Belgium</name>
  <capital>
    <name>Brussels</name>
    <population>951580</>
  </capital>
  <city>
    <name>Brussels</name>
    <population>951580</>
  </city>
  <city>
    <name>Antwerp</name>
    <population>459072</>
  </city>
</country>
```

- query virtual integrated view:

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

- view created on-the-fly for queries

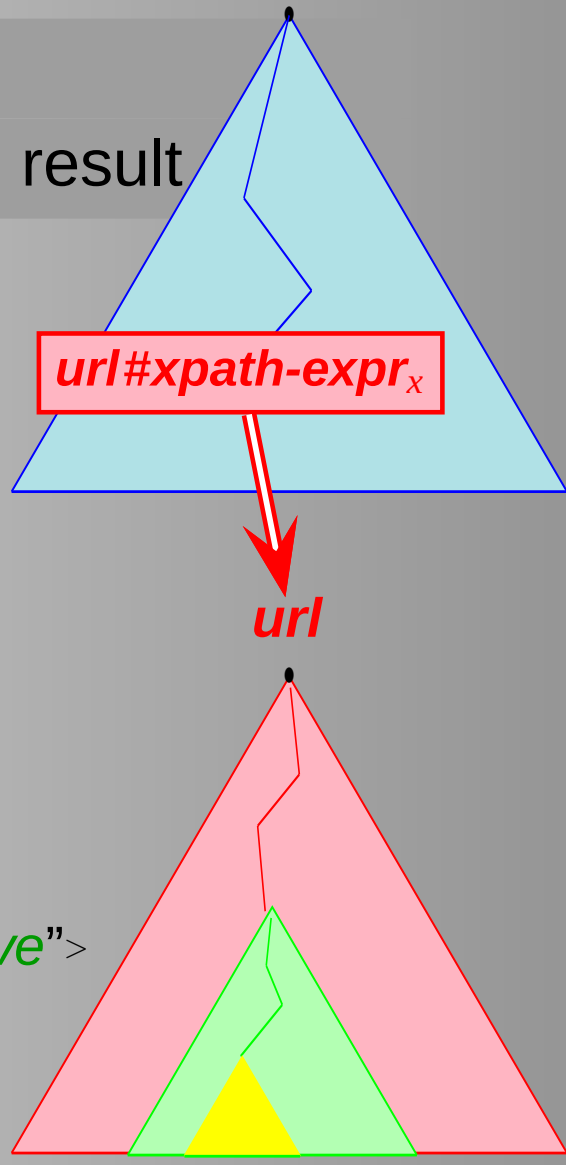
- reduction to relevant links

Modeling Switches = Integration Mapping

(a) Mapping of the target

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

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



<linkelement xlink:href="*xpointer*" *attributes*
dbxlink:transparent="*left-hand-directive right-hand-directive*">
content
</linkelement>

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.

Related Work

ActiveXML [Abiteboul et al, VLDB 02 etc.]

- General approach for invoking arbitrary Web Services that return XML (as embedded views),
- <axml:call> elements,
- no special mapping/transformation functionality.

dbxlink via Active XML

Provide a service that answers XPointers and does the transformation mapping:

```
<axml:call href="http://.../dbxlink-servlet">
  <capital xlink:type="simple"
    xlink:href="http://.../cities-B.xml#xpointer(//city[name='Brussels'])"
    dbxlink:transparent="keep-body insert-nodes"/>
</axml:call>
```

- make-attribute not possible.

Related Work

ActiveXML [Abiteboul et al, VLDB 02 etc.]

- General approach for invoking arbitrary Web Services that return XML (as embedded views),
- <axml:call> elements,
- no special mapping/transformation functionality.

Active XML via dbxlink

```
<call xlink:type="simple"
  xlink:href="service-url"
  dbxlink:transparent="drop-element insert-nodes"/>
```

- no parameters for call allowed (except in the url).

⇒ dbxlink not only suitable for XPointer-XML views, but also calling for Web Services.

Summary

Basic Functionality of XLink

XML Requirements “fulfilled”:

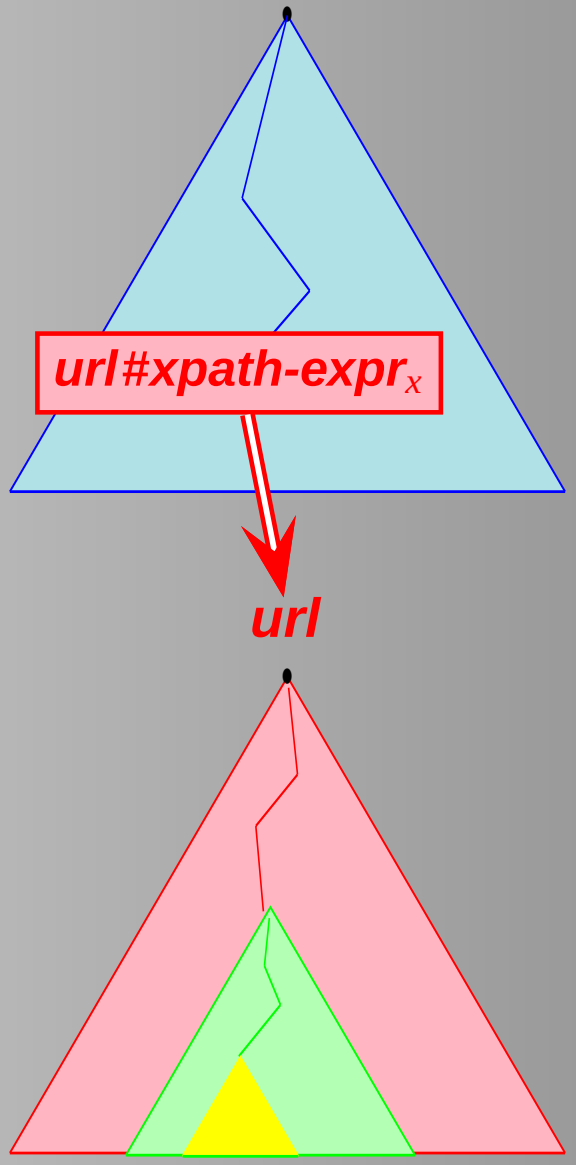
- Data Model: seamless integration of *view definitions* into the database,
- transparent semantics,
- querying in XPath (adaptation of internal evaluation).

Implementation

- extension to the eXist [http://exist-db.org] XML database system.
- handling of cyclic instances.
- different evaluation modes (query, data and hybrid shipping).
- caching according to evaluation strategies.

Optimizations and Perspectives

- Resource descriptions (Path indexes, XML Schema): check a priori *whether* the view can contribute to the answer of the remaining query.
- Evaluate the view already as a “projected document” [Marian, Simeon VLDB03] wrt. the remaining query.
- Parallel evaluation of views during querying.
- Caching: utilize query containment for similar XPath expressions used in XPointers.
- Arcs and link bases.



Thank You !

Questions ??