Datenmanipulation und -integration in XML

Wolfgang May
Universität Freiburg

“Computerlinguistik und Semistrukturierte Daten”
21.2.2002
Herrsching/Ammersee

XML

- Internet-weites Datenformat
  Dokumente und “Datenbank-style”
- verteilte, autonome Quellen
- electronic data interchange
- Standards und Standardsoftware

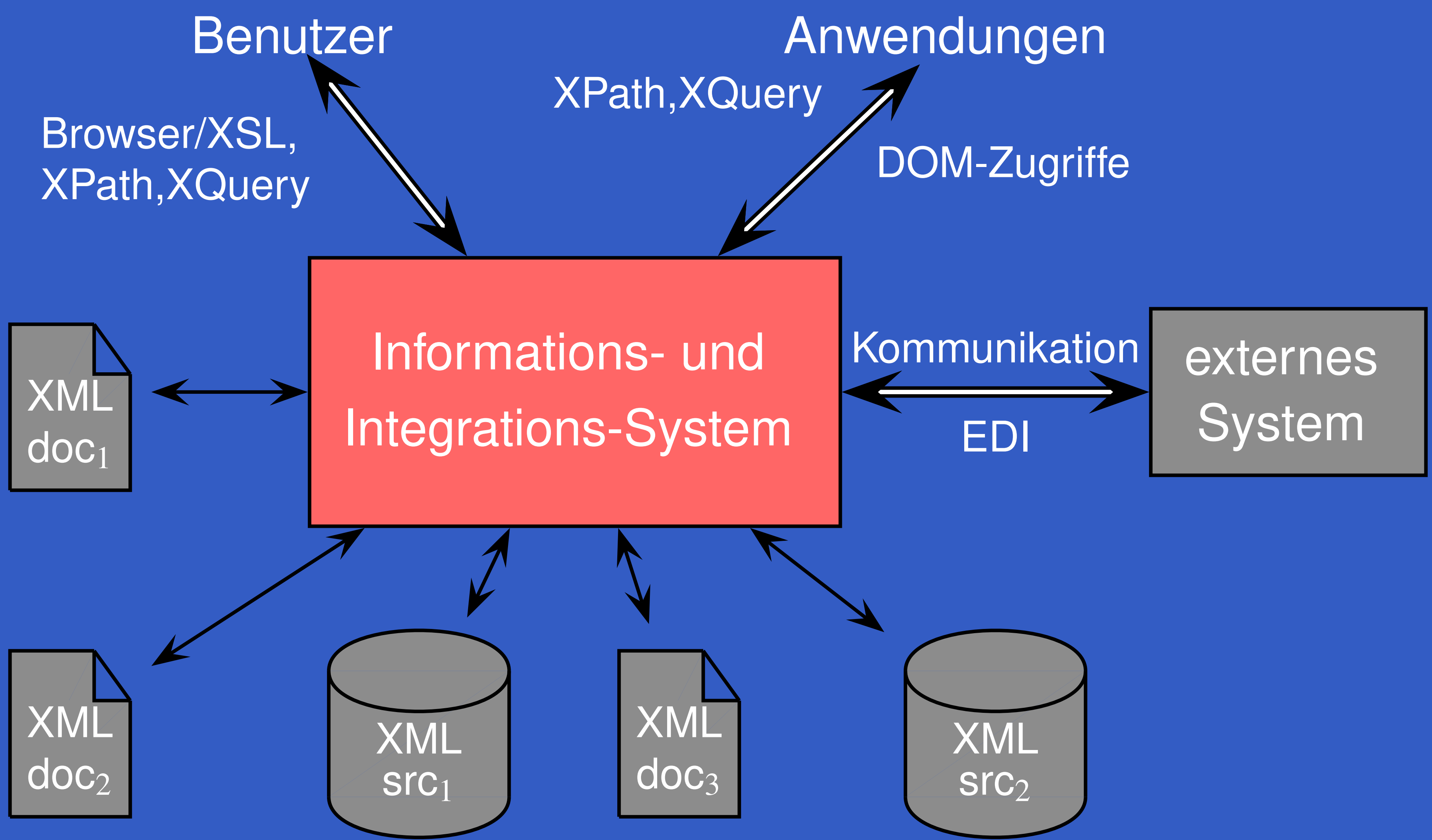
Vorgängerkonzepte

- Dokumente: SGML
  - "Anfragen" durch Navigation
  - Transformationen
- Semistructured Data:
  TSIMMIS/OEM, Strudel, F-Logic/FLORID, KIF
  - semistrukturierte, selbstbeschreibende Daten(bank)
  - Navigation
  - Anfragen: SQL-artige Sprachen

Beide Aspekte in Computerlinguistik & Semistrukturierte Daten:

- Dokumente
- Daten: Metadaten zu Dokumenten, Ontologien, Wissen über Sprache

Szenario



Benutzer
Anwendungen
Browser/XSL,
XPath,XQuery
XPath,XQuery
DOM-Zugriffe
Informations- und
Integrations-System
Kommunikation
EDI
externes
System
XML
doc1
XML
doc2
XML
src1
XML
doc3
XML
src2

State of the Art

- Zwei Anfragesprachen:
  W3C XPath + XQuery vs. XML-QL
- W3C XML Query Requirements/Data Model/Algebra
- erste noch inoffizielle Vorschläge für Sprachkonstrukte + Modelle zur Datenmanipulation

Anfragen

Adressierung: XPath

- Navigationsausdrücke
- Ergebnis: “Result set” aus XML-Knoten

Anfragen

Adressierung: XPath

- Navigationsausdrücke
- Ergebnis: “Result set” aus XML-Knoten

/mondial/country/name

/mondial/country/@car_code

/mondial/country/name/text()

/mondial/country[name = “Germany”]//city/name/text(),

//city[population $>$ 5000000]/name/text()

//city[population[@year $<$ 1990] $>$ 5000000]/name/text()

/mondial/organization[name=“EU”]/@seat$\Rightarrow$city/name/text()

Anfragen

Adressierung: XPath

- Navigationsausdrücke
- Ergebnis: “Result set” aus XML-Knoten

Anfragen: XQuery

FOR *variable* IN *xpath-expr*
LET *additional_variable* := *xpath-expr*
WHERE *filters*
RETURN *xml-expr*

- beeinflusst von
  - SQL (SFW-Klauseln, Variablenbindungen)
  - XPath (Adressierung)
  - XSLT und XML-QL (Erzeugung des Ergebnisses)

Updates in XML

FOR *variable* IN *xpath-expr*
LET *additional_variable* := *xpath-expr*
WHERE *filters*
apply update method to variable

- e.Delete($member$)
- e.Insert($content$)
- e.Rename($member$,$name$)
- e.Replace($member$, $content$)
- *content*: Variable oder XML Pattern
- falls *content* ein bereits existierendes XML-Element ist oder enthält?

Datenintegration

- Dokumente: geordnete Baumstruktur
  “integration follows structure”
- Datenbanken: Graph, keine Ordnung
  Semantische Integration:
  - Elemente in unterschiedlichen Quellen repräsentieren dasselbe Objekt

  $\Rightarrow$ Element/Objekt“fusion”
  - Synonyme, Ontologien
  - nicht kompatibel mit dem XML-Datenmodell

Design-Überlegungen

- Erfahrungen mit F-Logic zur Integration semistrukturierter Daten

Datenmodell: XTreeGraph

- erweitert das DOM/XML Query Data Model
- Datenbank nicht als Baum, sondern als Graph aus überlappenden Bäumen
- unterstützt Updates und Datenintegration
- Ergebnisse: Views über diesem Graphen

Design-Überlegungen

- Erfahrungen mit F-Logic zur Integration semistrukturierter Daten

Sprache: XPathLog

- Datalog-artige Erweiterung von XPath
- Deklarative regelbasierte Sprache
- Regeln mit Bottom-up-Semantik

Beispiel: Mondial

```
<mondial>
  <country car_code="B"
    capital="cty-Brussels"
    memberships="org-eu org-nato ..">
    <name>Belgium</name>
    <population>
      10170241
    </population>
    <city id="cty-Brussels"
      country="B">
      <name>Brussels</name>
      <population year="95">
        951580
      </population>
    </city>
    :
  </country>
```

```
  <organization id="org-eu"
      seat="cty-Brussels">
    <name>Europ. Union</name>
    <abbrev>EU</abbrev>
    <members type="member"
      country="GR F E A D I B L . . . "/>
    <members type="applicant"
      country="AL CZ . . . "/>
  </organization>
  <organization id="org-nato"
    seat="cty-Brussels" . . . >
    :
  </organization>
  :
  :
  :
</mondial>
```

XPathLog in Beispielen

- Reine XPath-Ausdrücke

  ?- //country[name/text() = “Belgium”]//city/name/text().
  true

- Ausgabe einer Ergebnismenge

  ?- //country[name/text() = “Belgium”]//city/name/text()$\rightarrow$N.
  N/“Brussels”
  N/“Antwerp”
  ⋮

XPathLog in Beispielen

- Reine XPath-Ausdrücke

  ?- //country[name/text() = “Belgium”]//city/name/text().

  true

- Zusätzliche Variablen

  ?- //country[name/text()→N1 and
  @car_code→C]//city/name/text()→N2.

  N1/“Belgium” C/“B” N2/“Brussels”

  N1/“Belgium” C/“B” N2/“Antwerp”

  ⋮

⋮

XPathLog in Beispielen

- Metadaten: Navigationsvariablen

?- //Type→X[name/text()→"Monaco"].

Type/country	X/*country-monaco*
Type/city	X/*city-monaco*

- Metadaten: Schemaanfragen

?- //city/N.

N/name

N/population

⋮

Regelköpfe

Konstruktive Semantik von definiten XPathLog-Atomen:

- Nur *child*, *sibling* und *attribute*-Achse
- Keine Negation, Funktionsanwendungen, Aggregation und *proximity position predicates*

"/" und "[. . .]" als Konstruktoren:

- *host*[*property*→*value*] modifiziert *host*
- *host*/*property remainder*
  erzeugt ein neues Element *host/property*, das *remainder* erfüllt
- *property* ist *axis::name* oder *axis(i)::name*

⋮

Regelköpfe: Attribute

C[@datacode→"be"], C[@memberships→O] :-
//country→C[@car_code="B"],
//organization→O[abbrev/text()→"EFTA"].

⇓

<country datacode="be" car_code="B"
memberships="org-eu org-un org-efta . . . ">
⋮
</country>

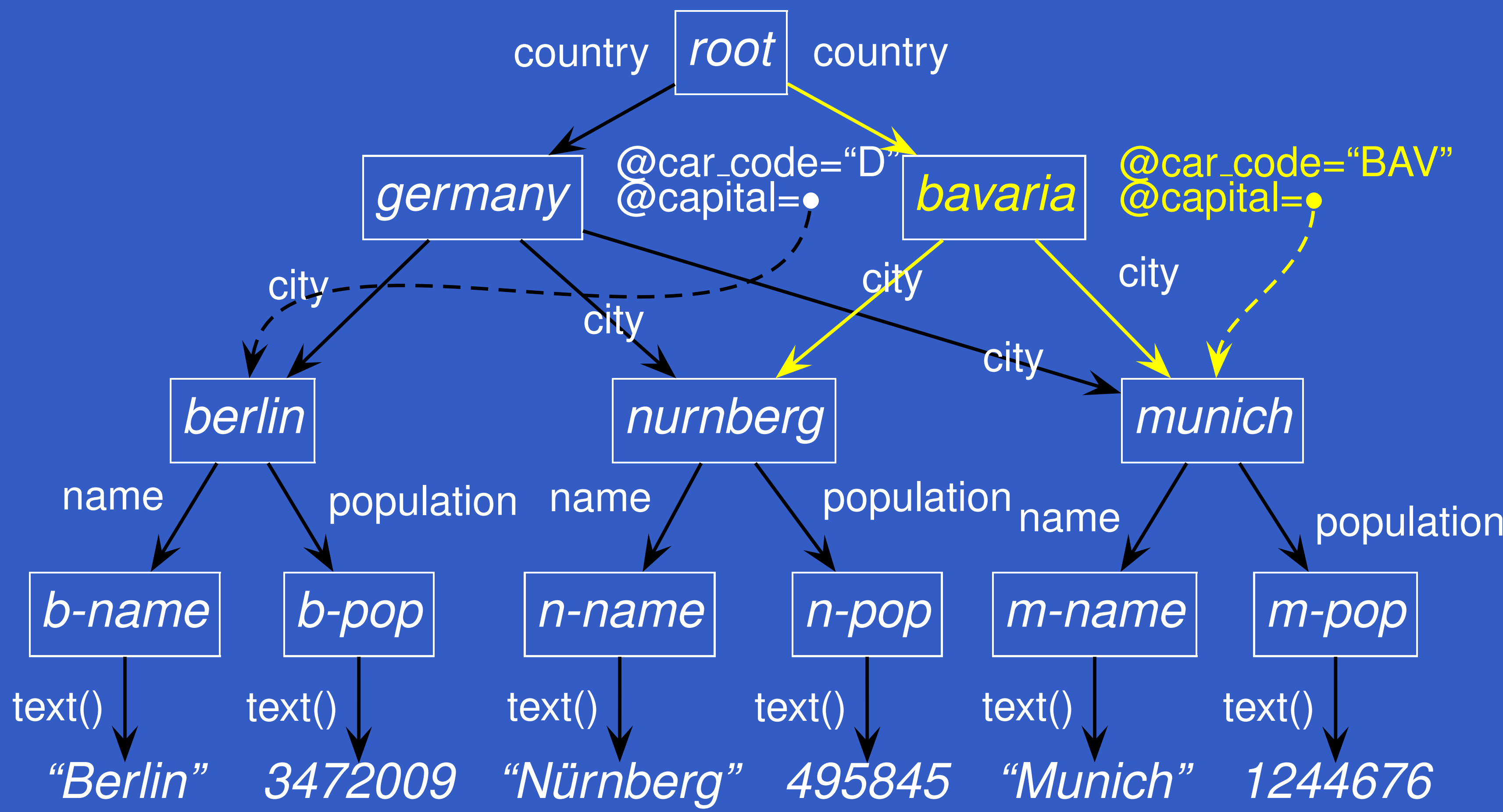
Erzeugung “freier” Elemente

/country[@car_code→“BAV”].

⇓

<country car_code=“BAV”> </country>

Linking



- Elemente besitzen mehrere Eltern

Hinzufügen von Subelement-Beziehungen

C[@capital→X and city→X and city→Y] :-
//country→C[@car_code→“BAV”],
//city→X[name/text()=“Munich”],
//city→Y[name/text()=“Nurnberg”].

- city-Elemente werden als Subelements gelinkt
- effiziente *in-place*-Restrukturierung und -Integration

Integration

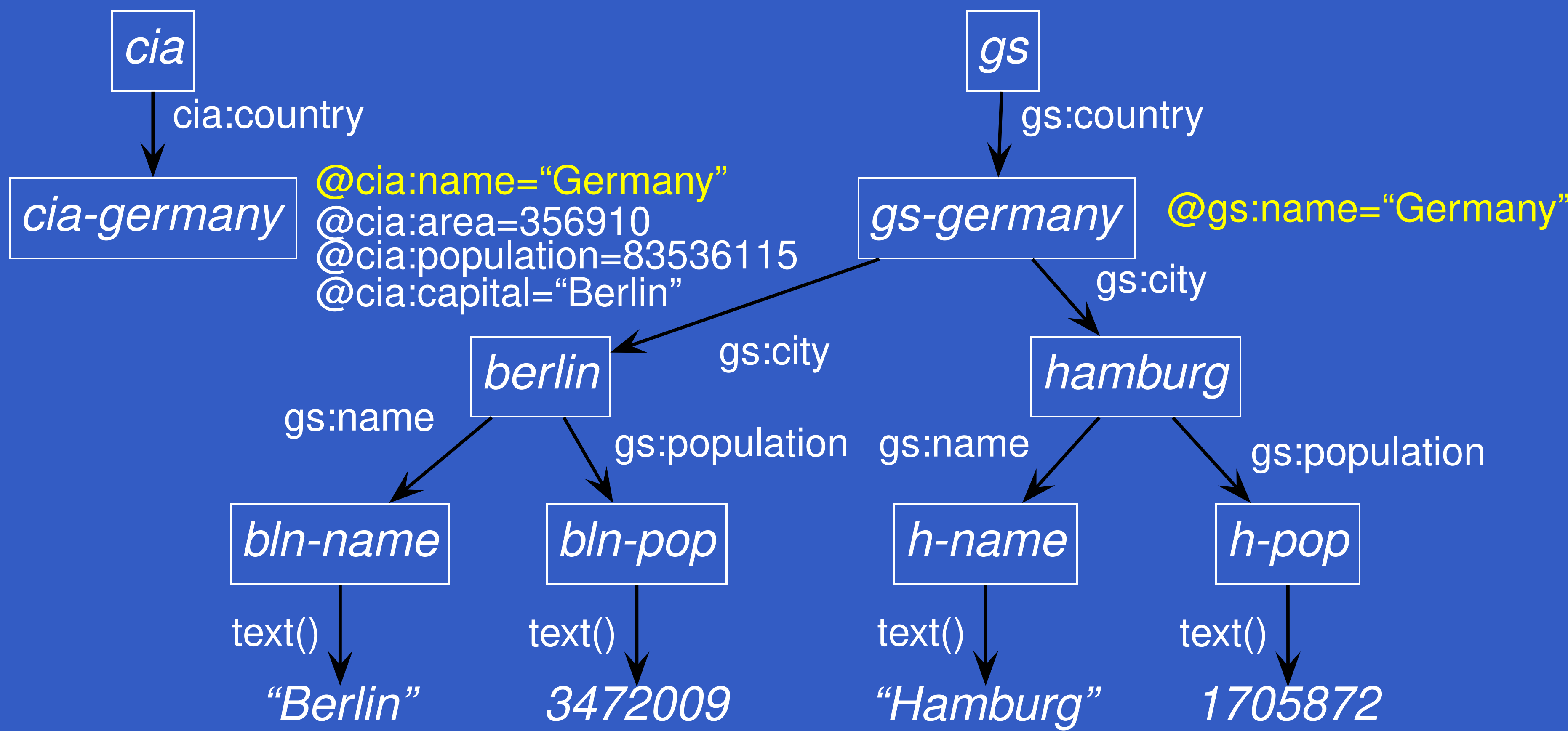
“Three-level”-Modell

Lesen mehrerer Datenquellen:

- “basic” layer: Quellen als Baumstrukturen
- wahlweise mit Namespaces

Datenquellen: Bäume mit Namespaces

- cia: Name, Fläche, Bevölkerung, Hauptstadt (Name)
- gs: Städte mit Name und Bevölkerung



Integration

"Three-level"-Modell (2)

Verbinden der Daten:

- "internal" layer: XTreeGraph
  - Überlappende Bäume
- Verschmelzen von Elementen
- Einfügen zusätzlicher Subelement-Beziehungen
- Erzeugung neuer Elemente
- Einführen von Synonymen für Eigenschaften

Synonyme

- Einige Eigenschaften der Quellen werden direkt in den Ergebnisbaum übernommen - (ggf. mit anderen Namen):

cia:name = name.	gs:name = name
cia:area = area.	
cia:population = population.	gs:population = population.
cia:text() = text().	gs:text() = text()

- erzeugt *keine* neuen Element- oder Attributknoten,
- sondern “nur” weitere Navigationspfade,
- ordnungsbewahrend

Elementfusion

- Elemente repräsentieren dasselbe Objekt in verschiedenen Quellen
- vereinigen zu einem einzigen Element: $e_1 = e_2$

Ergebniselement

1. ist ein Element *beider* Bäume,
2. vereinigt die Attribute beider Elemente,
3. vereinigt die Subelemente beider Elemente,

Elementfusion: Beispiel

$result$[country→C1],

C1 = C2 :- cia/cia:country[@name→N]→C1,

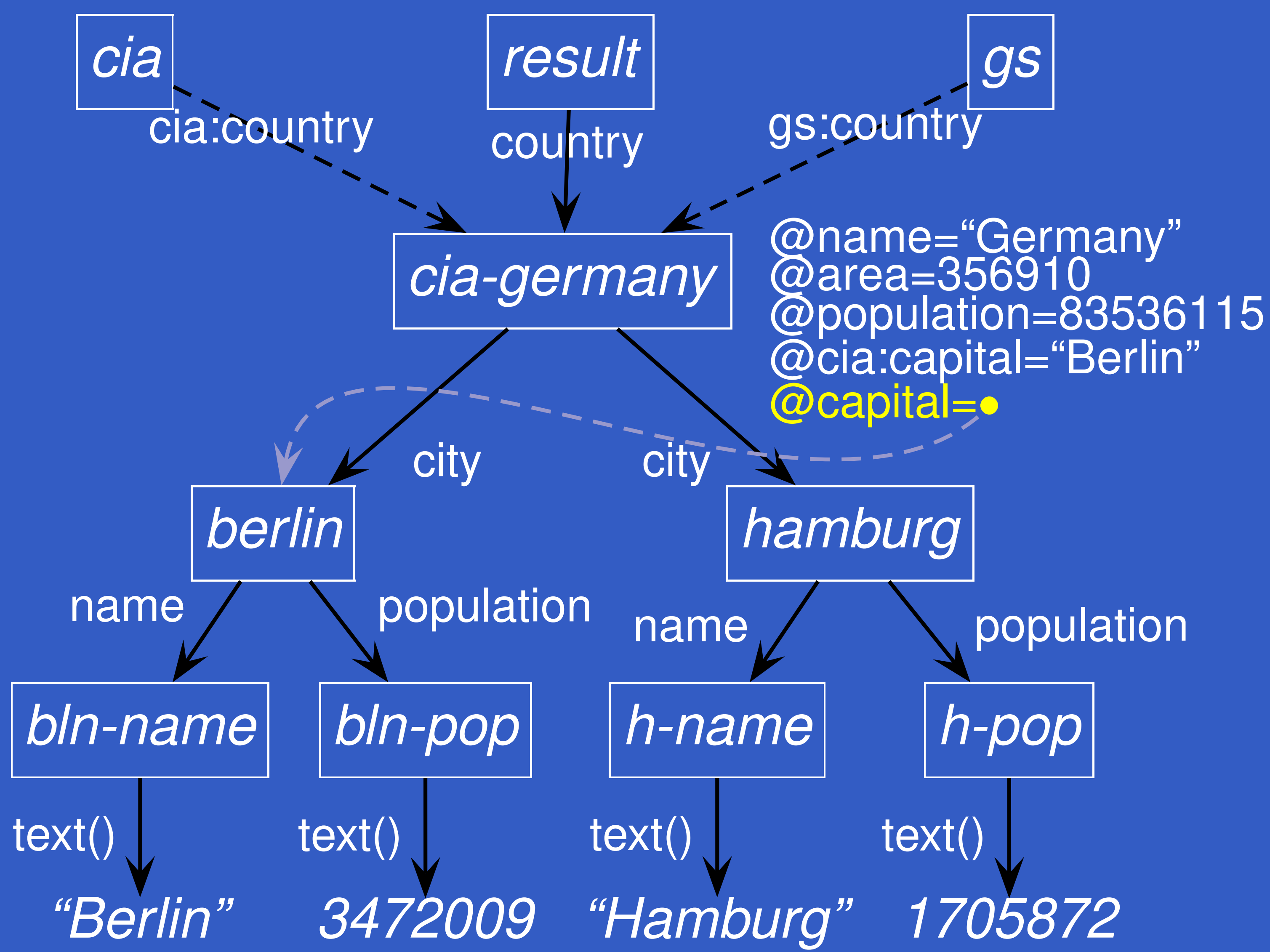
gs/gs:country[@name→N]→C2.

C[@capital→Cap] :-

$result$/country→C[@cia:capital→N and

city→Cap[name/text()→N]].

Elementfusion: Beispiel



C[@capital→City] :-
result/country→C[@cia:capital→Name and
city→City[name/text()=Name]].

Integration

"Three-level"-Modell (3)

- "export" layer: XML-Ergebnisbäume als Sichten/Projektionen
- Wurzel
- Signatur
  - Klassenhierarchie (+ nichtmonotone Vererbung)
  - Signaturen

    country[@car_code⇒string].

    country[@area⇒numeric].

    country[@capital⇒city].

    country[city⇒city].

Ergebnisse

- Implementierung “LoPiX”
- Fallstudie “Mondial”
- Ausdruckskräftige und flexible Sprache/Datenmodell
  - Klassenhierarchie + Vererbung
  - Anfragen an Metadaten/Schema
- Updates: deklarative Semantik zur Erzeugung von XML-Daten mit XPath
- Integrationsoperatoren: Elementverschmelzung und Synonyme
- ... OMAR-Projekt (Orientalische Handschriften, Kooperation mit dem Orientalistischen Seminar)

Questions ??