

Intelligente Systeme im WWW: Semantic Web

RDF Anfragesprachen

Dr. Raphael Volz
Institut AIFB, Universität Karlsruhe (TH)

*(Most) Material courtesy
Prof. Dr. Heiner Stuckenschmidt
Uni Mannheim*

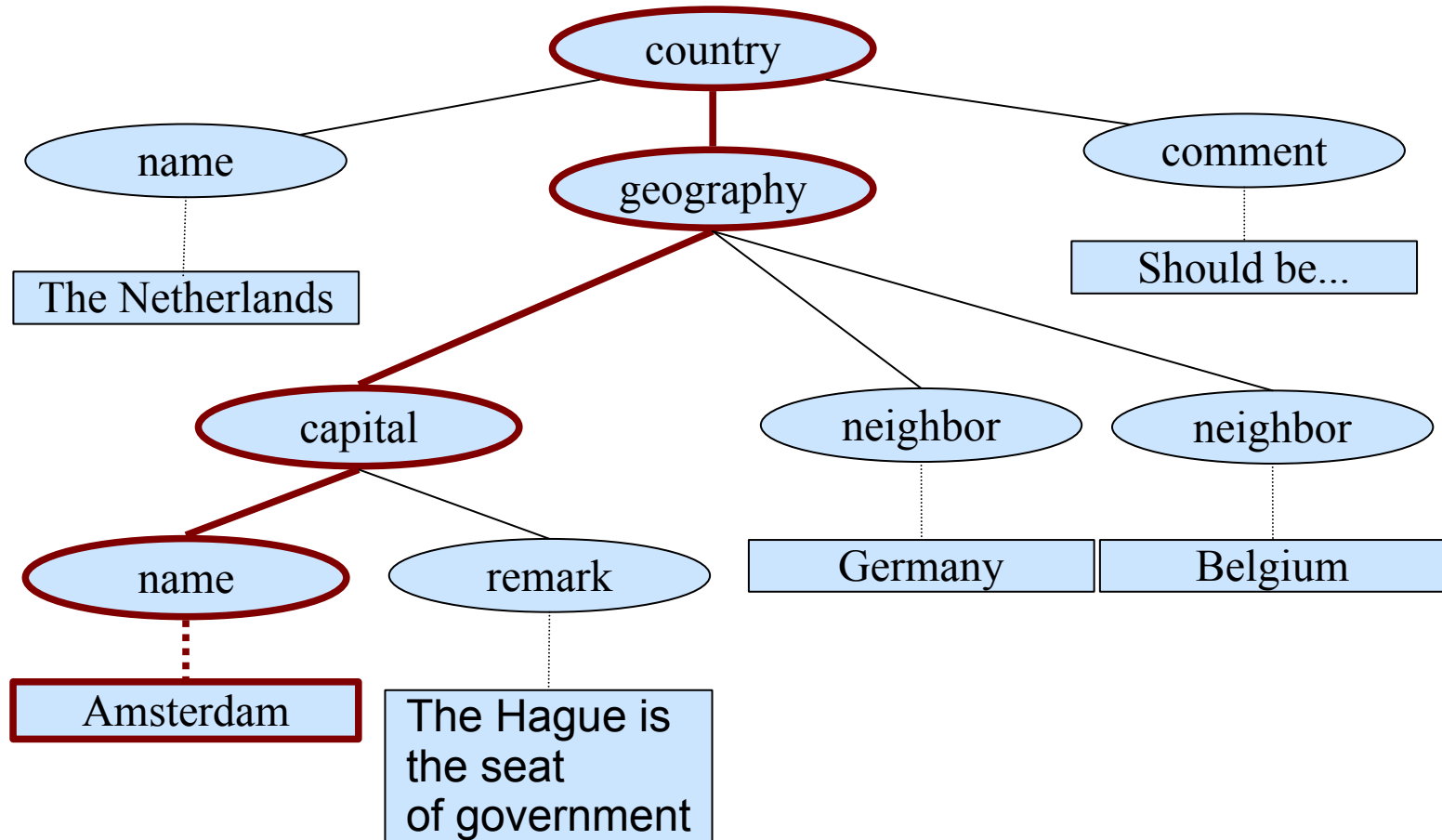
Agenda

- RDF Anfragesprachen
- APIs und Anfragesprachen
- Speicherung von RDF in relationalen Datenbanken

Eine Anfragesprache für RDF

- Frage:
 - RDF hat eine XML Syntax. Warum benutzt man also nicht eine XML-Anfragesprache?
 - XPath
 - XQuery
 - ...

Erinnerung: Pfade in XML



Pfadausdruck:
 /country/geography/capital/@name="Amsterdam"

Das Beispiel in RDF:

```
<rdf:Description rdf:about="#Netherlands">  
  <rdf:type rdf:resource="#Country"/>  
  <geo:hasCapital rdf:resource="#Amsterdam"/>  
</rdf:Description>
```

XML path Ausdruck:

/rdf:Description/[rdf:type/@rdf:resource="#Country"]/geo:hasCapital/@rdf:resource

```
<geo:Country rdf:about="#Netherlands">  
  <geo:hasCapital rdf:resource="#Amsterdam"/>  
</geo:Country>
```

XML path Ausdruck:

/geo:Country/geo:hasCapital/@rdf:resource

Eine Anfragesprache für RDF

- Frage:
 - RDF hat eine XML Syntax. Warum benutzt man also nicht eine XML-Anfragesprache?
 - XPath
 - XQuery
 - ...
- Antwort: Weil das XML Datenmodell im Bezug auf RDF nicht eindeutig ist!

Noch ein Beispiel:

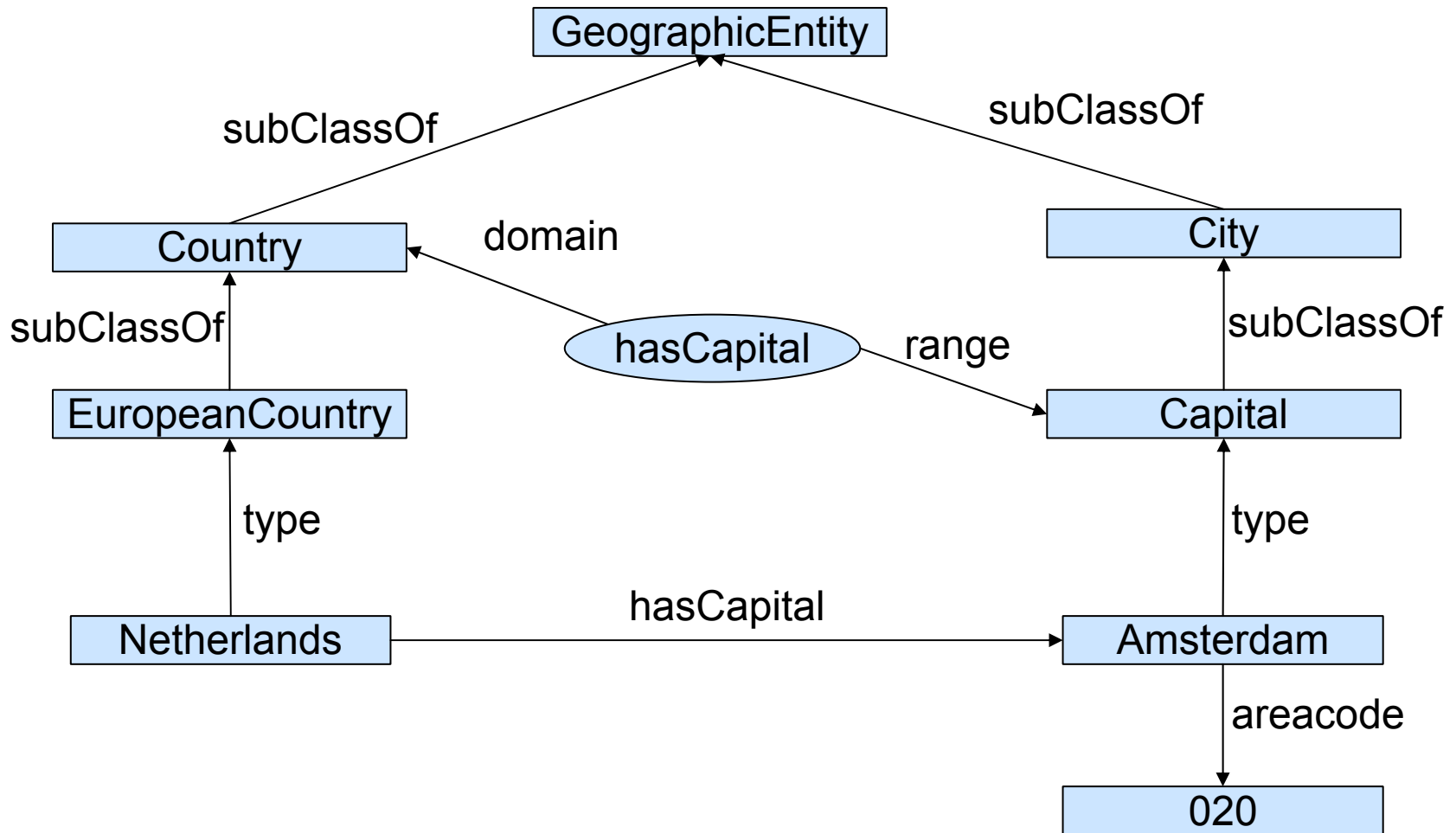
```
<rdf:Description rdf:about="#Netherlands">
  <rdf:type rdf:resource="#Country"/>
  <geo:hasCapital rdf:resource="#Amsterdam"/>
</rdf:Description>

<rdf:Description rdf:about="#Amsterdam"/>
  <geo:areacode> 020 </geo:areacode>
</rdf:Description>
```

Frage:

Welche Relation besteht zwischen den Niederlanden und der Zahl 020 ?

Pfade in RDF



Pfadausdrücke in RDF

- Es gibt kein Wurzelement
 - Pfade können an jedem Knoten starten
- Knoten und Kanten haben Namen
 - Es wird zwischen unterschiedlichen Arten von Relationen unterschieden
- Eine Pfad-basierte Anfragesprache muß diese Aspekte unterstützen

Anforderungen an eine RDF Anfragesprache

- Unterstützung des RDF Datenmodells
 - gerichtet, markiert
 - semi-strukturiert
- Pfadausdrücke
 - Pfade im RDF graphen (nicht im XML Modell)
 - Spezifikation von Knoten und Kanten
- Kompositionalität
 - Komplexe Anfrage können auf einfacheren aufbauen
- Unterstützung von RDF Schema
 - Verwendung der Semantic spezieller preproperties

Stand der Entwicklung

- Viele Vorschläge:
 - RQL
 - RDQL
 - Squish
 - SeRQL (“circle”, Sesame RDF Query Language)
- Zur Zeit Entwicklung eines Sprachstandards “SPARQL”, noch nicht abgeschlossen
- Im folgenden wird SeRQL verwendet

SeRQL

- SeRQL verbindet Features bestehender Anfragesprachen und RDF Syntax-Varianten
- Die Wichtigsten Features:
 - Graphtransformation
 - Unterstützung von RDF Schema Semantik .
 - Unterstützung von XML Schema Datentypen.
 - Kompakte Spezifikation von Pfadausdrücken
 - Optionale Pfadausdrücke Ausdrücke
- Implementierung: Sesame
 - <http://www.openrdf.org/>

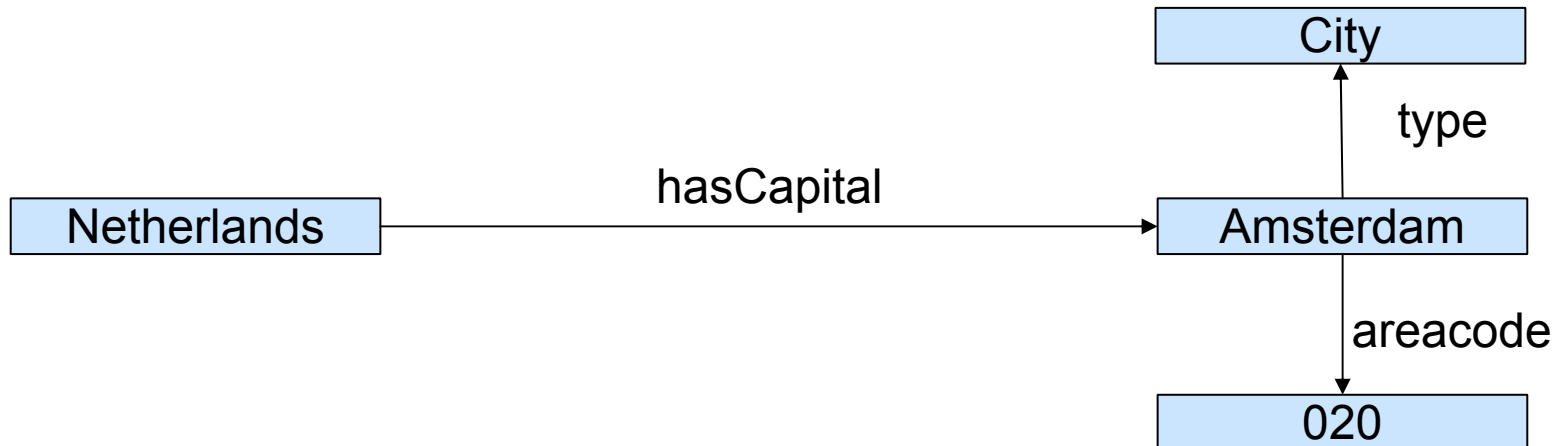
Aufbau einer SeRQL Anfrage

- Struktur in Anlehnung an SQL:
 - select: Rückgabeveriablen
 - from: Pfadausdruck, der gegen das Rdf Modell gematcht wird
 - where: Einschränkung der Variablenwerte
- Beispiel:

```
select X, Y
from {X} geo:hasCapital {Y} geo:areacode {Z}
where Z like "020"
using namespace
      geo = <http://www.geography.org/schema.rdf#>
```

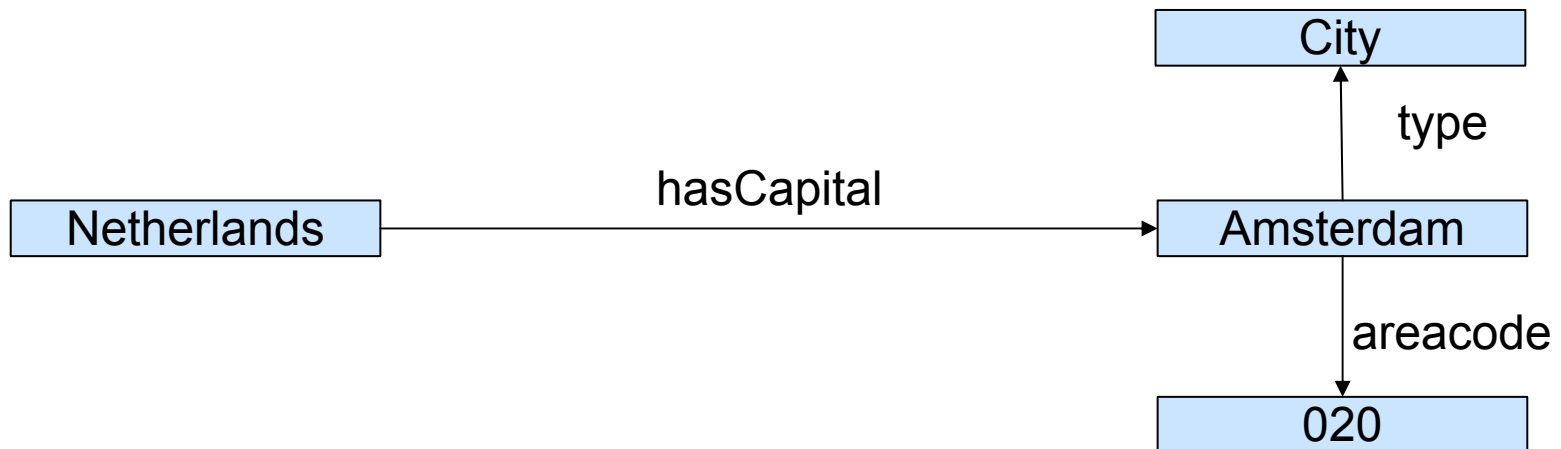
SeRQL Pfadausdrücke

- { X } geo:hasCapital { geo:Amsterdam }
- { X } geo:hasCapital { Y }
- { X } P { Y }



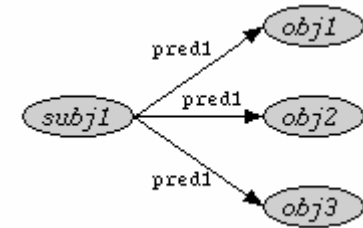
Kompositionsmöglichkeiten

- Chaining:
 - { X } geo:hasCapital { Y } geo:areacode { Z }
- Branching:
 - {X} rdf:type {Y};
 geo:areacode {Z}

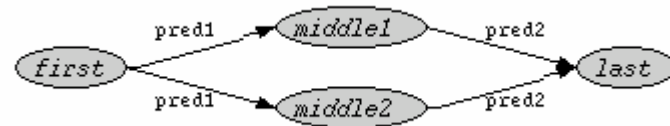


Abkürzungen

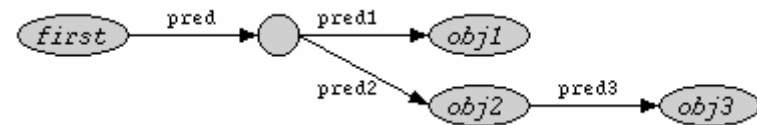
{subj1} pred1 {obj1,obj2,obj3}



{first} pred1 {middle1,middle2} pred2 {last}

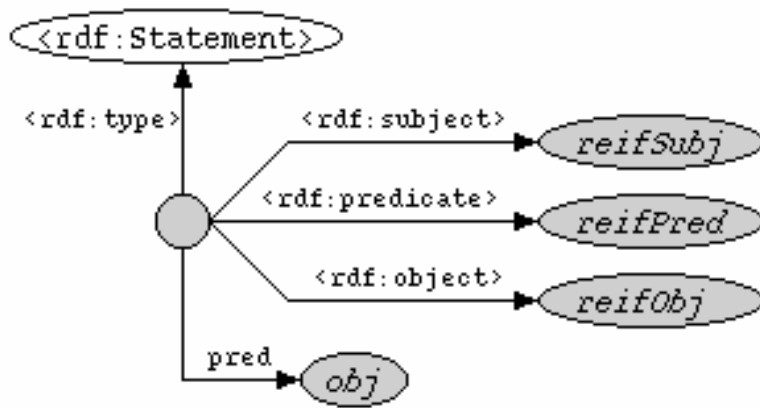


{first} pred {} pred1 {obj1};
 pred2 {obj2} pred3 {obj3}



Reifikation

- Direkte Abfrage:

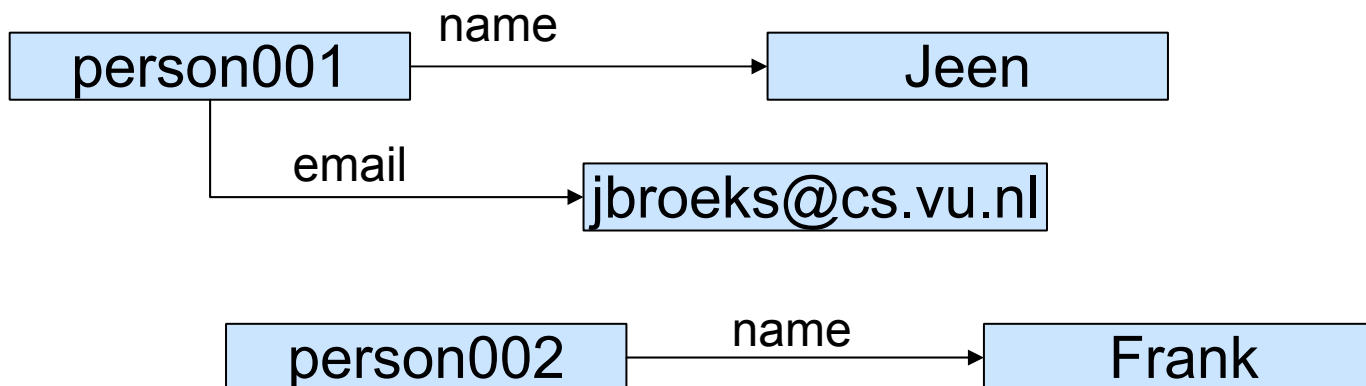


```
{ _Statement } rdf:type { rdf:Statement },
{ _Statement } rdf:subject { reifSubj },
{ _Statement } rdf:predicate { reifPred },
{ _Statement } rdf:object { reifObj },
{ _Statement } pred { obj }
```

```
{ { reifSubj } reifPred { reifObj } } pred { obj }
```

Optionale Pfadausdrücke

- RDF ist semistrukturiert
 - Nicht alle Elemente, die im Schema spezifiziert werden, sind auch in der Daten zu finden:
 - **Personen haben einen Namen und eine email-Adresse, aber nicht alle:**



Optionale Pfadausdrücke (2)

- Anfrage:
 - Die Namen aller Personen und, *wenn bekannt* Ihre email Adresse

```
SELECT
    Person, Name, Email
FROM
    {Person} my:name {Name};
    [my:email {Email}]
```

Optionale Pfadausdrücke (3)

- Optionale Pfadausdrücke können geschachtelt werden
- Wie ist folgender Ausdruck zu interpretieren?

```
{Document} ex:title {Title};  
                [ex:author {Author} [ex:name {Name}];  
                [ex:email {Email}]]
```

Wertevergleich

- Variablenwerte können weiter eingeschränkt werden:
 - Stringvergleich:
 - **X like “*Netherlands”**
 - **Y like “A*”**
 - Boolesche Ausdrücke:
 - **X < Y, X <= Y, Z < 20, Z = Y, etc.**

Wertevergleiche und Datentypen

- RDF unterstützt Datentypen
 - Standard XML schema Datentypen :
 - **xsd:int, xsd:float, xsd:string**
 - Darstellung von Datentypen in RDF:
 - **“20”^^xsd:int**
- Diese Darstellung von Datentypen kann in SeRQL-Anfragen verwendet werden
 - $X < \text{“21”}^{\wedge\wedge}\text{xsd:int}$

Datatypes

- SeRQL versucht Werte für die kein oder ein anderer Datentyp angegeben ist, als im Wertevergleich automatische zu casten:

```
select X, Y
from {X} geo:hasCapital {Y} geo:areacode {Z}
where Z < "21"^^xsd:int and Z > "19"^^xsd:int
using namespace
```

- Ein Wert kann gecastet werden wenn ihm kein Datentyp zugewiesen ist, oder wenn der zugewiesene Datentyp ein Untertyp des im Wertevergleich angegebenen ist
`geo = <http://www.geography.org/schema.rdf#>`

Darstellung von Ergebnissen

- Standard-Rückgabeformat
 - Ergebnistabelle mit Variablenbindungen
 - Spalten: Variablen, Zeilen: Ergebnisse
 - Serialisiert als RDF/XML File
- Alternative
 - Konstruktion eines speziellen RDF Modells
 - Analog zu einer Sicht in relationalen Datenbanken

Beispiel

- Anfrage: Alle Hauptstädte

```
select Y
from {X} geo:hasCapital {Y}
```

- Ergebnis als RDF/XML File:

```
<rs:ResultSet rdf:about=''>
  <rs:resultVariable>Y</rs:resultVariable>
  <rs:solution>
    <rs:ResultSolution>
      <rs:binding rdf:parseType='Resource'>
        <rs:variable>Y</rs:variable>
        <rs:value rdf:resource='http://www.geo.com/cities#London' />
      </rs:binding>
    </rs:ResultSolution>
    <rs:ResultSolution> ....
  </rs:solution>
</rs:ResultSet>
```

Construct-Anfragen

- Construct-Anfragen liefern RDF Modelle
 - Das Ergebnis ist entweder ein Teilmodell des angefragten Modells, oder ein neu konstruiertes:

Teilmodell:

```
construct *  
from {X} geo:hasCapital {Y}
```

Netherlands

hasCapital

Amsterdam

Konstruktion:

```
construct {Y} my:inCountry {X}  
from {X} geo:hasCapital {Y}
```

Amsterdam

inCountry

Netherlands

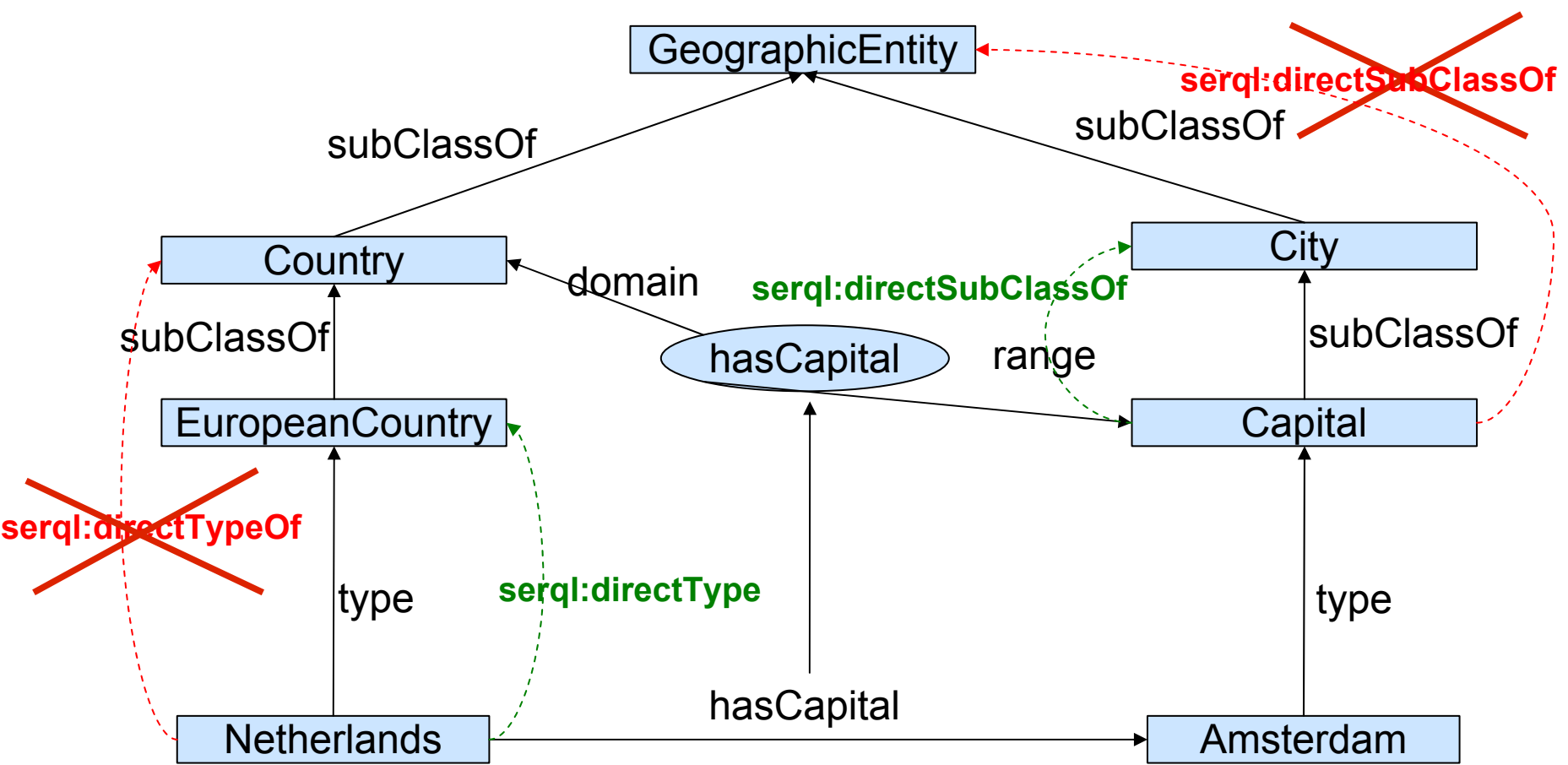
Anfragen und Schemas

- SeRQL unterstützt die RDF Schema Elemente
 - rdfs:class, rdfs:subClassOf
 - rdfs:subPropertyOf, rdfs:domain, rdfs:range
 - ...
- SeRQL unterstützt die RDF Schema Semantik
 - Es werden nicht nur explizit vorhandene Fakten abgefragt, sondern auch abgeleitete Informationen
 - **Objekttypen und abgeleitete Relationen**
 - **Transitive Vererbungsrelationen**

Built-In Prädikate

- Spezielle Prädikate:
 - {X} serql:directSubClassOf {Y}
gilt wenn
 - **X subclassOf Y und;**
 - **X ≠ Y und;**
 - **Es gibt keine Klasse Z so dass**
 - X subclass Z und Z subclass Y
 - {X} serql:directSubPropertyOf {Y}
 - {X} serql:directType {Y}
- Wofür braucht man diese Prädikate ?

Built-In Prädikate



Beispiel für Schema Anfragen

- Anfrage: Alle Instanzen der Klasse Country

```
select X
from {X} rdf:type {geo:Country}
```

- Anfrage : Alle ressourcen, die eine Hauptstadt haben und deren Typ

```
select X, Z
from {X} geo:hasCapital {Y};
        serql:directType {Z}
```

Beispiele für Schema Anfragen (2)

- Anfragen: Alle Unterklassen von GeographicEntity:

```
select X
from {X} rdfs:subClassOf {geo:GeographicEntity}
```

- Anfrage: Die Eigenschaften aller Objekte vom Typ Country

```
select X, P, Y
from {X} rdf:type {geo:Country};
      P {Y}
where P != rdf:type
```

Material

- SeRQL Manual
<http://www.openrdf.org/doc/SeRQLmanual.html>
- Sesame Framework
<http://www.openrdf.org/>
- Other stuff:
 - Dave Beckett's RDF Resource guide
<http://www.ildt.bris.ac.uk/discovery/rdf/resources/>
 - RDQL and RQL online tutorial + demo
<http://www.openrdf.org/>

Andere Anfragesprachen

	N3	Versa	RDQL	SeRQL	RQL
Path expressions	+	+	+	+	+
Optional path expressions	-	-	-	+	+/-
Containers	+/-	+/-	+/-	+/-	+/-
Reification	-	-	-	+	-
Schema support	+/-	+/-	+/-	+	+
Recursion	+	+	-	-	-
Namespaces	+	+	+/-	+/-	+/-
Language tag	-	-	-	+	-

(Siehe:

P. Haase, J. Broekstra, A. Eberhart and R. Volz:

“A Comparison of RDF Query Languages”. Proceedings of ISWC2004)

<http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/>

Agenda

- RDF Query Languages
- APIs and Anfragesprachen
- Storing RDF Data in relational databases

APIs

- APIs bieten eine Alternative Möglichkeit, auf RDF Modelle zuzugreifen
 - Analog zu XML Technologien
- In praktischen Anwendungen werden häufig beide Alternativen verwendet:
 - Anfragen: Retrieval von komplexen Zusammenhängen
 - APIs: Modifikation spezieller Aspekte der Modells oder Anfrageergebnisse

Ein Szenario

“I have a large RDF model containing information about employees. Each employee has a salary associated with him. The salary of each **employee needs to be increased by 5%.**”

(Alternatives Szenario für Deutschland: “increase work week from 38 to 40 hours”)

Anfrage an Sesame

- Extraktion des Teilmodells in dem Gehälter repräsentiert sind:

```
String query =  
    "construct {employee} <corp:salary> {salary}  
    from      {employee} <rdf:type> {<corp:Employee>};  
             <corp:salary> {salary}";
```

```
Graph subGraph = repository.performGraphQuery(QueryLanguage.SERQL, query);
```

Update des Modells

```
StatementIterator iter = subGraph.getStatements();

// loop over all statements in the subgraph
while (iter.hasNext()) {
    Statement st = iter.next();
    Literal object = (Literal)st.getObject();
    float salary = Float.parseFloat(object);

    // increase salary by 5%
    salary = salary * 1.05;
    Literal newSalary = new LiteralImpl(new String(salary), XSD:Float);

    // add the new salary to a new graph
    newGraph.add(st.getSubject(), st.getPredicate(), newSalary);
}

// finally, replace the information in the repository
repository.remove(subGraph);
repository.add(newGraph);
```

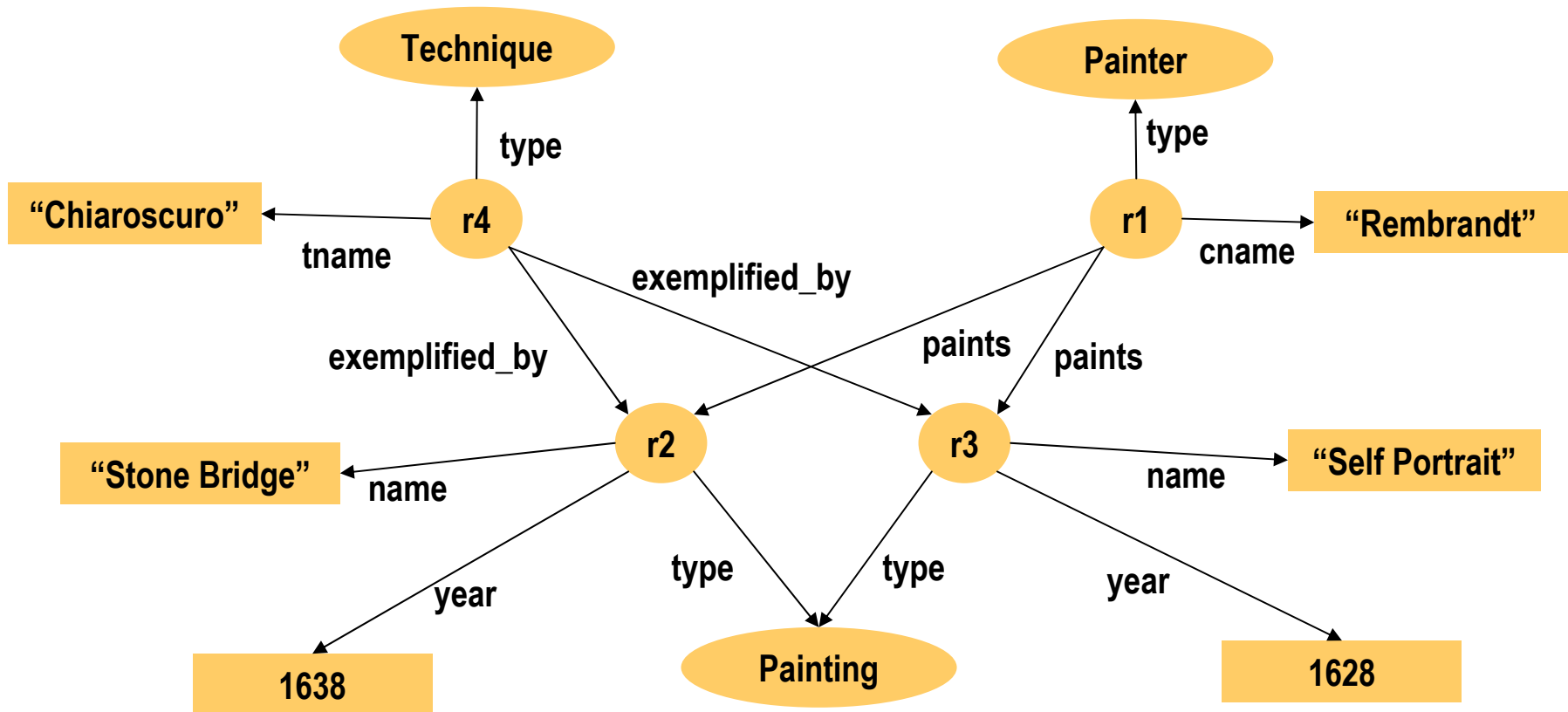
Agenda

- RDF Query Languages
- APIs und Anfragesprachen
- Speicherung von RDF in Datenbanken

Warum ?

- Relationale Datenbanktechnologie ist etabliert und verlässlich
- Bestehende Konzepte bekommt man “umsonst”:
 - Persistenz
 - Transaktionsmanagement
 - Integritätstests
 - Anfrageoptimierung
 - **Indices**
 - **Anfrageplanung**
- Problem: Abbildung von RDF Modell auf Relationen

Ein anderes Beispiel



Horizontale Speicherung

- Vorteile:
 - Einfache Abbildung
 - Einfaches Zielschema
- Nachteile:
 - Sehr viele Spalten
 - “Sparseness”: Viele leere Einträge
 - Update ist aufwendig: das Schema kann sich ändern
 - Schlechte Speicherkomplexität
 - Schlechte Uploadzeiten

resource	type	name	year	cname	paints	tname	exem.
r1	Painter			Rembrandt	r2, r3		
r2	Painting	Stone Bridge	1638				
r3	Painting	Self Potrait	1628				
r4	Style					chiaroscuro	r2, r3

Vertikale Speicherung

- Vorteile
 - Einfaches Modell
 - Festes Schema
 - Unterstützt Streaming
- Nachteile
 - Sehr große Tabelle
 - Tablescan ist aufwendig
 - Viele Joins notwendig

Subject	Predicate	Object
r1	type	Painter
r1	cname	Rembrandt
r1	paints	r2
r1	paints	r3
r2	name	Stone Bridge
r2	year	1638
r2	type	Painting
r3	name	Self Portrait
r3	year	1628
r3	type	Painting
r4	type	Style
r4	tname	chiaroscuro
r4	exemplified_by	r2
r4	exemplified_by	r3

Gemischte Modelle (1)

- Horizontale Darstellung jeder einzelnen Klasse
- Vorteile
 - Analog zu horizontaler Speicherung
 - Kleinere Tabellen
 - Weniger Sparse
- Nachteile
 - Analog zu horizontaler Speicherung
 - Properties ohne Domäne nicht eindeutig zuzuordnen

Painter:

resource	cname	paints
r1	Rembrandt	r2,r3

Style:

resource	tname	exem.
r4	chiaroscuro	r2,r3

Painting:

resource	name	year
r2	Stone Bridge	1638
r3	Self Portrait	1628

Gemischte Modelle (2)

- Vertikale Speicherung pro Property
- Vorteile
 - Analog zu Vertikaler Speicherung
 - kleinere Relationen
- Nachteile
 - Analog zu Vertikaler Speicherung
 - Potentiell hohe Anzahl an Tabellen

paints:

Subject	Object
r1	r2
r1	r3

year:

Subject	Object
r2	1638
r3	1628

cname:

Subject	Object
r1	Rembrandt

tname:

Subject	Object
r4	Chiaroscuro

type:

Subject	Object
r1	Painter
r2	Painting
r3	Painting
r4	Style

exemplified_by:

Subject	Object
r4	r2
r4	r3

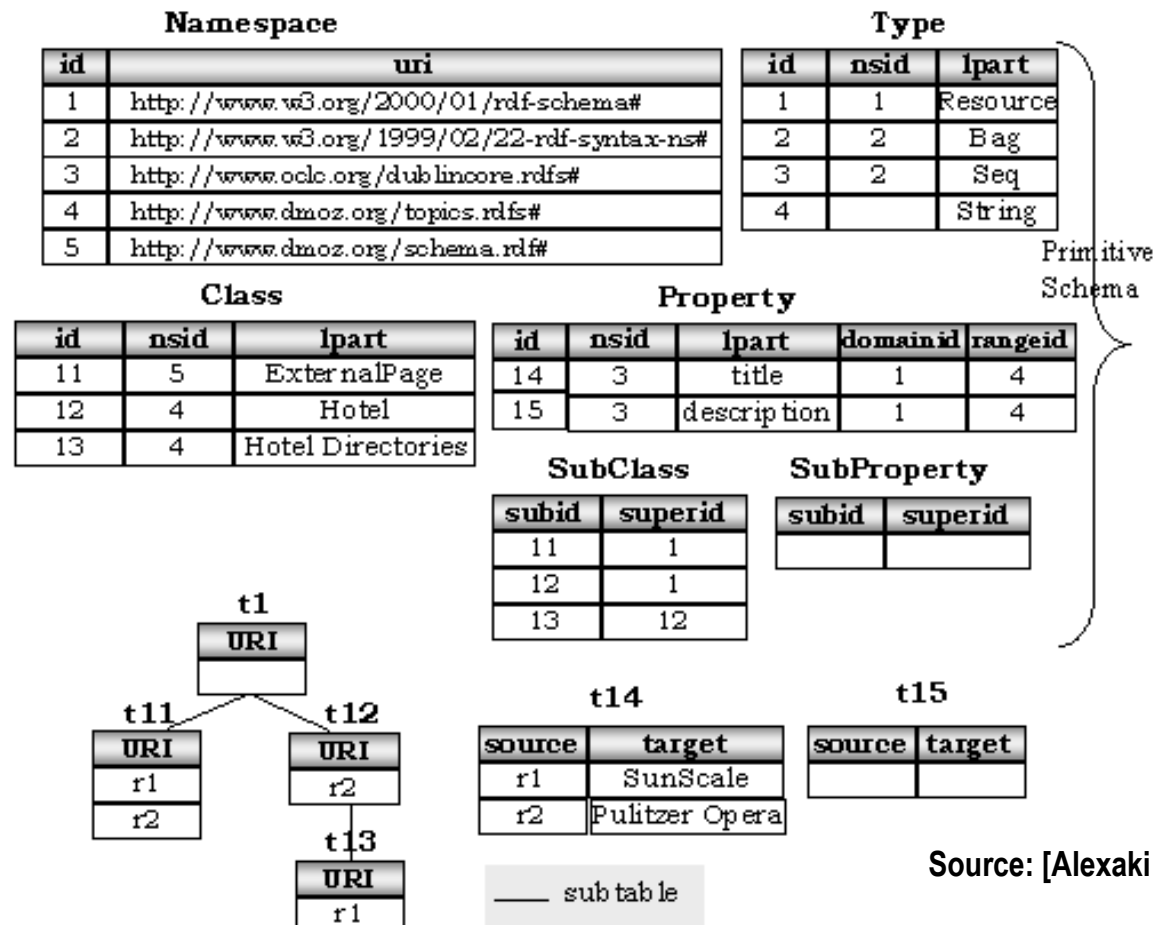
name:

Subject	Object
r2	Stone Bridge
r3	Self Portrait

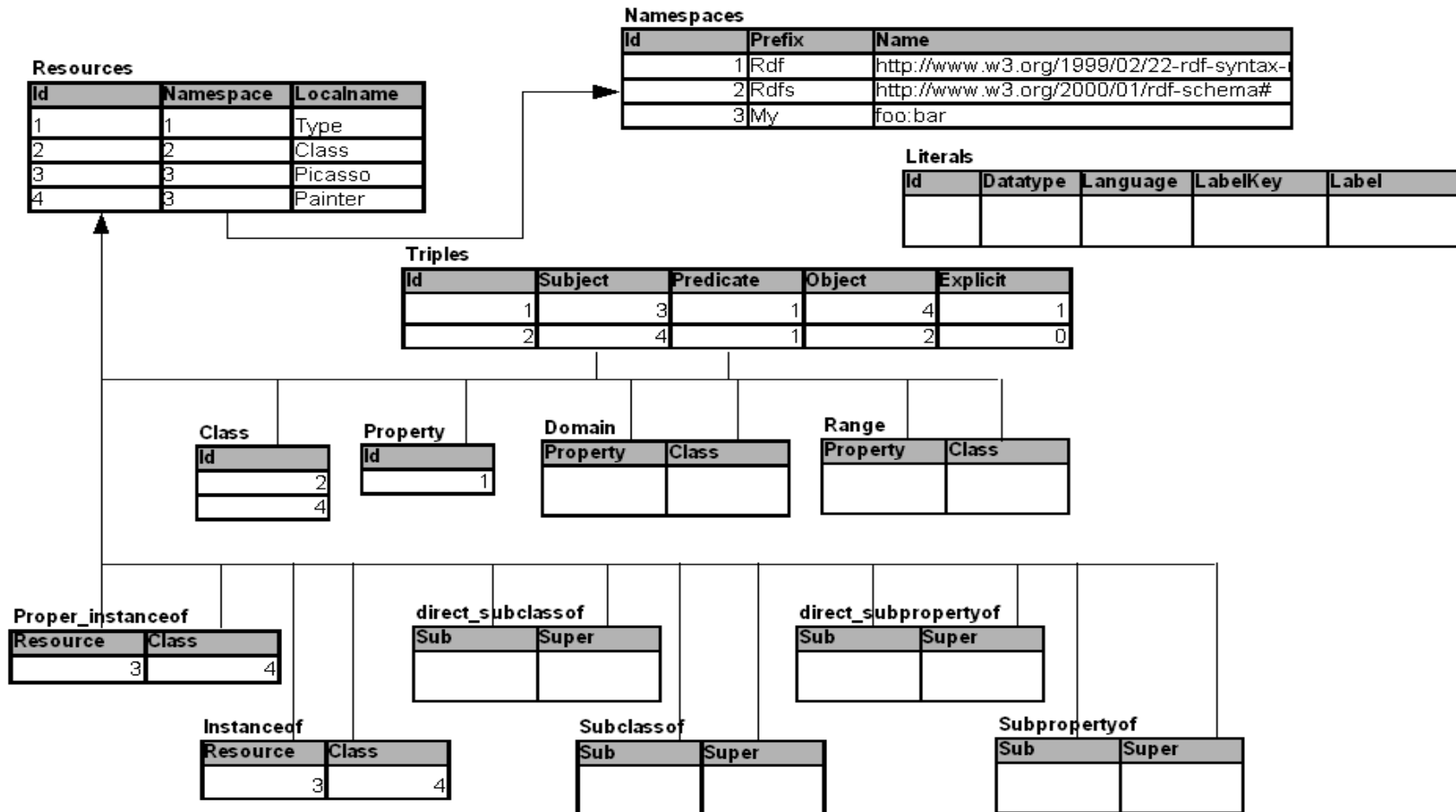
Erweiterungen

- Namespace Tabellen
 - Komplette Namespace Bezeichner werden in separater Tabelle gespeichert, die eigentlichen Datentabellen enthalten nur Referenzen
- Schlüssel und Wertetabellen
 - Lange Bezeichner von Ressourcen und Literalen werden in separater Tabelle gespeichert, Datentabellen enthalten nur Schlüsselwerte
- Index Tabellen
 - Zusätzliche Tabellen, die z.B. Verweise auf alle Instanzen einer Klasse enthalten
 - ...
- Metadaten
 - Zusätzliche Informationen über Herkunft von Informationen bzw. Informationen, ob es sich um abgeleitete Informationen handelt

Beispiel: RDFSuite (Objektrelational)



Beispiel: Sesame (nur relational)



Zusammenfassung

- Vertikale Speicherung hat Vorteile im Bezug auf Performanz und Flexibilität, besonders für irreguläre Daten
- Dekomposition nach Properties erhöht in der regel die die Leistung
- Die Leistung kann durch zusätzliche Maßnahmen weiter gesteigert werden:
 - Indices für Klassen
 - Extra Tabellen für konkrete Werte
 - ...
- Wie immer: abhängig von den konkreten Daten

Material

- Peter Haase, Jeen Broekstra, Andreas Eberhart, Raphael Volz
A comparison of RDF query languages In *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004..*
November 2004.
- Z. Pan and J. Heflin. DLDB: Extending Relational Databases to Support Semantic Web Queries. Technical Report LU-CSE-04-006, Lehigh University, 2004.
- R Agrawal, A Somani, Y Xu - Storage and Querying of E-Commerce Data - Proc. of VLDB, 2001