Effiziente Anfragebearbeitung auf verteilten RDF-Datenquellen

Katja Hose

Max-Planck-Institut für Informatik
Saarbrücken, Germany

17. Januar 2012

Semantic Web und Linked Open Data

Semantic Web

- Web 1.0, Web 2.0, Web 3.0, Web ???
- Informationen für Maschinen “interpretierbar”
- URIs für Dokumente und Konzepte
- Standards: OWL, RDF, RDFS, RDFa, SPARQL, etc.

Linked Open Data Sources

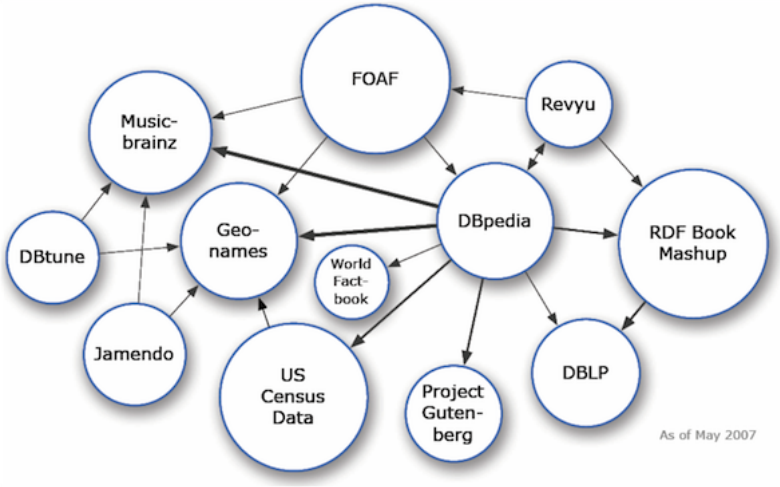
- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

Semantic Web und Linked Open Data

Linked Open Data Sources

- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

Mai 2007

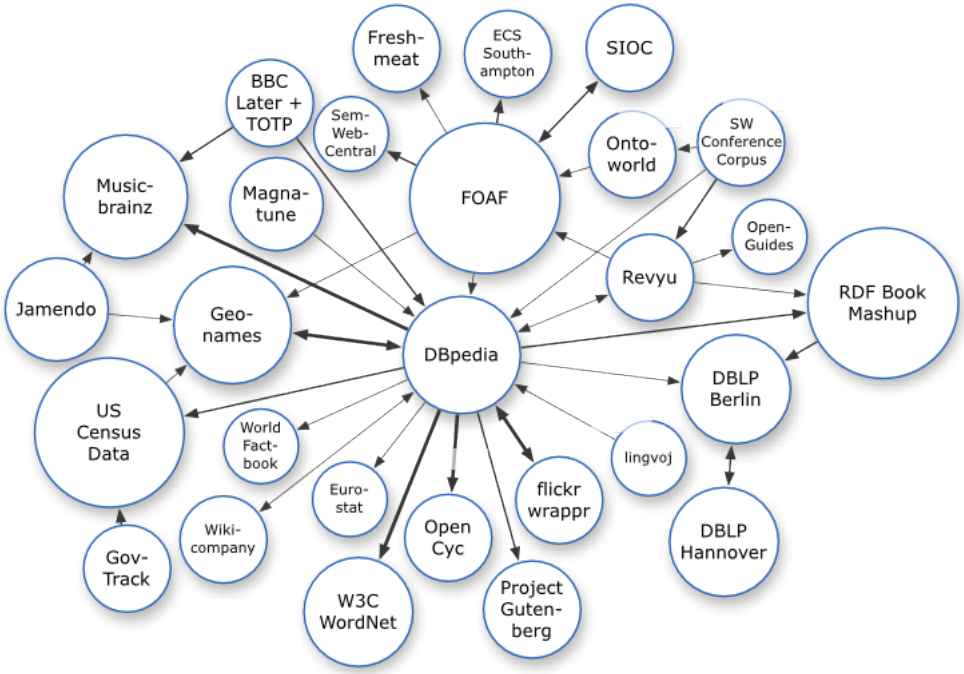


Semantic Web und Linked Open Data

Linked Open Data Sources

- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

November 2007

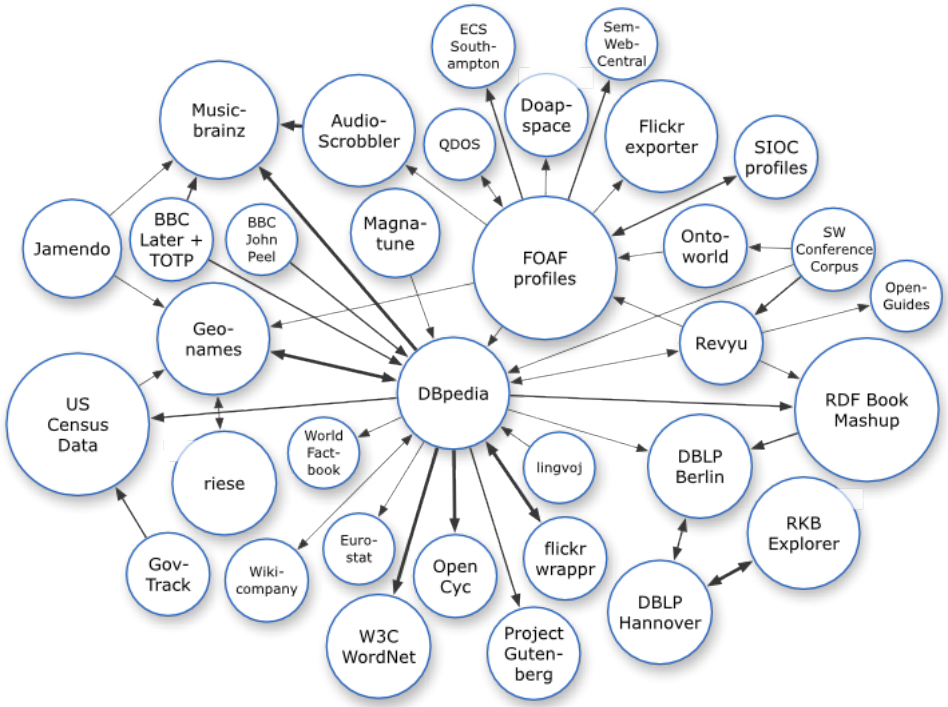


Semantic Web und Linked Open Data

Linked Open Data Sources

- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

März 2008

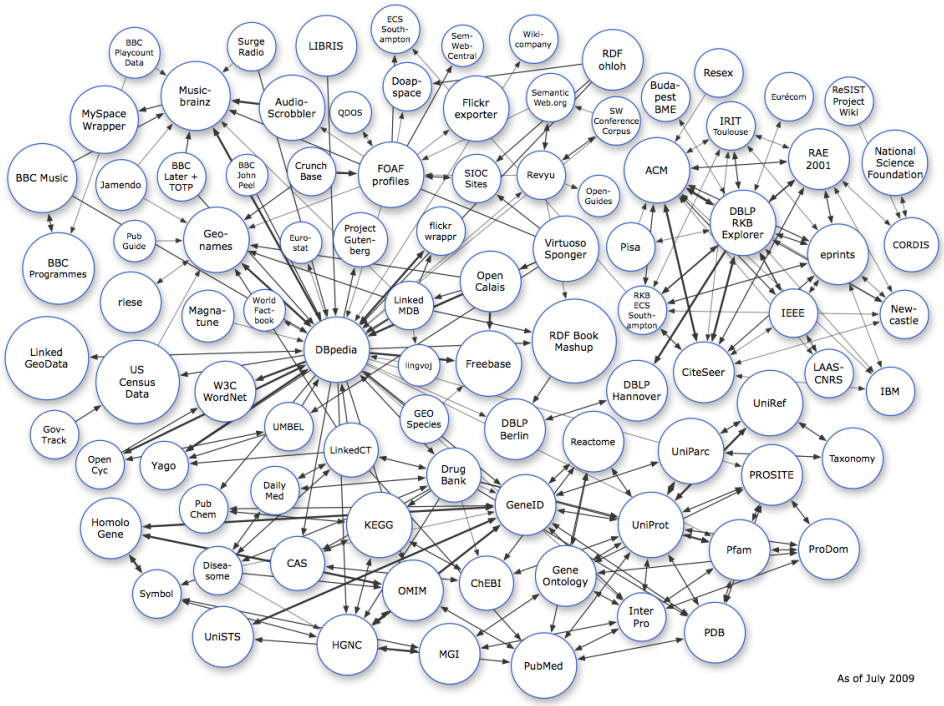


Semantic Web und Linked Open Data

Linked Open Data Sources

- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

Juli 2009

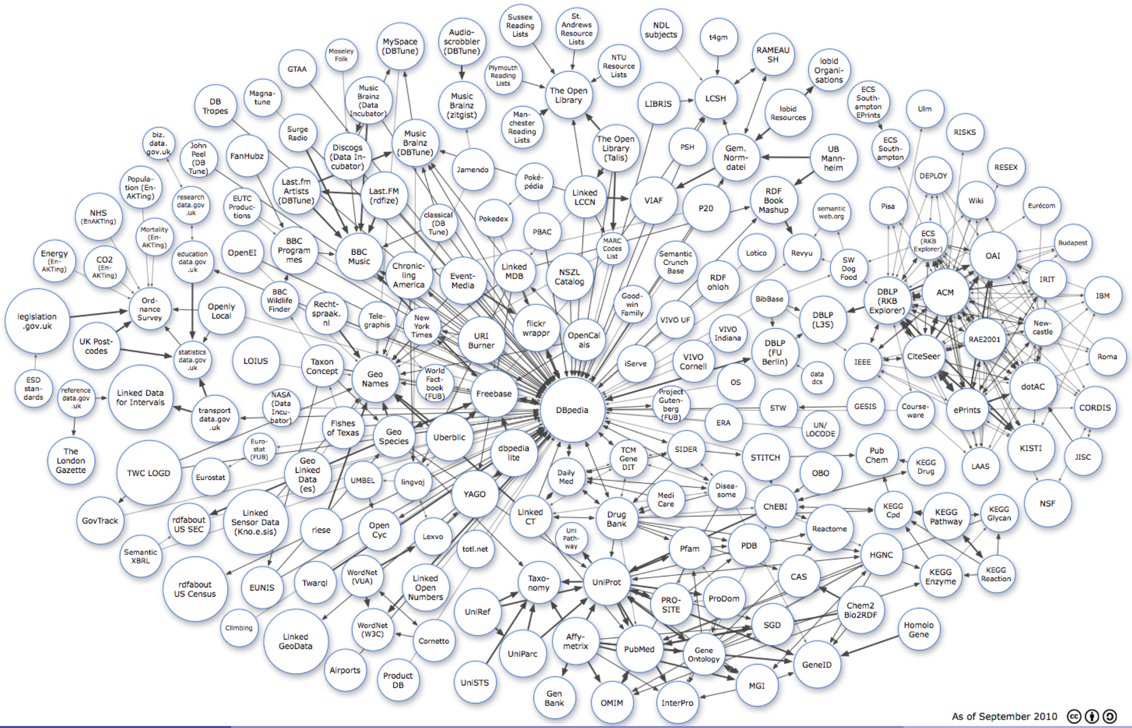


Semantic Web und Linked Open Data

Linked Open Data Sources

- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

September 2010

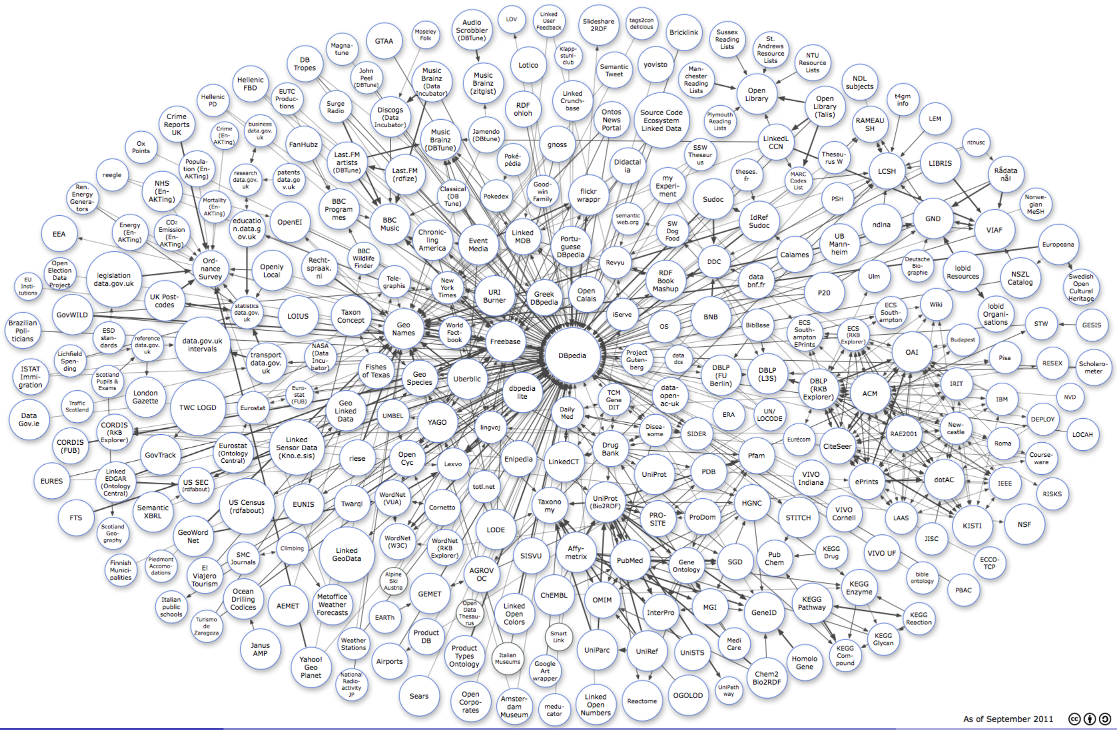


Semantic Web und Linked Open Data

Linked Open Data Sources

- DBpedia, YAGO, OpenCyc, DBLP, ACM, ...
- RDF, SPARQL
- Stetiges Wachstum
- Links/Referenzen zwischen Quellen

September 2011



Linked Open Data

Linked Open Data (LOD)

- Daten liegen als RDF verteilt vor (Media, Geographic, Publications, User-Generated Content, Government, Cross-Domain, Life Sciences)
- Inhaltliche Überlappungen
  - Schauspieler, Filme: DBpedia, Yago, IMDB,...
  - Orte: Geonames, Freebase, NYTimes, DBpedia, Yago,...
- Links beschreiben Beziehungen (RDF Triple)

```
<http://www.imdb.com/name/nm0817881> owl:sameAs
<http://dbpedia.org/resource/Carlo_Pedersoli>
```

Linked Open Data

Design Issues[1]

- Use URIs as names for things
- Use HTTP URIs so that people can look up those names
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
- Include links to other URIs, so that they can discover more things

[1] http://www.w3.org/DesignIssues/LinkedData.html

Datenquellen

Daten stehen auf unterschiedliche Weise zur Verfügung

- SPARQL Endpoints
- Dereferenzierbare URIs
- RDF Dumps
- Annotierte Webseiten
- Web Services
- ...

Wer ist Carlo Pedersoli?

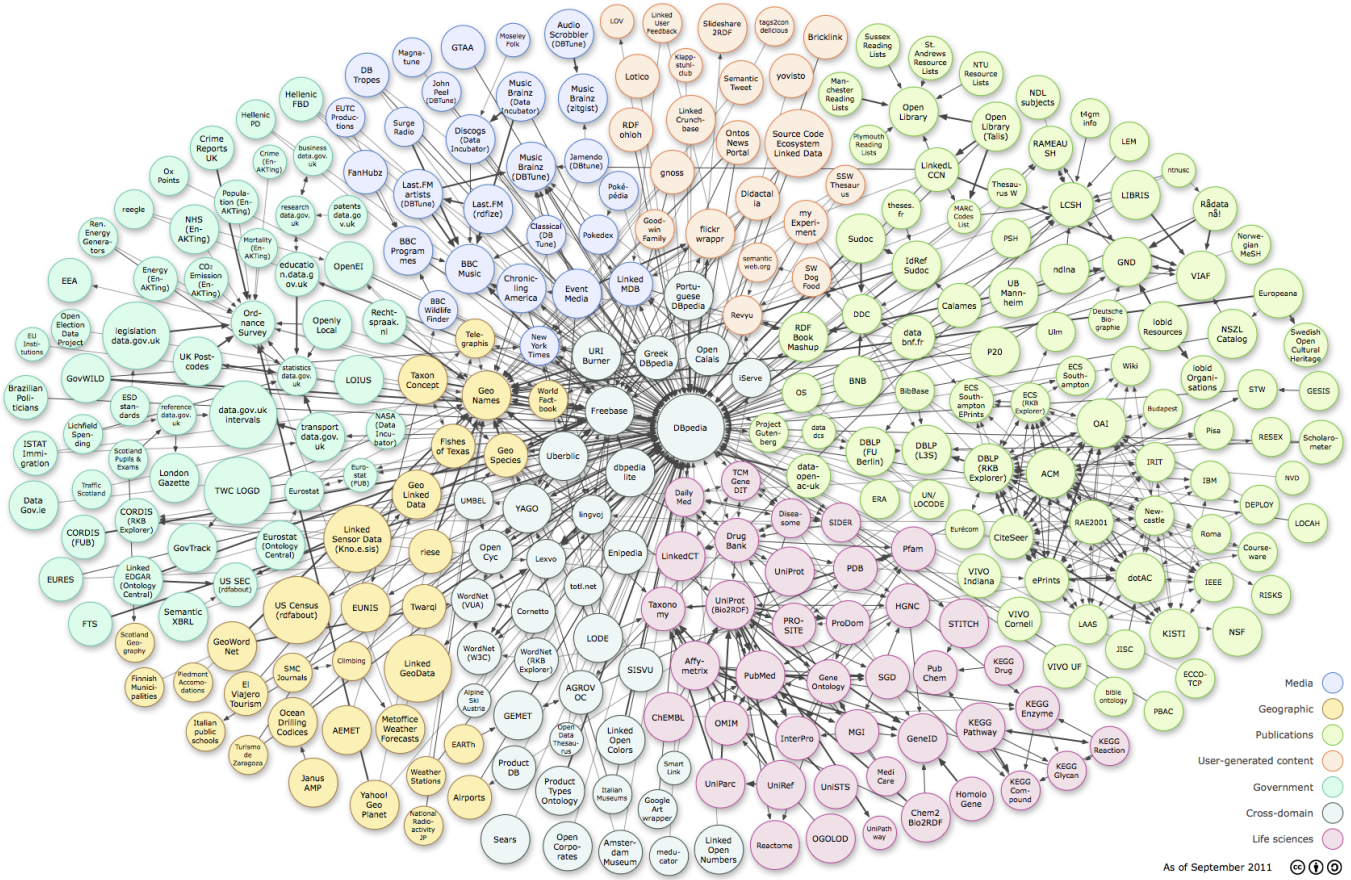
Linked Open Data

Wer ist Carlo Pedersoli?

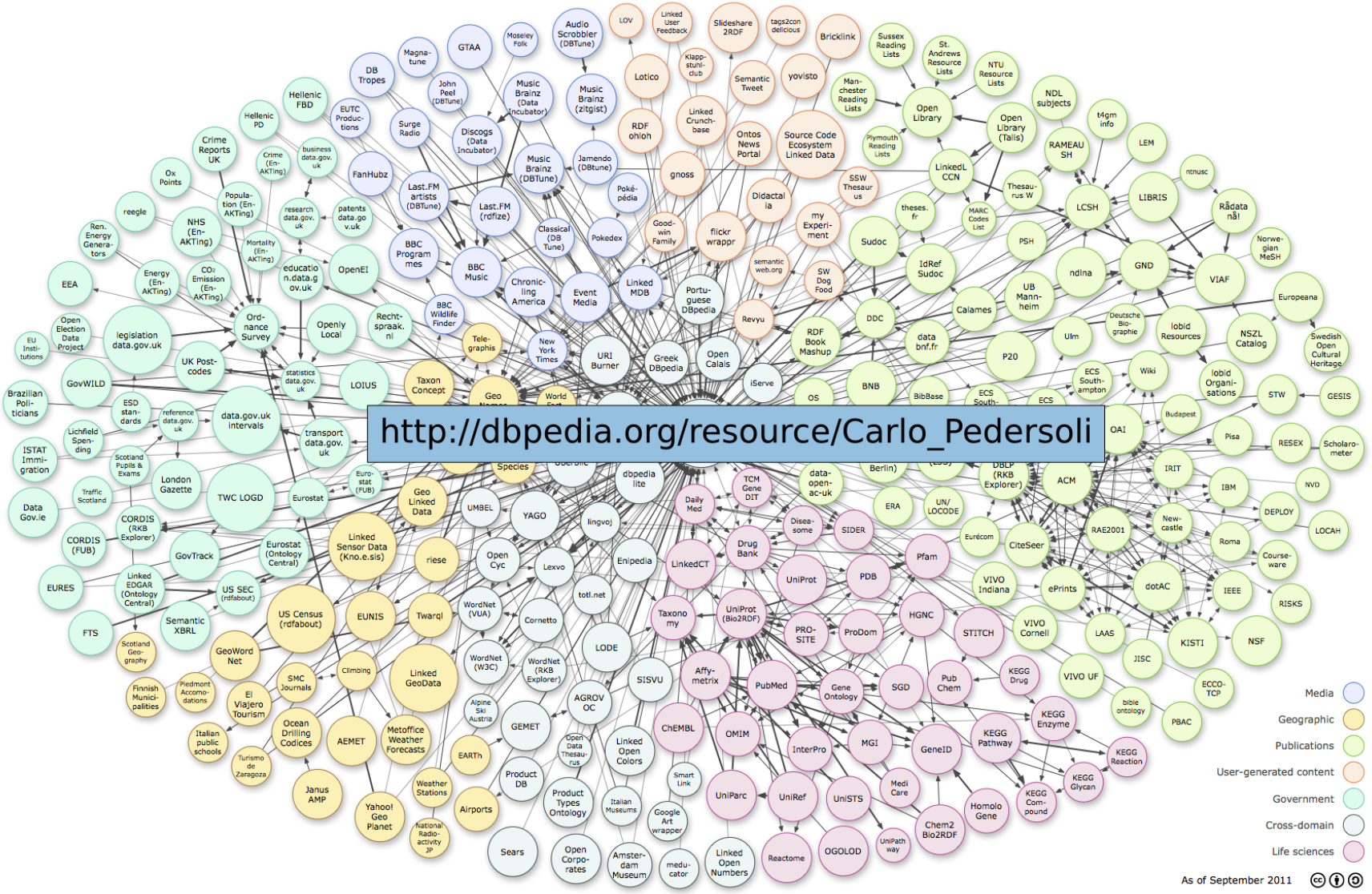
Wer ist Carlo Pedersoli?

Linked Open Data

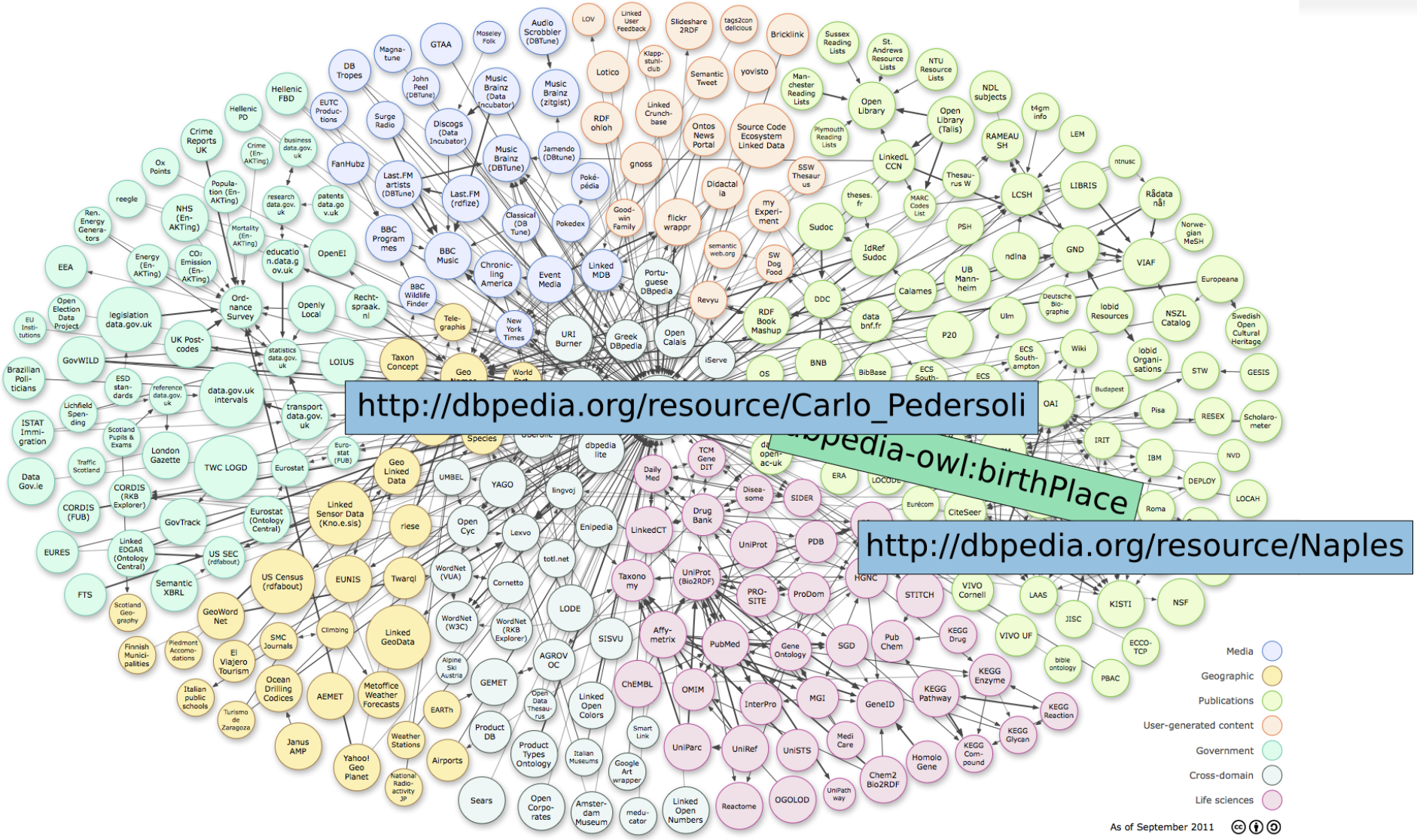
Wer ist Carlo Pedersoli?



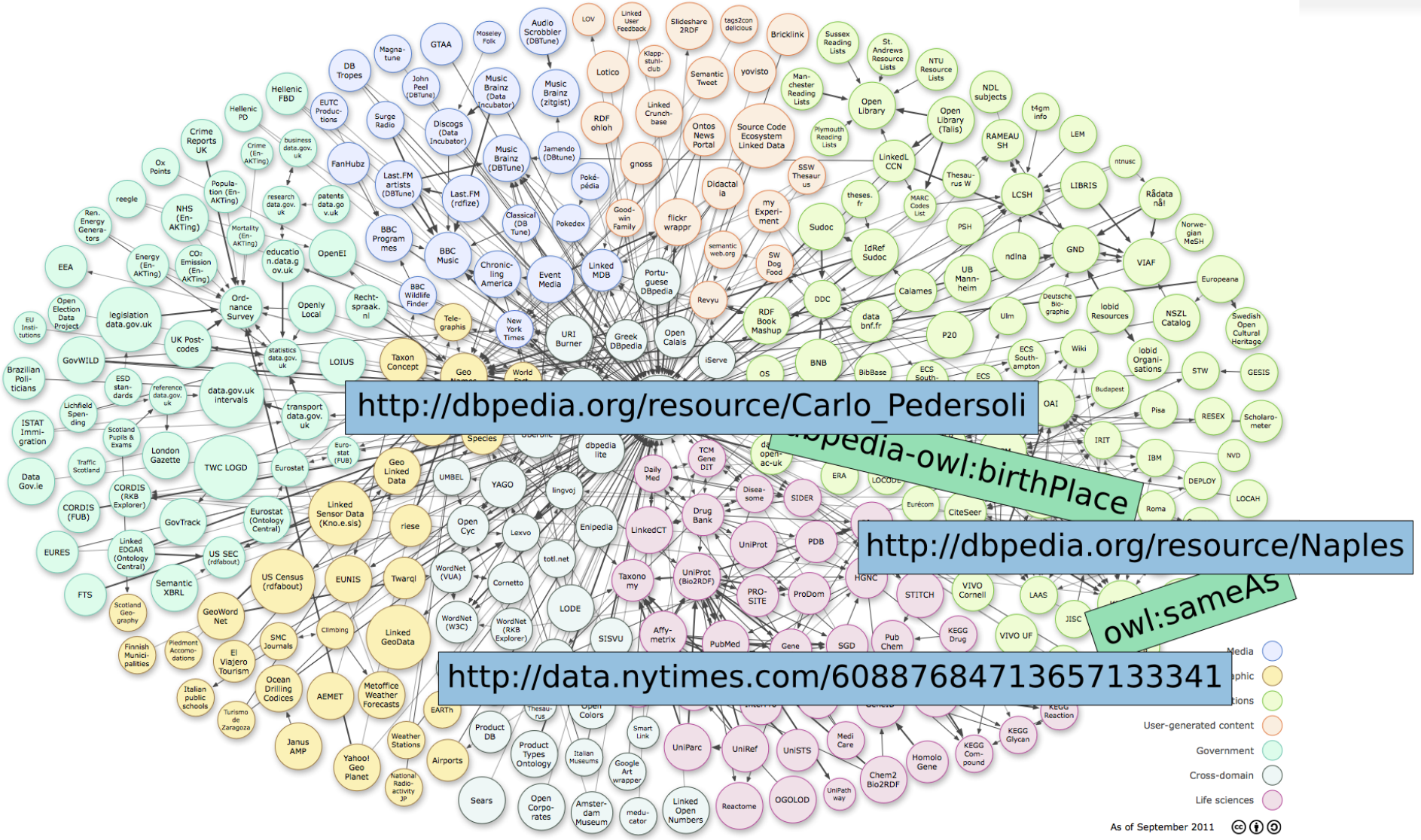
Wer ist Carlo Pedersoli?



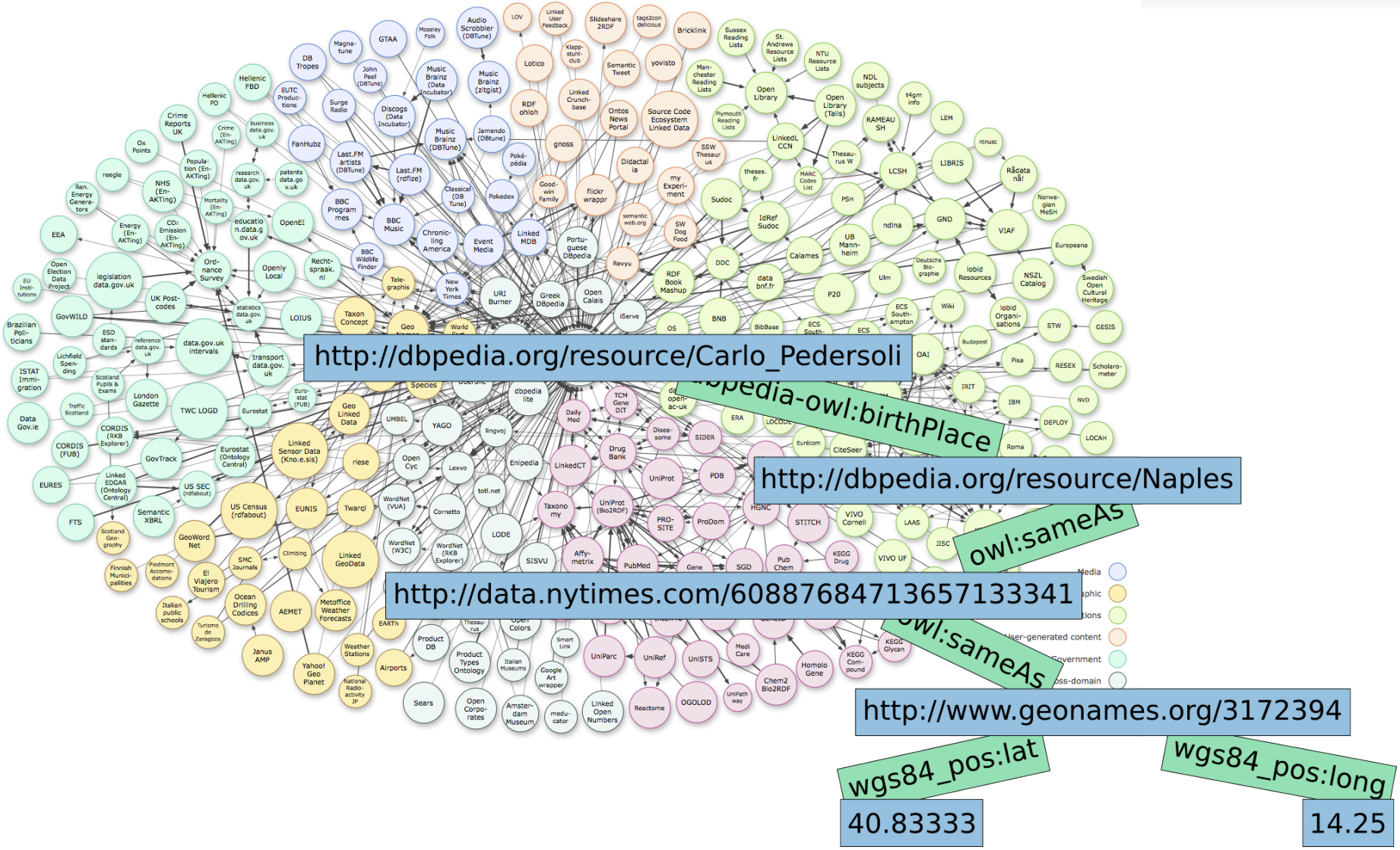
Wer ist Carlo Pedersoli?



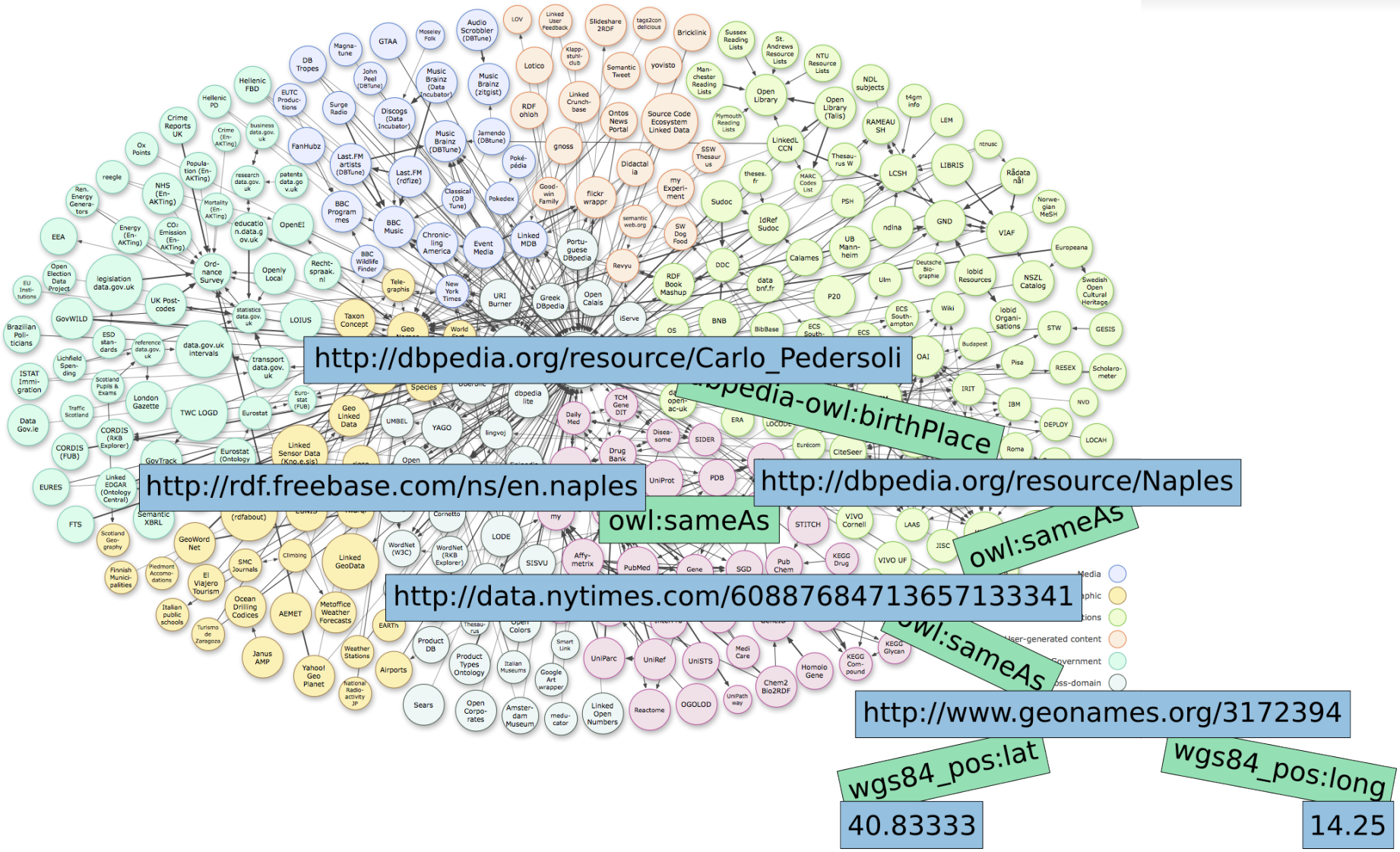
Wer ist Carlo Pedersoli?



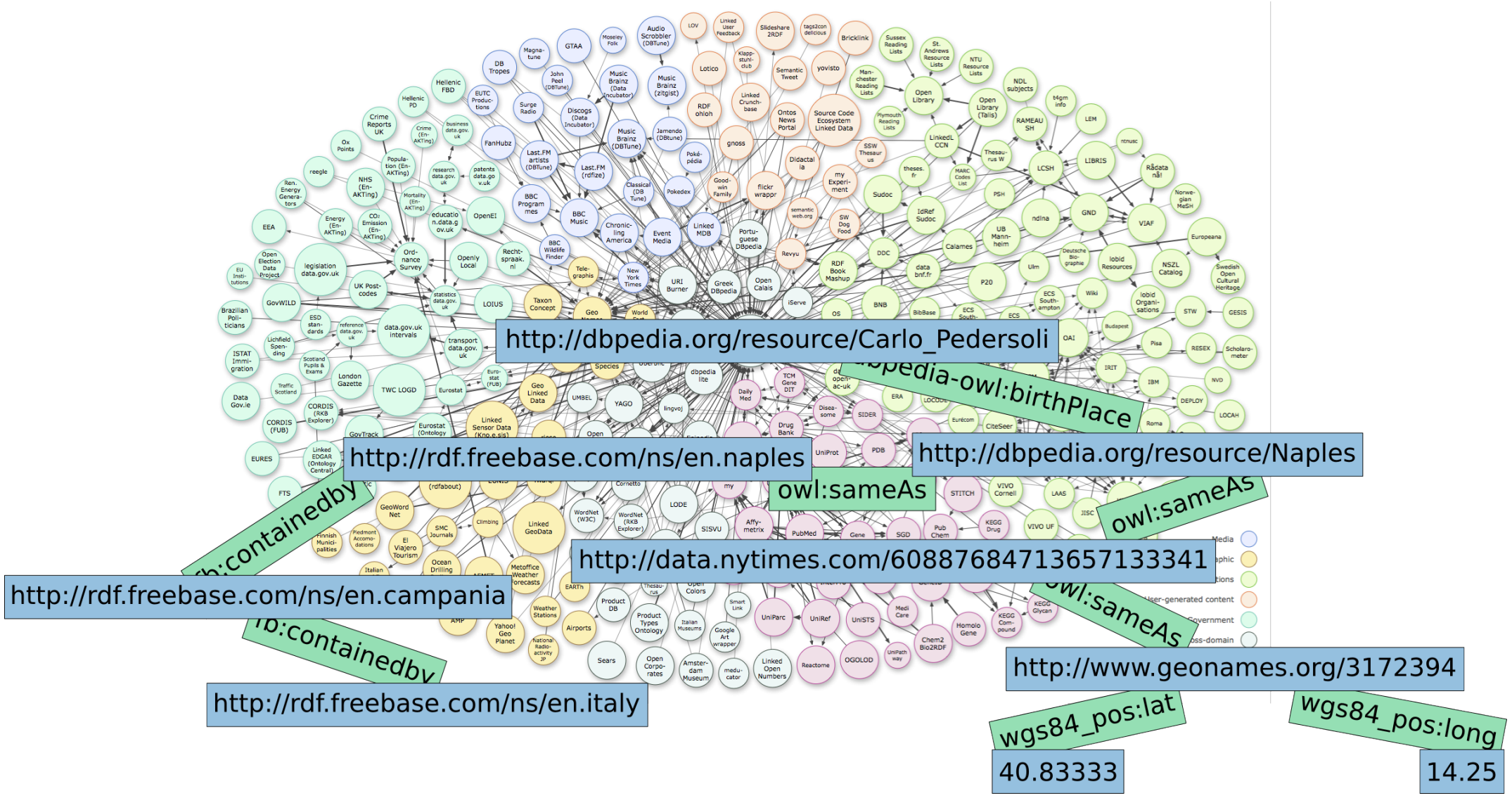
Wer ist Carlo Pedersoli?



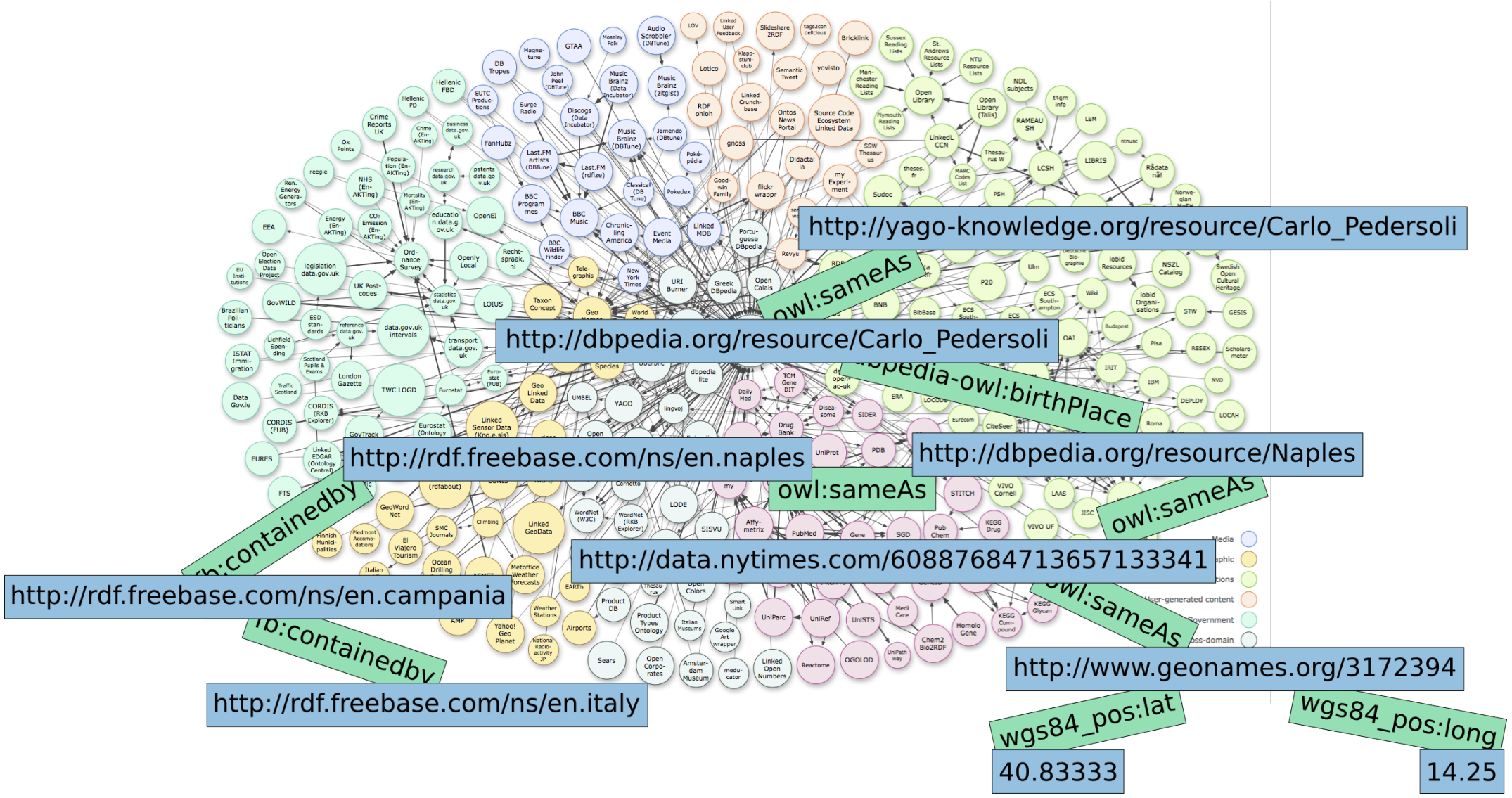
Wer ist Carlo Pedersoli?



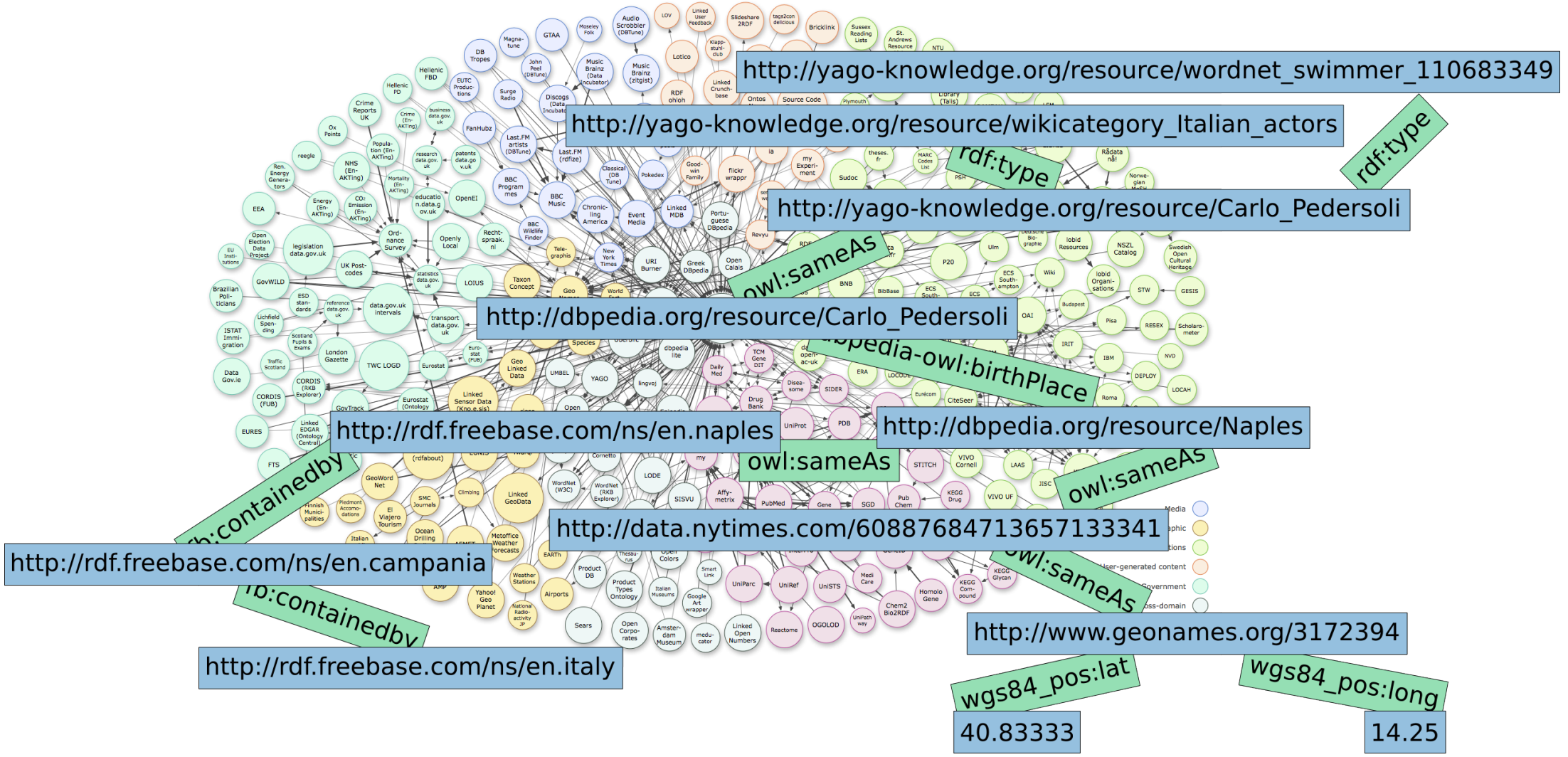
Wer ist Carlo Pedersoli?



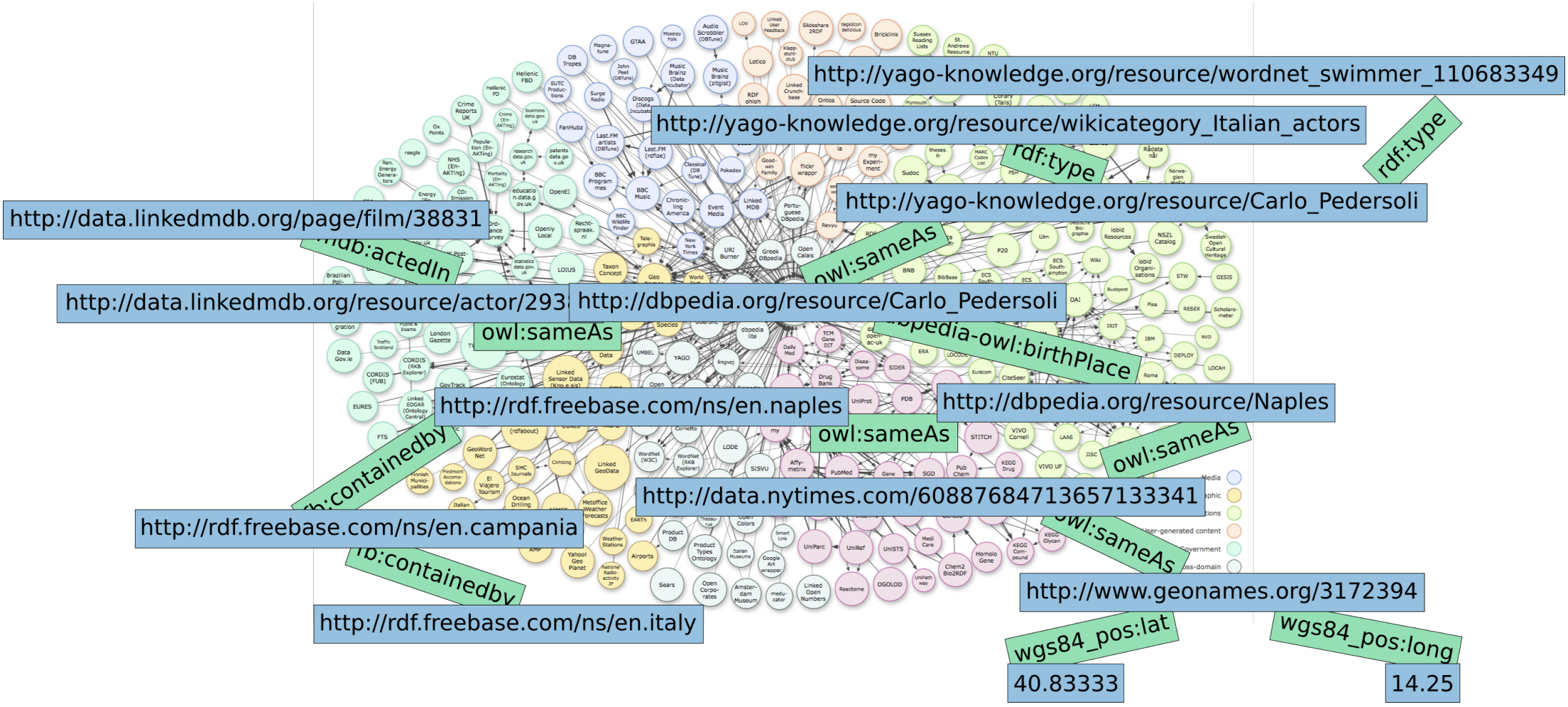
Wer ist Carlo Pedersoli?



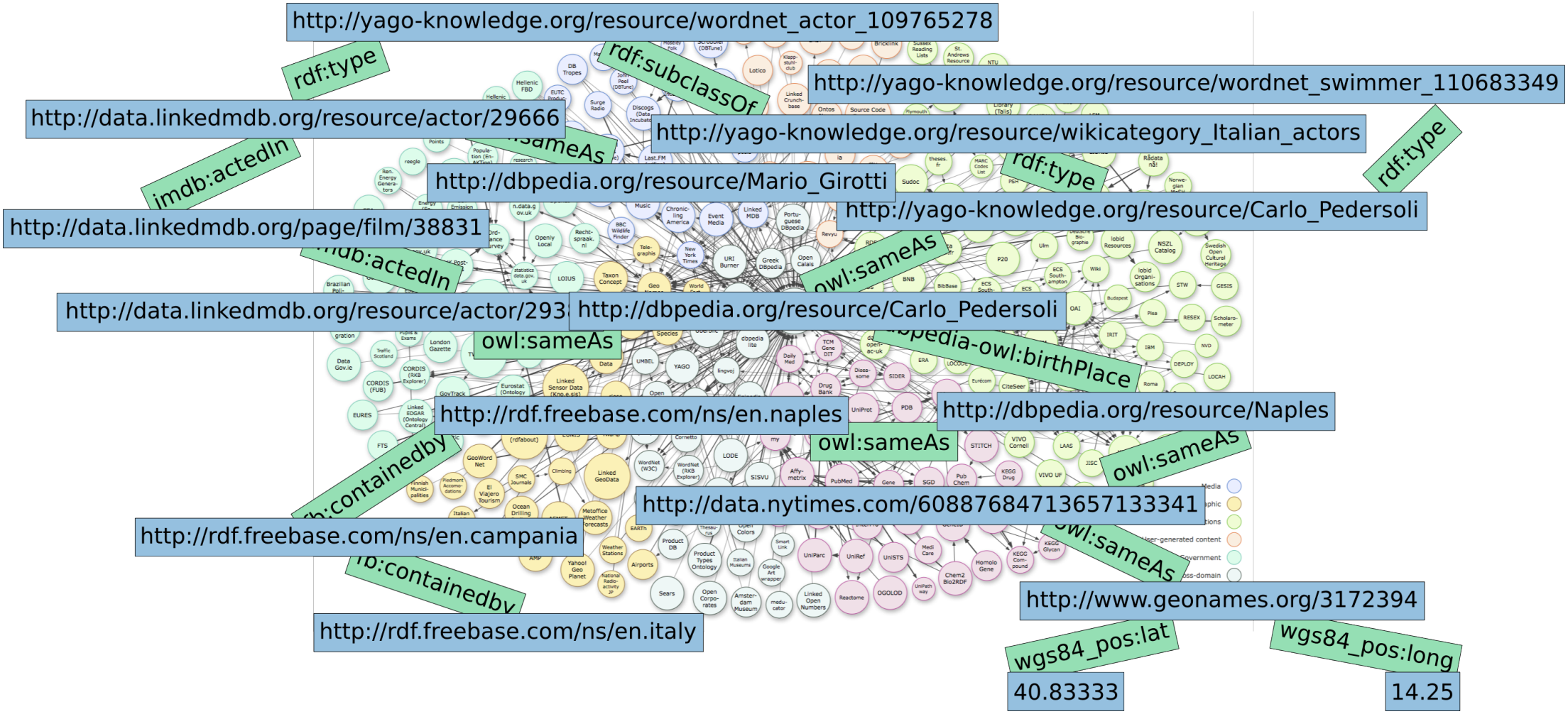
Wer ist Carlo Pedersoli?



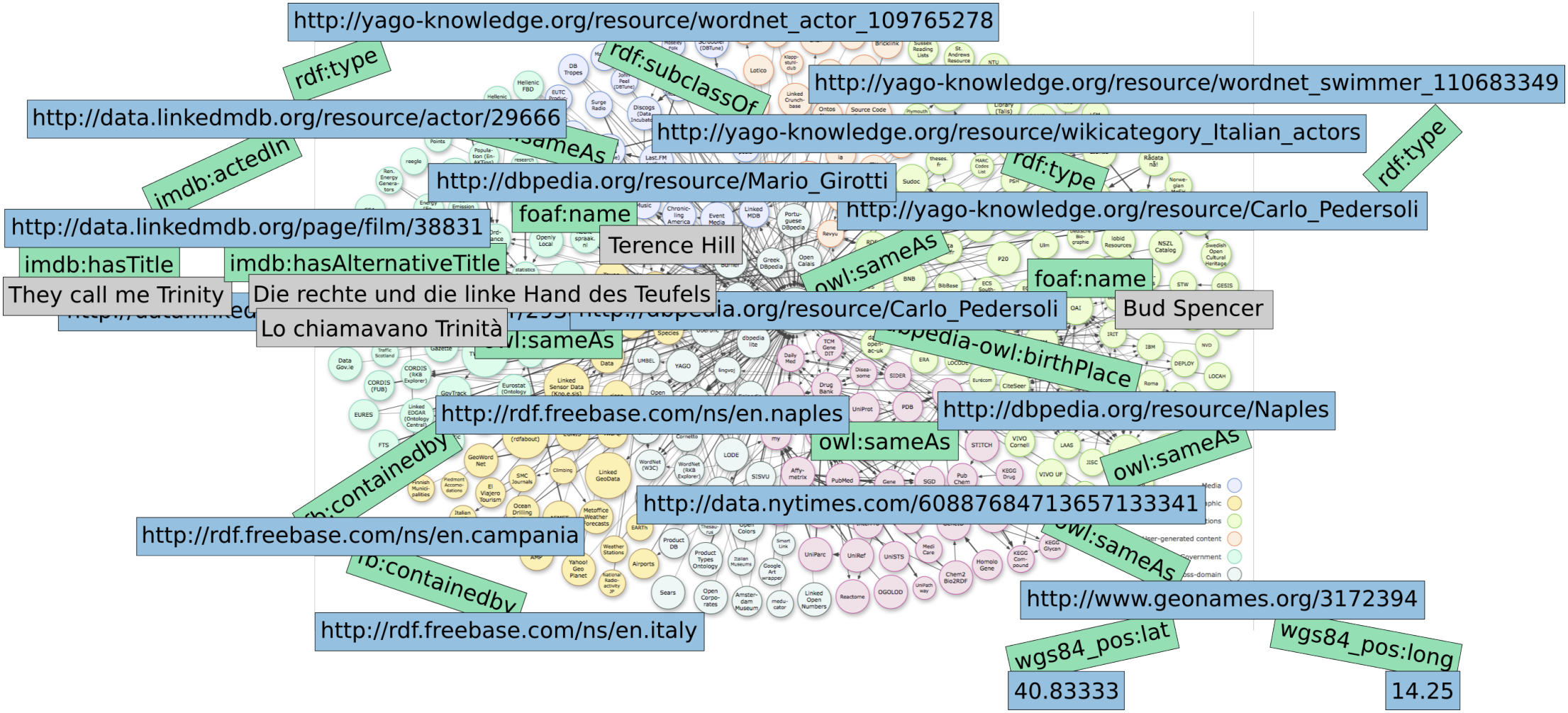
Wer ist Carlo Pedersoli?



Wer ist Carlo Pedersoli?



Wer ist Carlo Pedersoli?



Möglichkeiten der Suche

Wie können Benutzer Daten finden?

- Browsing
  - Start-URI
- Suchmaschinen
  - Keywords
- Systeme zur Anfragebearbeitung in heterogenen Datenquellen
  - Strukturierte Anfragen
    - **SPARQL**
    - RDQL
    - SeRQL
    - RQL
    - RDFQL
    - ...

Suchmaschinen

Initialisierung

- RDF-Daten aus dem Web crawlen
- Index erstellen

Benutzeranfragen bearbeiten

- Input: Schlüsselwort
- Im Index suchen (Index Lookup)
- Output: Liste relevanter Quellen

Probleme

- Updates der Originaldaten

Suchmaschinen

Initialisierung

- RDF-Daten aus dem Web crawlen
- Index erstellen

Benutzeranfragen bearbeiten

- Input: Schlüsselwort
- Im Index suchen (Index Lookup)
- Output: Liste relevanter Quellen

Probleme

- Updates der Originaldaten

Suchmaschinen

Initialisierung

- RDF-Daten aus dem Web crawlen
- Index erstellen

Benutzeranfragen bearbeiten

- Input: Schlüsselwort
- Im Index suchen (Index Lookup)
- Output: Liste relevanter Quellen

Probleme

- Updates der Originaldaten

Typen von Suchmaschinen

Architekturen

- Lokale Kopie der Daten (Local Copy)
- Ausschließlich Indexe (Index Only)

Ergebnistypen

- RDF-Dokumente und -Quellen
- RDF-Entitäten und -Konzepte

Manchmal ist die Zuordnung nicht ganz eindeutig möglich, z.B. Caching

Swoogle

Fokus: Dokumente und Quellen

Crawler, um RDF-Dokumente zu finden

- Beginn zum Beispiel: Google search for „.rdf“
- Metadaten extrahieren

  Encoding (N3, RDF/XML, N-Triple), Language (OWL, RDFS, RDF, DAML),...
- Beziehungen zu anderen Dokumenten bestimmen

  import, extend, references,...
- Google/Jena Crawler
  - Suche nach bestimmten Dokumenttypen (.rdf, .owl, .n3, .daml)
  - Neue Links beim Parsen extrahieren

Swoogle

SQL-Datenbank für Metadaten

- Encoding, Language, Statistiken, Beziehung zu anderen Dokumenten (Imports), etc.

Anfragebearbeitung

- Schlüsselwörter, Field-Constraints (encoding, file type, etc.)
- Ranking ähnlich wie PageRank
- Ergebnis: Links zu Dokumenten

Swoogle

SQL-Datenbank für Metadaten

- Encoding, Language, Statistiken, Beziehung zu anderen Dokumenten (Imports), etc.

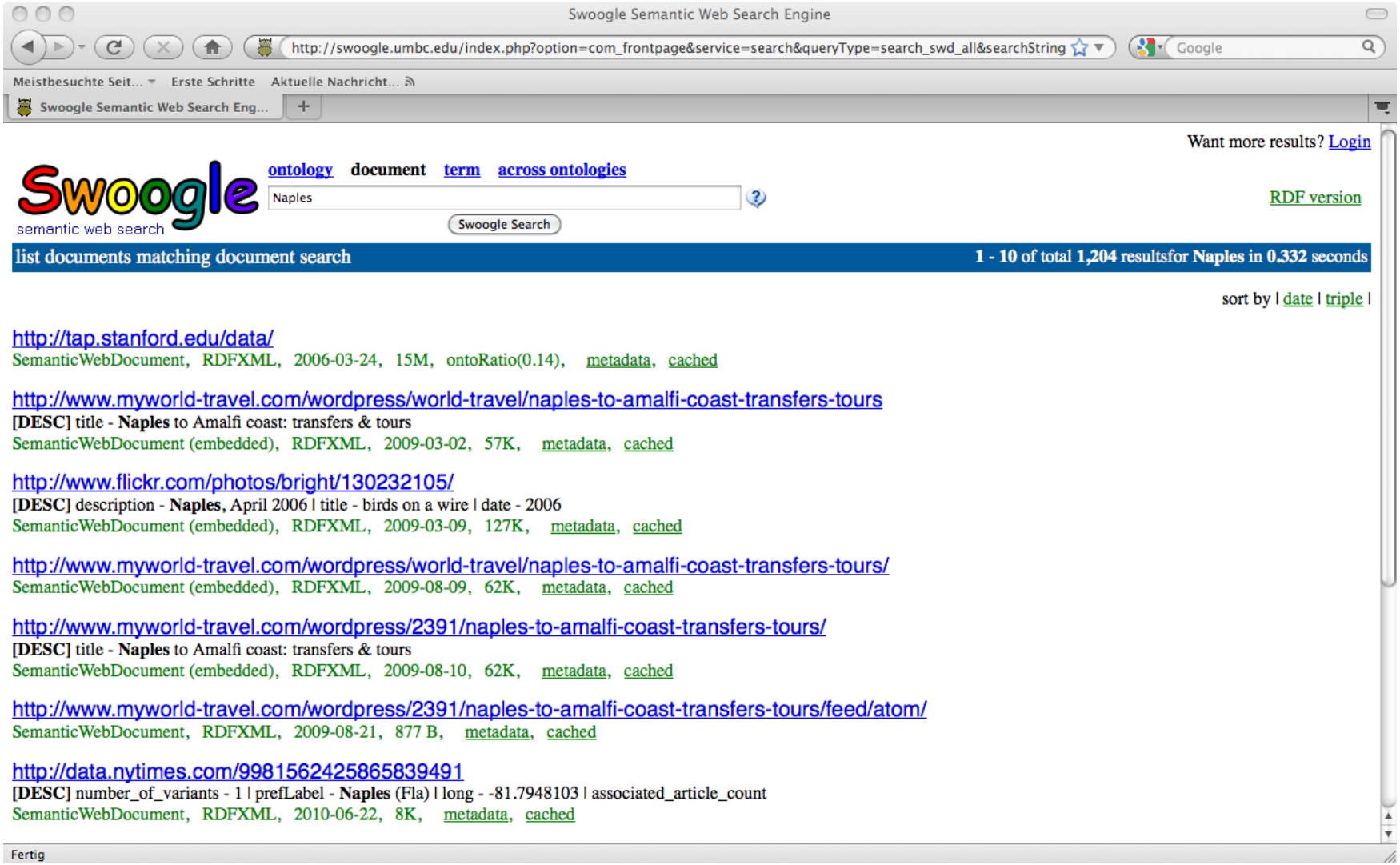
Anfragebearbeitung

- Schlüsselwörter, Field-Constraints (encoding, file type, etc.)
- Ranking ähnlich wie PageRank
- Ergebnis: Links zu Dokumenten

Suchmaske

http://swoogle.umbc.edu/

Suche nach Naples



http://swoogle.umbc.edu/

Falcons

Fokus: Entitäten

Crawling

- Folge den in RDF-Dokumenten enthaltenen Links
- Erstelle virtuelle Dokumente für Entitäten
  - Literals
  - Human-readable Names and Descriptions (rdfs:label, rdfs:comment)
- Indexe erstellen
  - Terme in virtuellen Dokumenten
  - Klassen von Entitäten
- RDF-Triples in einer Datenbank hinterlegen

Falcons

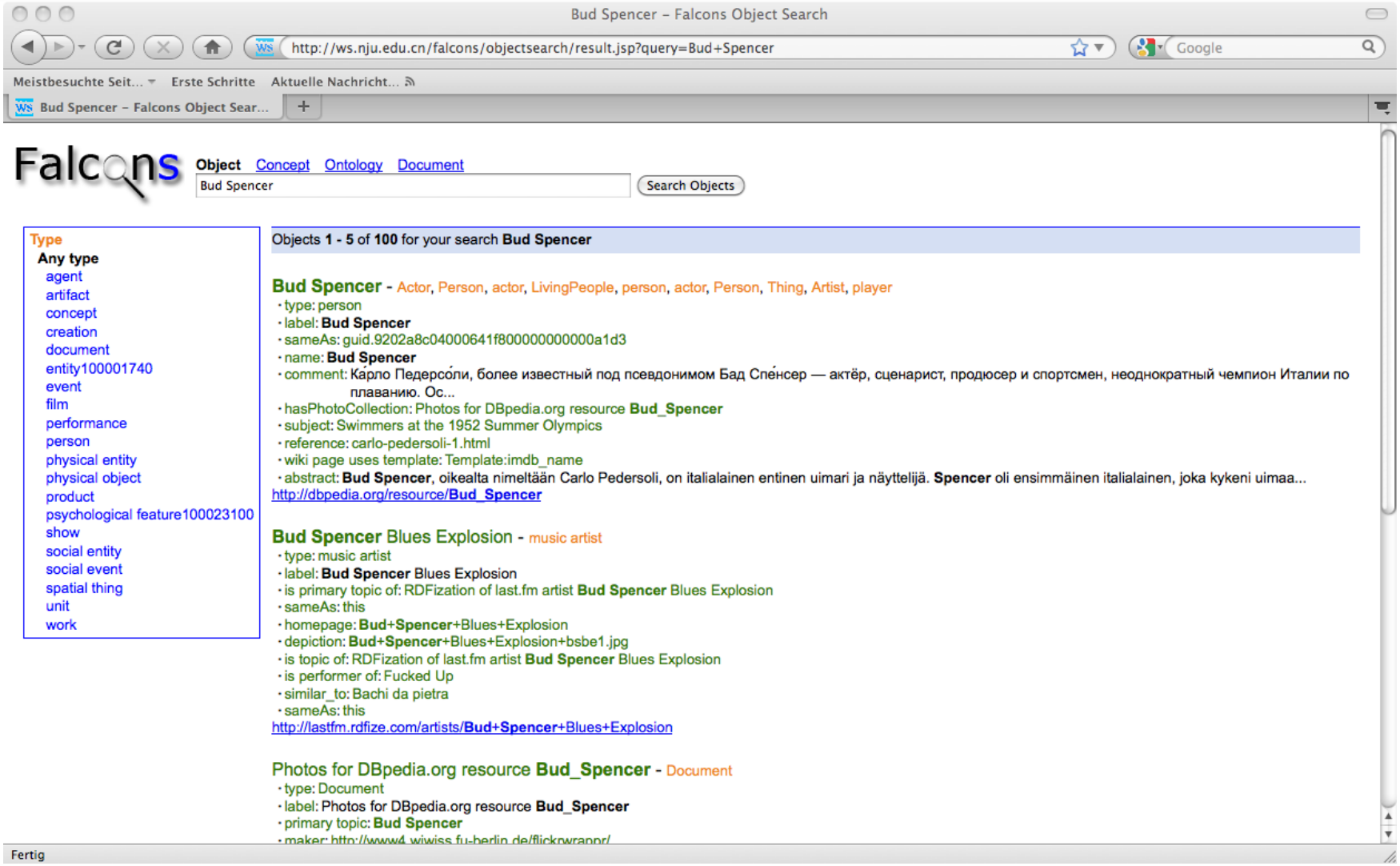
Anfragebearbeitung

- Keywords
- Filterung durch Klassen/Typen
- Ranking basierend auf Ähnlichkeit zur Keyword-Anfragen und Popularität
- Output: Entitäten mit Textausschnitten (Snippets)

Suchmaske

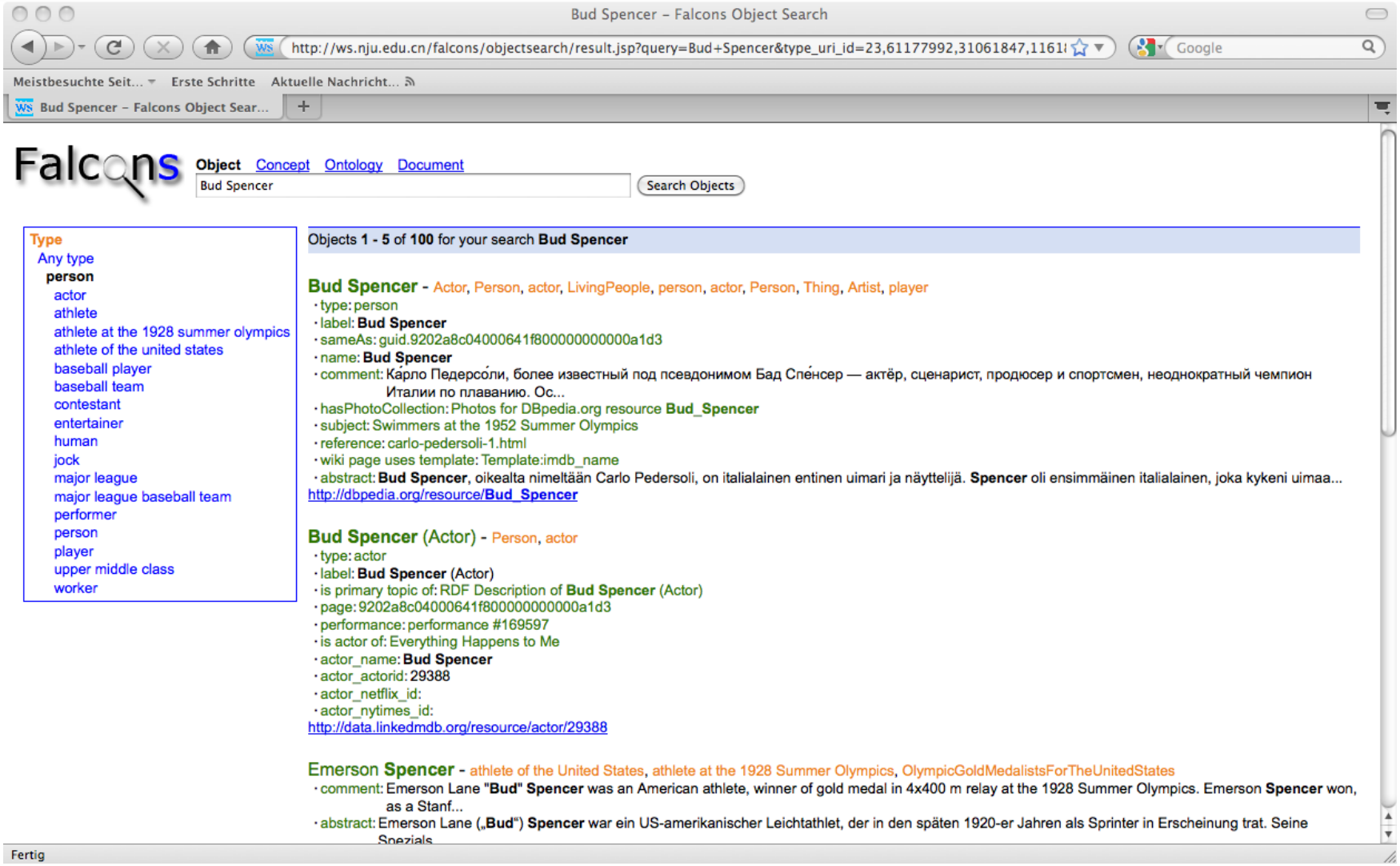
http://ws.nju.edu.cn/falcons/objectsearch/index.jsp

Suchmaske



http://ws.nju.edu.cn/falcons/objectsearch/index.jsp

Suchmaske



http://ws.nju.edu.cn/falcons/objectsearch/index.jsp

Sindice

Fokus: Dokumente und Quellen

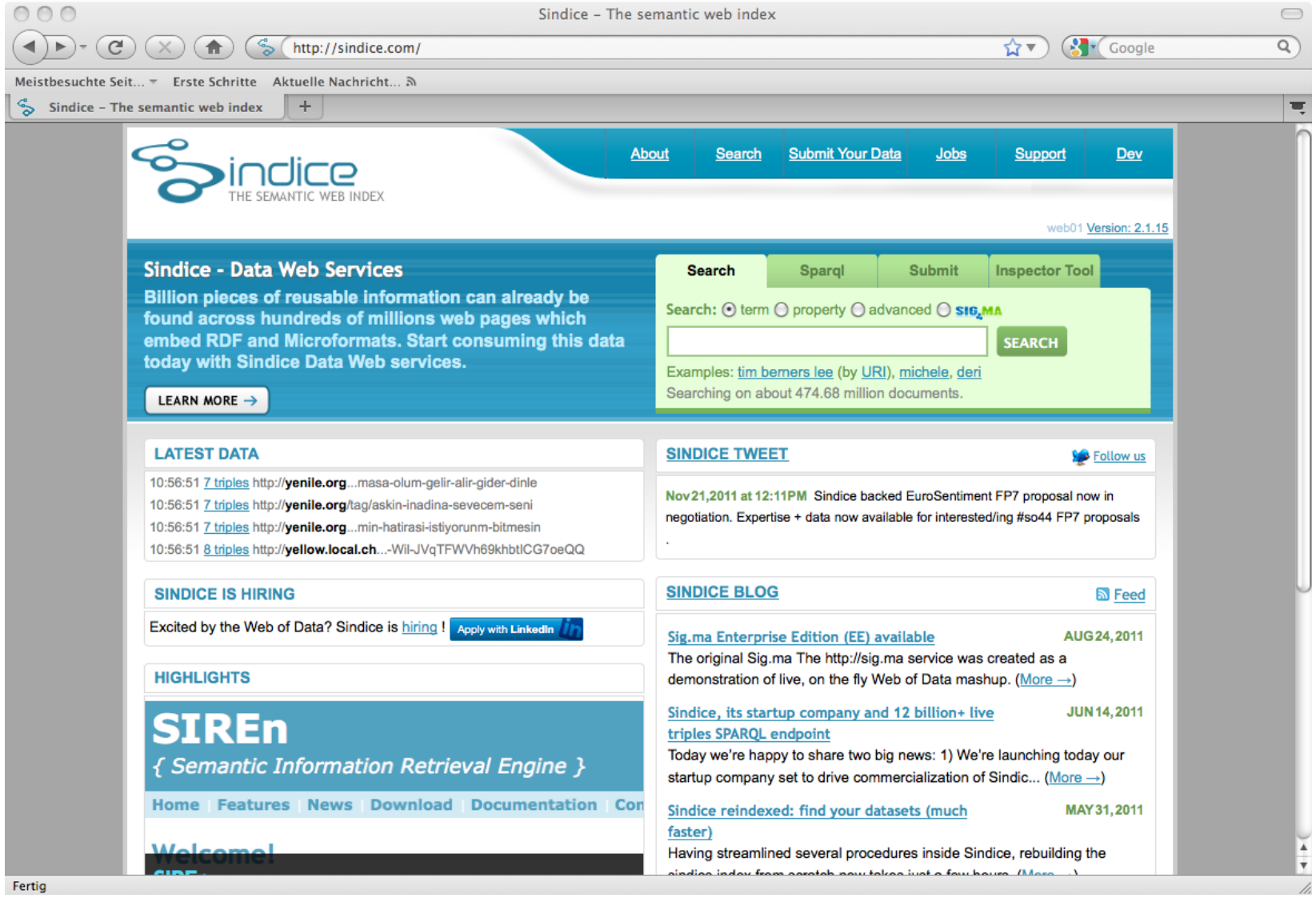
Indexe

- URI (Unified Resource Identifier)
- IFP (Inverse Functional Properties)
- Literal

Features

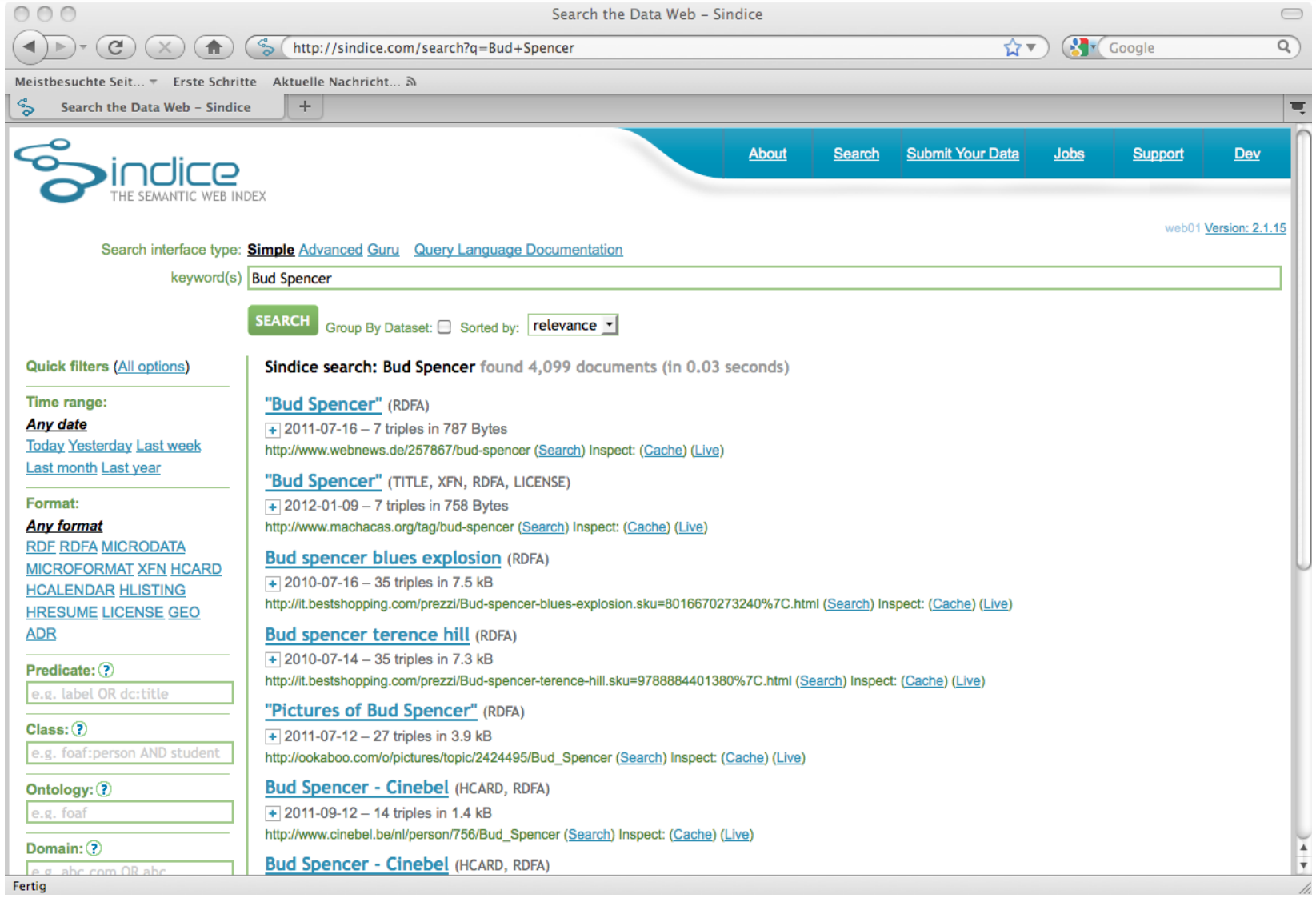
- Reasoning
- SPARQL endpoint indexing

Suchmaske



http://sindice.com

Suchmaske



http://sindice.com

Sig.ma

Fokus: Browser und Mashup-Generator

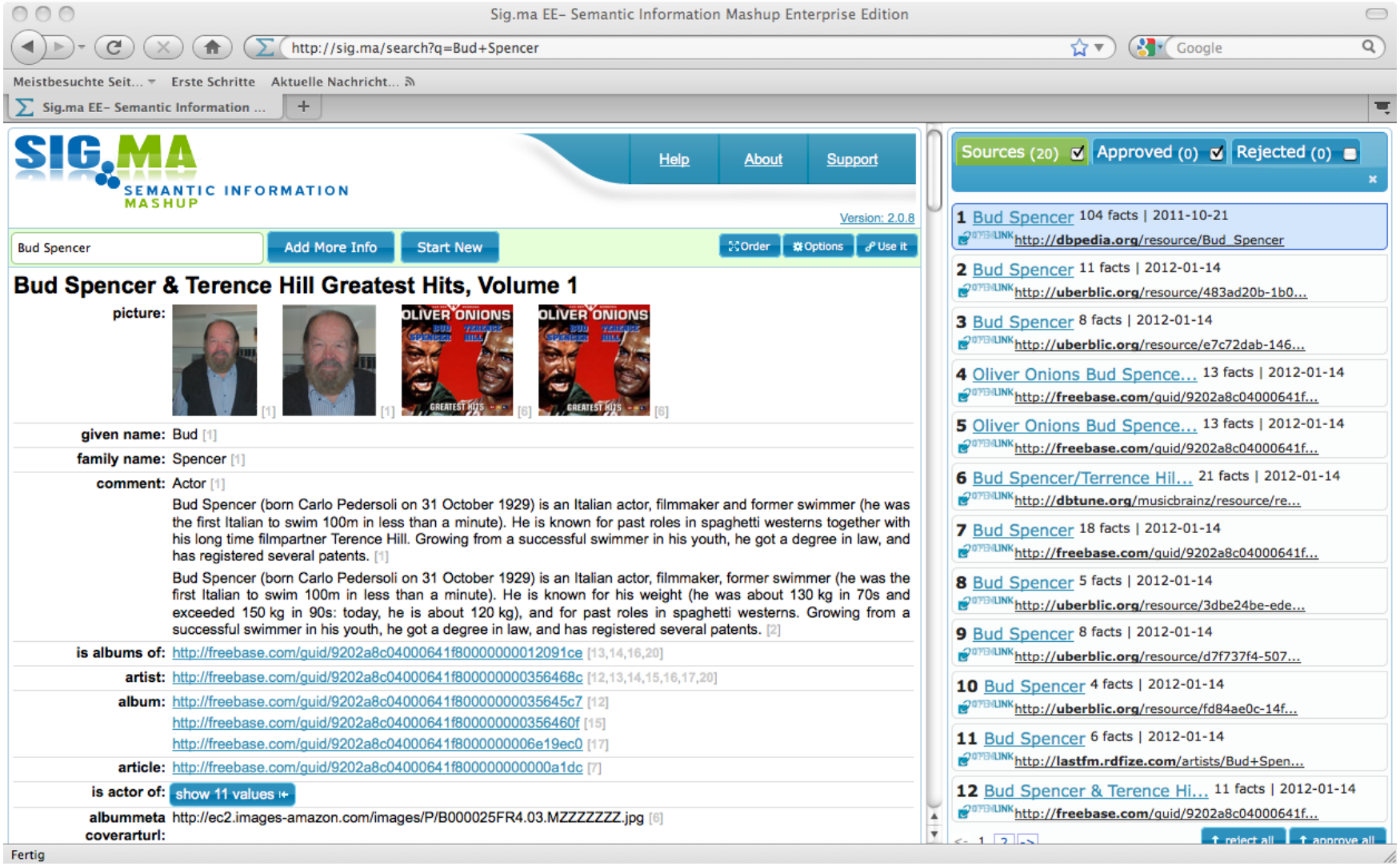
Funktionsweise

- Quellen finden
- Daten extrahieren
- Gruppieren und ausgeben

Sig.ma

http://sig.ma

Sig.ma



http://sig.ma

Indexe

Verschiedene Arten

- „Vollständige“ Indexierung
  - Triple liegen vollständig vor
  - Effizienter Zugriff auf lokale Daten
- Zusammenfassende Indexe
  - Auswahl von relevanten Quellen
  - Kompromiss: Detaillierungsgrad vs. Speicherplatz

Zusammenfassende Indexe

Indexe auf Schemaebene

- Properties (URIs die als Prädikate benutzt werden)
- Classes (Objekte in rdf:type Triples)

Beispiel

1. `http://data.linkedmdb.org/resource/actor/29388 owl:sameAs http://rdf.freebase.com/ns/m/018gm`
2. `http://data.linkedmdb.org/resource/actor/29388 rdf:type http://data.linkedmdb.org/resource/movie/actor`
3. `http://rdf.freebase.com/ns/m/018gm fb:place_of_birth http://rdf.freebase.com/ns/en/naples`

Search Key	Sources
owl:sameAs	imdb
rdf:type	imdb
`http://data.linkedmdb.org/resource/movie/actor`	imdb
`fb:place_of_birth`	freebase

Zusammenfassende Indexe

Indexe auf Schemaebene

- Properties (URIs die als Prädikate benutzt werden)
- Classes (Objekte in rdf:type Triples)

Beispiel

1. `http://data.linkedmdb.org/resource/actor/29388 owl:sameAs http://rdf.freebase.com/ns/m/018gm`
2. `http://data.linkedmdb.org/resource/actor/29388 rdf:type http://data.linkedmdb.org/resource/movie/actor`
3. `http://rdf.freebase.com/ns/m/018gm fb:place_of_birth http://rdf.freebase.com/ns/en/naples`

Search Key	Sources
owl:sameAs	imdb
rdf:type	imdb
`http://data.linkedmdb.org/resource/movie/actor`	imdb
`fb:place_of_birth`	freebase

Zusammenfassende Indexe

Invertierte URI Indexe

- Alle URIs, die in einer Quelle vorkommen

Beispiel

1. `http://data.linkedmdb.org/resource/actor/29388 owl:sameAs http://rdf.freebase.com/ns/m/018gm`
2. `http://data.linkedmdb.org/resource/actor/29388 rdf:type http://data.linkedmdb.org/resource/movie/actor`
3. `http://rdf.freebase.com/ns/m/018gm fb:place_of_birth http://rdf.freebase.com/ns/en/naples`

Search Key	Sources
http://data.linkedmdb.org/resource/actor/29388	imdb
owl:sameAs	imdb
http://rdf.freebase.com/ns/m/018gm	imdb, freebase
rdf:type	imdb
http://data.linkedmdb.org/resource/movie/actor	imdb
fb:place_of_birth	freebase
http://rdf.freebase.com/ns/en/naples	freebase

Zusammenfassende Indexe

Invertierte URI Indexe

- Alle URIs, die in einer Quelle vorkommen

Beispiel

1. `http://data.linkedmdb.org/resource/actor/29388 owl:sameAs http://rdf.freebase.com/ns/m/018gm`
2. `http://data.linkedmdb.org/resource/actor/29388 rdf:type http://data.linkedmdb.org/resource/movie/actor`
3. `http://rdf.freebase.com/ns/m/018gm fb:place_of_birth http://rdf.freebase.com/ns/en/naples`

Search Key	Sources
http://data.linkedmdb.org/resource/actor/29388	imdb
owl:sameAs	imdb
http://rdf.freebase.com/ns/m/018gm	imdb, freebase
rdf:type	imdb
http://data.linkedmdb.org/resource/movie/actor	imdb
fb:place_of_birth	freebase
http://rdf.freebase.com/ns/en/naples	freebase

Suchmaschinen sind nicht genug...

Anfrage

- In welchen Filmen haben Carlo Pedersoli (Bud Spencer) und Mario Girotti (Terence Hill) zusammen gespielt?
- Oder in welchen Filmen hat nur einer von beiden (ohne den jeweils anderen) gespielt?

Suchmaschinen sind nicht genug...

Anfrage

- In welchen Filmen haben Carlo Pedersoli (Bud Spencer) und Mario Girotti (Terence Hill) zusammen gespielt?
- Oder in welchen Filmen hat nur einer von beiden (ohne den jeweils anderen) gespielt?

Anfragemöglichkeiten

- „Carlo Pedersoli Mario Girotti“
- „Bud Spencer Terence Hill“
- „Bud Spencer Mario Girotti“
- `http://data.linkedmdb.org/resource/actor/29388`
  `http://data.linkedmdb.org/resource/actor/29666`
- ...

Suchmaschinen sind nicht genug...

Anfrage

- In welchen Filmen haben Carlo Pedersoli (Bud Spencer) und Mario Girotti (Terence Hill) zusammen gespielt?
- Oder in welchen Filmen hat nur einer von beiden (ohne den jeweils anderen) gespielt?

Erkenntnis

Keyword-Anfragen und Suchmaschinen reichen hier einfach nicht mehr aus!

Explorative Anfragebearbeitung

Eigenschaften

- Dereferenzieren von URIs
- Keine Statistiken benötigt
- Daten werden während der Anfragebearbeitung heruntergeladen (Live Lookups)

Basisalgorithmus

- Abwechselnd
  - Teile der Anfrage evaluieren
  - Dereferenziere URIs im Zwischenergebnis (Daten herunterladen)
  - Benutze heruntergeladene Daten zur weiteren Bearbeitung der Anfrage

Explorative Anfragebearbeitung

Eigenschaften

- Dereferenzieren von URIs
- Keine Statistiken benötigt
- Daten werden während der Anfragebearbeitung heruntergeladen (Live Lookups)

Basisalgorithmus

- Abwechselnd
  - Teile der Anfrage evaluieren
  - Dereferenziere URIs im Zwischenergebnis (Daten herunterladen)
  - Benutze heruntergeladene Daten zur weiteren Bearbeitung der Anfrage

Explorative Anfragebearbeitung

Join-Berechnung durch Nested-Loop (Pull/Iterator-basierte Joins)

Beispielanfrage

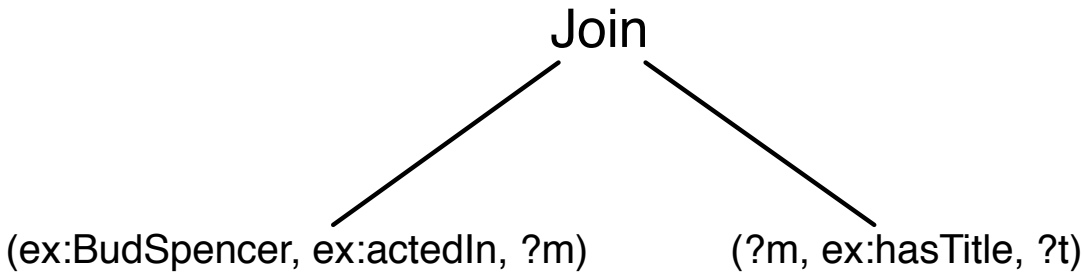
```
SELECT ?m ?t WHERE {
  ex:BudSpencer ex:actedIn ?m .
  ?m ex:hasTitle ?t .
}
```

Explorative Anfragebearbeitung

Join-Berechnung durch Nested-Loop (Pull/Iterator-basierte Joins)

Beispielanfrage

```
SELECT ?m ?t WHERE {
  ex:BudSpencer ex:actedIn ?m .
  ?m ex:hasTitle ?t .
}
```

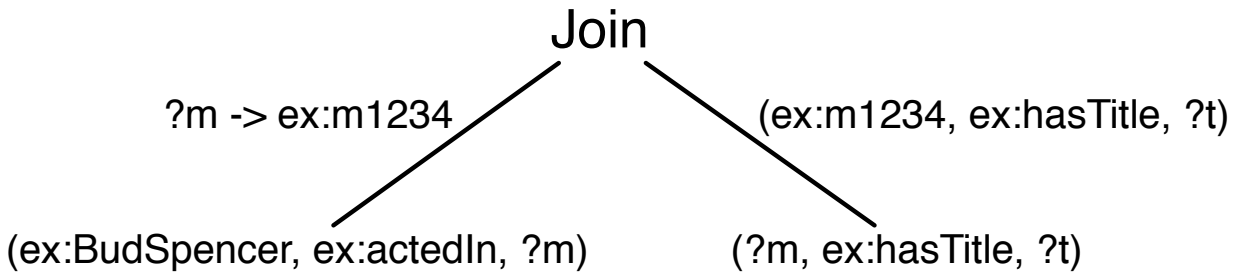


Explorative Anfragebearbeitung

Join-Berechnung durch Nested-Loop (Pull/Iterator-basierte Joins)

Beispielanfrage

```
SELECT ?m ?t WHERE {
  ex:BudSpencer ex:actedIn ?m .
  ?m ex:hasTitle ?t .
}
```

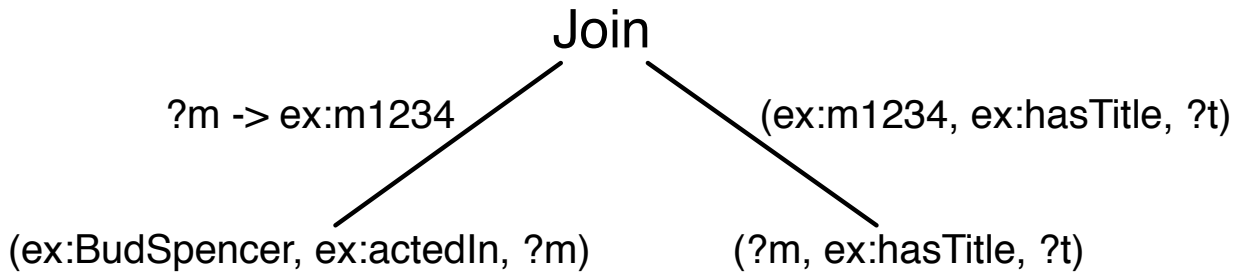


Explorative Anfragebearbeitung

Join-Berechnung durch Nested-Loop (Pull/Iterator-basierte Joins)

Beispielanfrage

```
SELECT ?m ?t WHERE {
  ex:BudSpencer ex:actedIn ?m .
  ?m ex:hasTitle ?t .
}
```



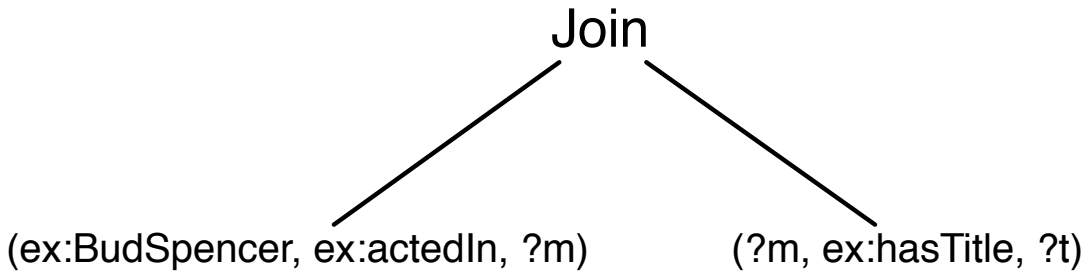
Problem, wenn die Daten für "ex:m1234" noch nicht heruntergeladen wurden

- Join-Operator wartet bis die Daten verfügbar sind (Blocking)
- Non-Blocking Iterator Join [HBF09]
  - Vorläufiges Zurückweisen von Join-Input der unteren Ebenen
  - Neuer Versuch später
- Reihenfolge der Dereferenzierung beeinflusst Vollständigkeit [Har11]

Explorative Anfragebearbeitung

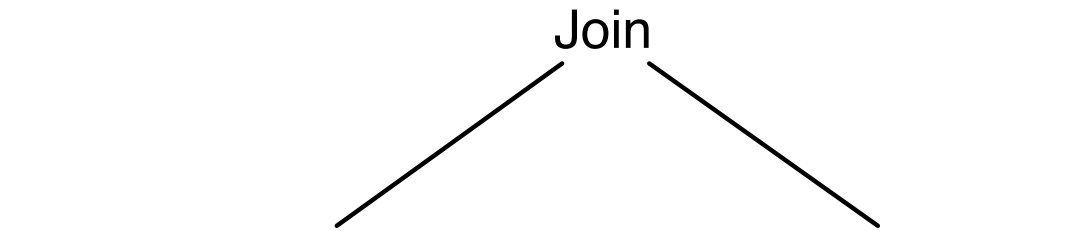
Symmetrischer Hash Join über Linked Data [LT10]

- Push-basiert statt pull-basiert
- Vermeidung von Wartezeiten
- Effizientere Laufzeit



Explorative Anfragebearbeitung

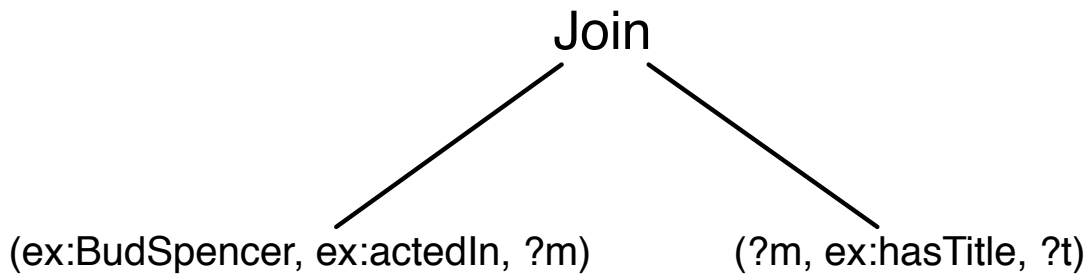
Symmetrischer Hash Join über Linked Data [LT10]



hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)

Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]

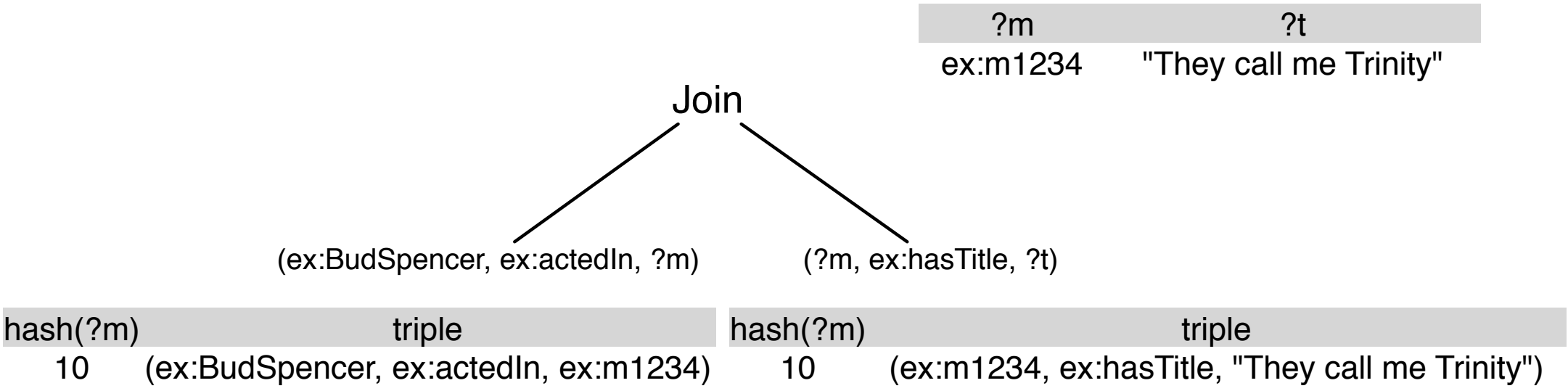


hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")

Explorative Anfragebearbeitung

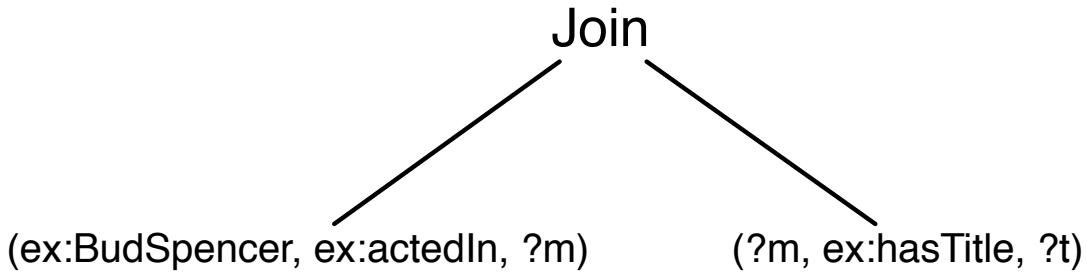
Symmetrischer Hash Join über Linked Data [LT10]



Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]

?m	?t
ex:m1234	"They call me Trinity"



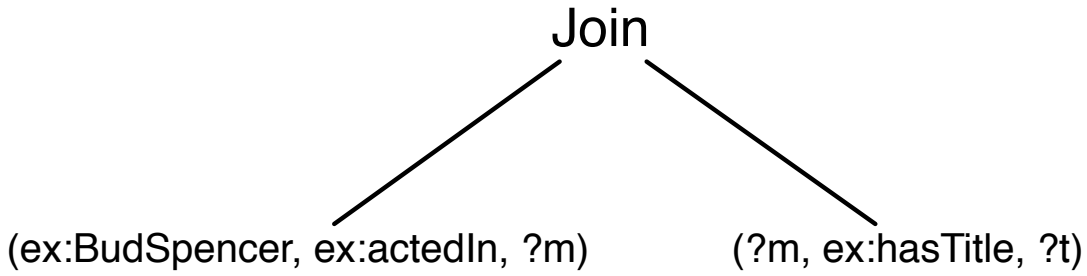
hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)
23	(ex:BudSpencer, ex:actedIn, ex:m5678)

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")

Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]

?m	?t
ex:m1234	"They call me Trinity"

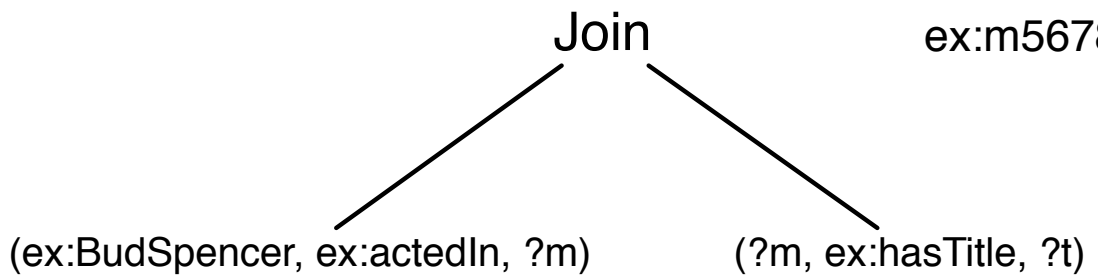


hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)
23	(ex:BudSpencer, ex:actedIn, ex:m5678)

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")
23	(ex:m5678, ex:hasTitle, "Troublemakes")

Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]



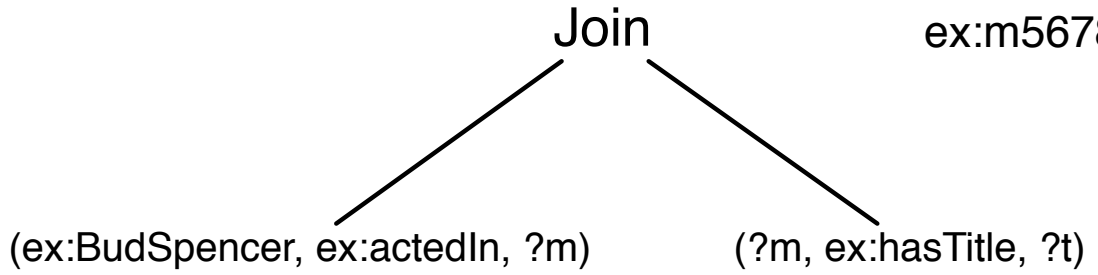
?m	?t
ex:m1234	"They call me Trinity"
ex:m5678	"Troublemakers"

hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)
23	(ex:BudSpencer, ex:actedIn, ex:m5678)

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")
23	(ex:m5678, ex:hasTitle, "Troublemakes")

Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]



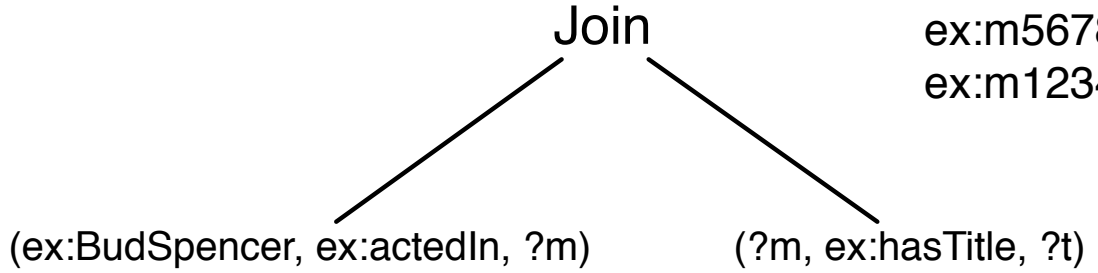
?m	?t
ex:m1234	"They call me Trinity"
ex:m5678	"Troublemakers"

hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)
23	(ex:BudSpencer, ex:actedIn, ex:m5678)

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")
	(ex:m1234, ex:hasTitle, "Lo chiamavano Trinità")
23	(ex:m5678, ex:hasTitle, "Troublemakes")

Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]



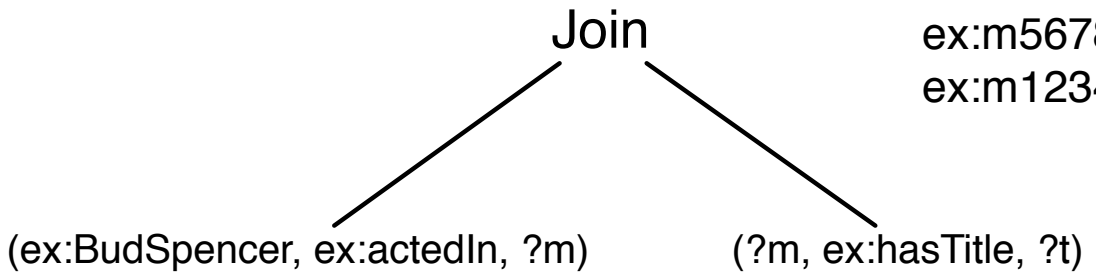
?m	?t
ex:m1234	"They call me Trinity"
ex:m5678	"Troublemakers"
ex:m1234	"Lo chiamavano Trinità"

hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)
23	(ex:BudSpencer, ex:actedIn, ex:m5678)
...	...

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")
	(ex:m1234, ex:hasTitle, "Lo chiamavano Trinità")
23	(ex:m5678, ex:hasTitle, "Troublemakes")
...	...

Explorative Anfragebearbeitung

Symmetrischer Hash Join über Linked Data [LT10]



?m	?t
ex:m1234	"They call me Trinity"
ex:m5678	"Troublemakers"
ex:m1234	"Lo chiamavano Trinità"

hash(?m)	triple
10	(ex:BudSpencer, ex:actedIn, ex:m1234)
23	(ex:BudSpencer, ex:actedIn, ex:m5678)
...	...

hash(?m)	triple
10	(ex:m1234, ex:hasTitle, "They call me Trinity")
	(ex:m1234, ex:hasTitle, "Lo chiamavano Trinità")
23	(ex:m5678, ex:hasTitle, "Troublemakes")
...	...

Erweiterung für die Kombination mit lokalen Daten [LT11]

- Einige Daten liegen lokal vor
- Indexe über lokale Daten zur Effizienzsteigerung

Index-basierte Anfragebearbeitung

Eigenschaften

- Potentielle Quellen im Voraus indexieren (Crawling, Dereferencing URIs, etc.)
- Relevante Quellen für eine Anfrage identifizieren
- Daten der Quellen parallel herunterladen

Erweiterter Index [UHK+11]

- Indiziere (S,P,O)-Kombination
- Basis: multidimensionale Histogramme
- Identifikation relevanter Quellen für Triple-Patterns
- Identifikation relevanter Quellen für Joins

Index-basierte Anfragebearbeitung

Eigenschaften

- Potentielle Quellen im Voraus indexieren (Crawling, Dereferencing URIs, etc.)
- Relevante Quellen für eine Anfrage identifizieren
- Daten der Quellen parallel herunterladen

Erweiterter Index [UHK+11]

- Indiziere (S,P,O)-Kombination
- Basis: multidimensionale Histogramme
- Identifikation relevanter Quellen für Triple-Patterns
- Identifikation relevanter Quellen für Joins

Index-basierte Anfragebearbeitung

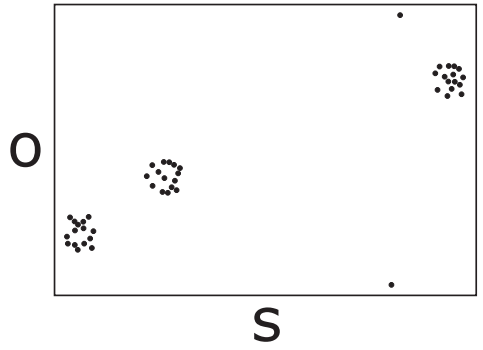
Histogramme

- Transformiere Triple in numerischen Raum (Hash-Funktionen)
  (`ex:BudSpencer ex:actedIn ex:m1234`) $\rightarrow$ (323, 232, 124)
- In entsprechendes Bucket einfügen
- Lookup: Transformiere Triple-Pattern in numerischen Raum
  (`ex:BudSpencer ex:actedIn ?m`)

Index-basierte Anfragebearbeitung

Histogramme

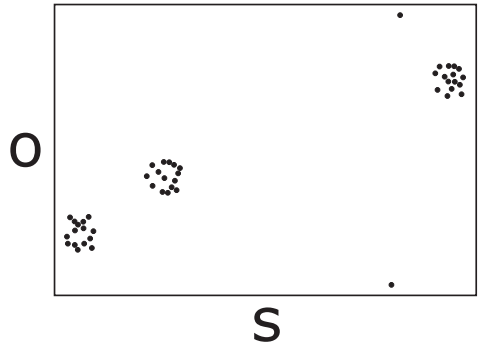
- Transformiere Triple in numerischen Raum (Hash-Funktionen)
  (`ex:BudSpencer ex:actedIn ex:m1234`) $\rightarrow$ (323, 232, 124)
- In entsprechendes Bucket einfügen
- Lookup: Transformiere Triple-Pattern in numerischen Raum
  (`ex:BudSpencer ex:actedIn ?m`)



Index-basierte Anfragebearbeitung

Histogramme

- Transformiere Triple in numerischen Raum (Hash-Funktionen)
  (`ex:BudSpencer ex:actedIn ex:m1234`) $\rightarrow$ (323, 232, 124)
- In entsprechendes Bucket einfügen
- Lookup: Transformiere Triple-Pattern in numerischen Raum
  (`ex:BudSpencer ex:actedIn ?m`)



o

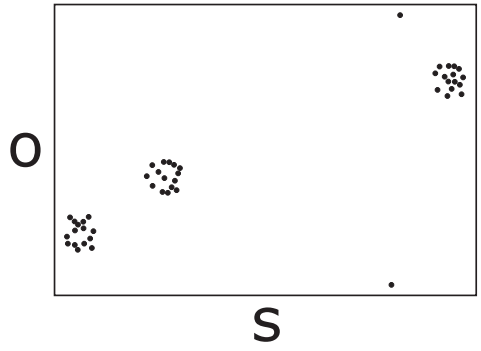
0	0	16
15	0	0
15	0	1

s

Index-basierte Anfragebearbeitung

Histogramme

- Transformiere Triple in numerischen Raum (Hash-Funktionen)
  (`ex:BudSpencer ex:actedIn ex:m1234`) $\rightarrow$ (323, 232, 124)
- In entsprechendes Bucket einfügen
- Lookup: Transformiere Triple-Pattern in numerischen Raum
  (`ex:BudSpencer ex:actedIn ?m`)



o \ s	0	0	16
15	0	0	
15	0	1	

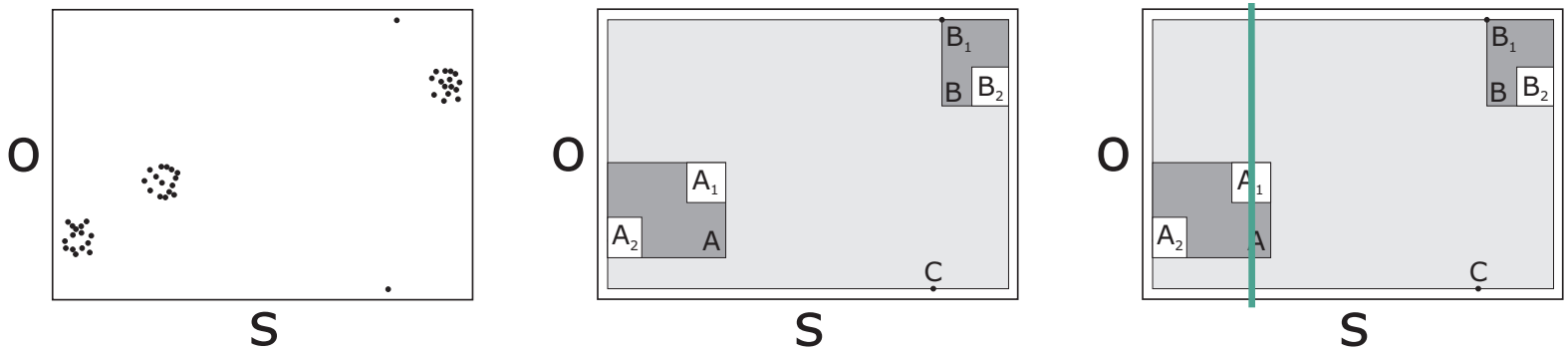
o \ s	0	0	16
15	0	0	
15	0	1	

Index-basierte Anfragebearbeitung

Histogramme

- Transformiere Triple in numerischen Raum (Hash-Funktionen)
  (`ex:BudSpencer ex:actedIn ex:m1234`) $\rightarrow$ (323, 232, 124)
- In entsprechendes Bucket einfügen
- Lookup: Transformiere Triple-Pattern in numerischen Raum
  (`ex:BudSpencer ex:actedIn ?m`)

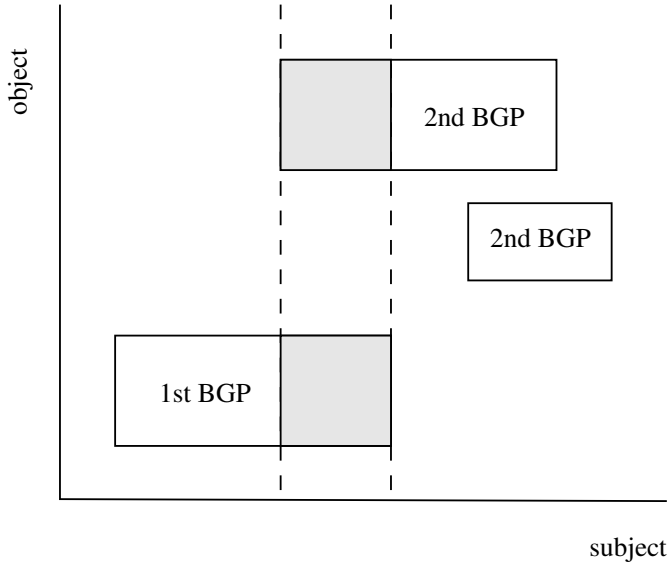
Alternative zu Histogrammen (QTree oder Clustering) [UHK+11]:



Index-basierte Anfragebearbeitung

Kardinalitätsabschätzung für Joins

- Buckets für 1. Triple-Pattern ermitteln
- Buckets für 2. Triple-Pattern ermitteln
- Buckets bestimmen, die in der Join-Dimension überlappen (z.B. subject)
- Schätze Ergebniskardinalität durch Überlappungsgrad



Lokale Anfragebearbeitung

Pro und Kontra

- Live-Lookups garantieren Aktualität
- Sehr viel Network-Traffic
- Alles wird von einer zentralen Instanz (Koordinator) gemacht

Warum nicht die Rechenleistung der SPARQL-Endpoints nutzen?

Verteilte Anfragebearbeitung

Lokale Anfragebearbeitung

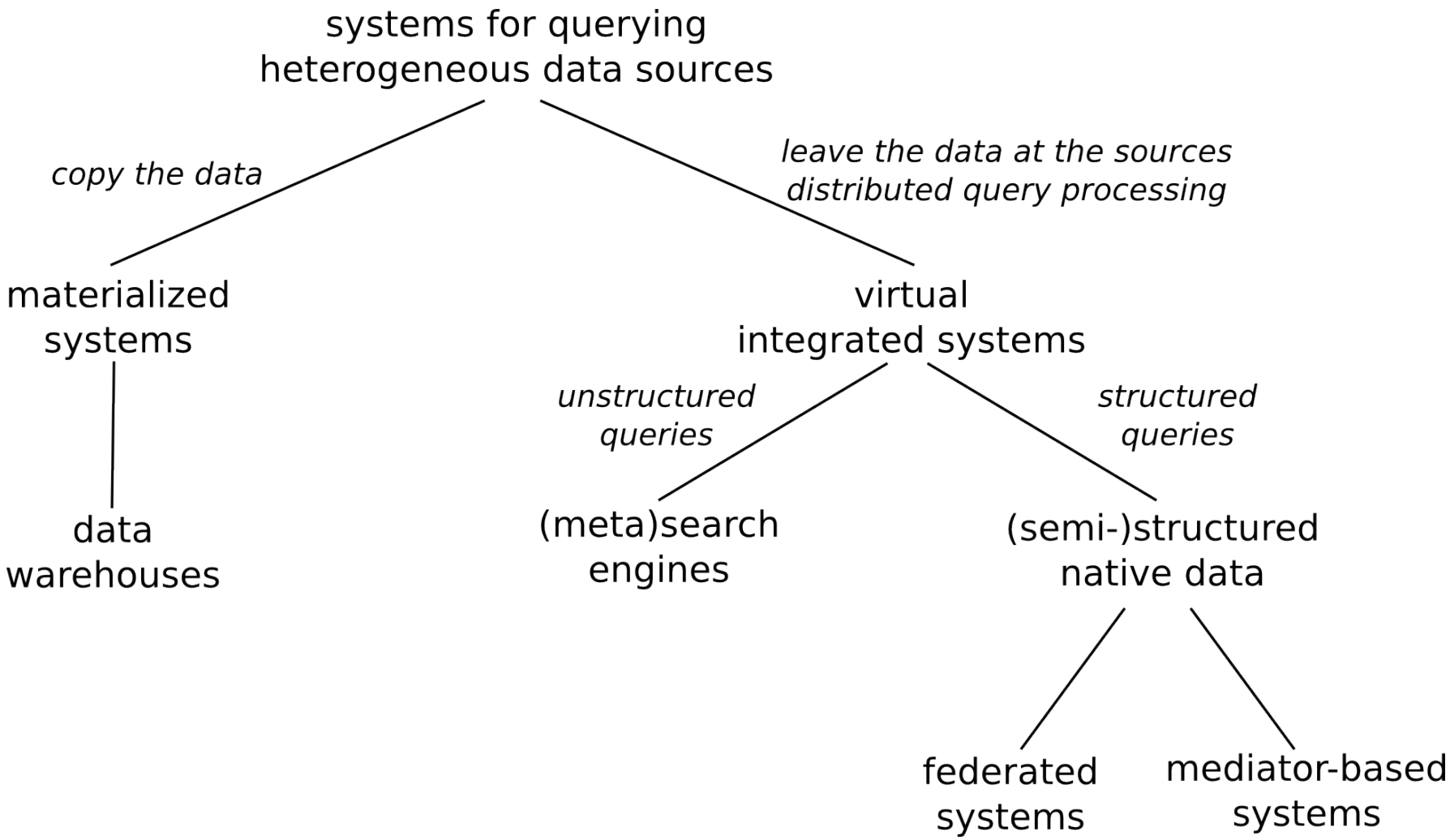
Pro und Kontra

- Live-Lookups garantieren Aktualität
- Sehr viel Network-Traffic
- Alles wird von einer zentralen Instanz (Koordinator) gemacht

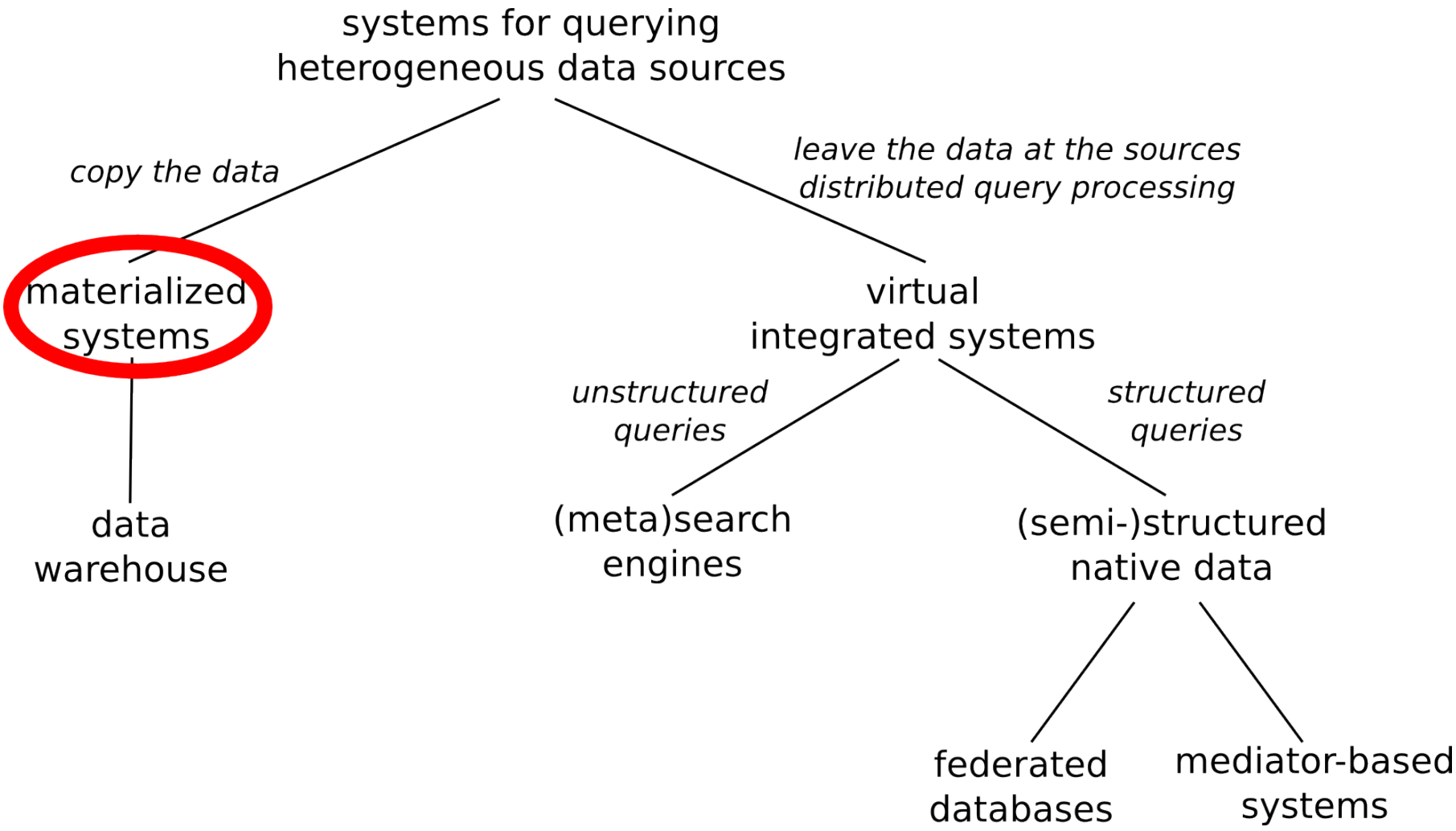
Warum nicht die Rechenleistung der SPARQL-Endpoints nutzen?

Verteilte Anfragebearbeitung

Datenintegrationssysteme [DD99]



Datenintegrationssysteme [DD99]



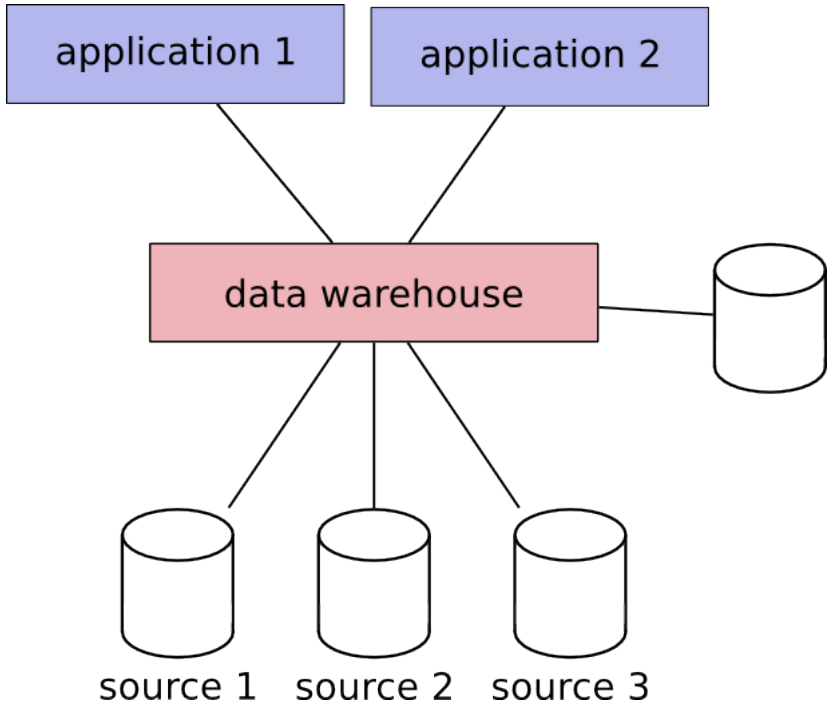
Data Warehousing

Datenfluss

- ETL-Prozess
  - Extraktion
  - Transformation
  - Laden
- Redundante Datenhaltung
- Periodischer Datenimport
- Aktualität

Anfragebearbeitung

- Zentrale Anfragebearbeitung mit lokaler Optimierung



Funktioniert für RDF und Linked Data im Prinzip genauso

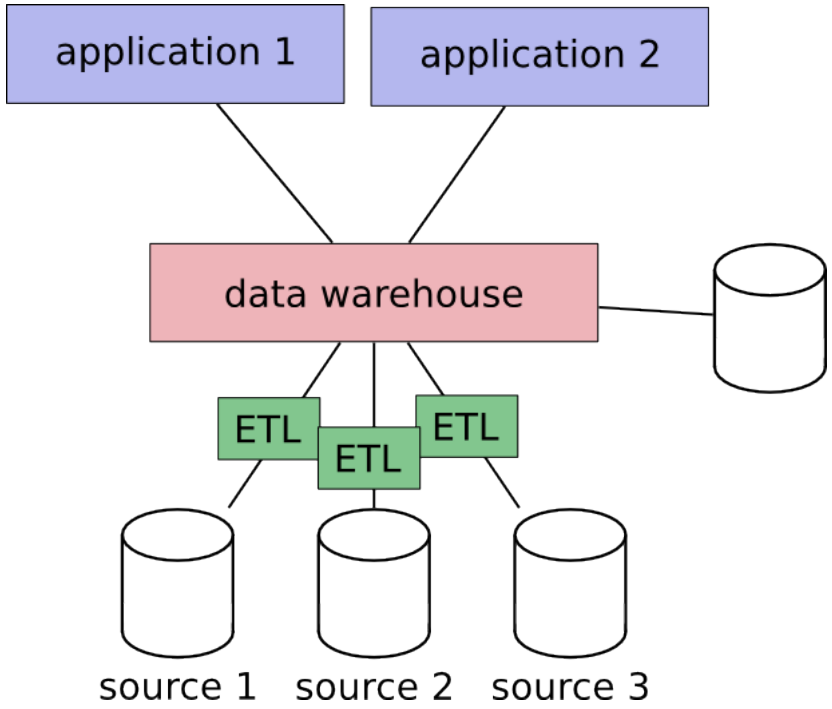
Data Warehousing

Datenfluss

- ETL-Prozess
  - Extraktion
  - Transformation
  - Laden
- Redundante Datenhaltung
- Periodischer Datenimport
- Aktualität

Anfragebearbeitung

- Zentrale Anfragebearbeitung mit lokaler Optimierung



Funktioniert für RDF und Linked Data im Prinzip genauso

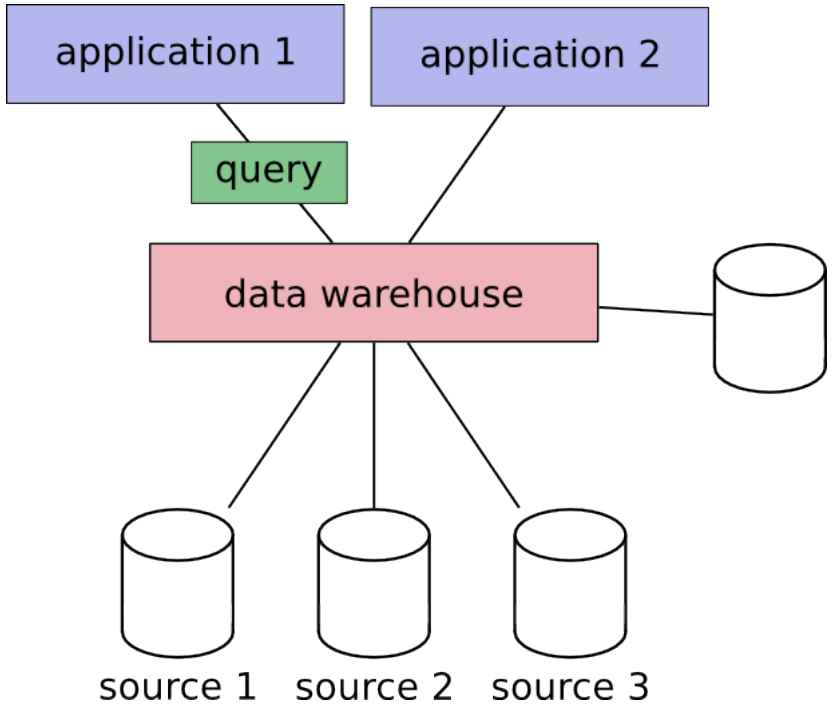
Data Warehousing

Datenfluss

- ETL-Prozess
  - Extraktion
  - Transformation
  - Laden
- Redundante Datenhaltung
- Periodischer Datenimport
- Aktualität

Anfragebearbeitung

- Zentrale Anfragebearbeitung mit lokaler Optimierung



Funktioniert für RDF und Linked Data im Prinzip genauso

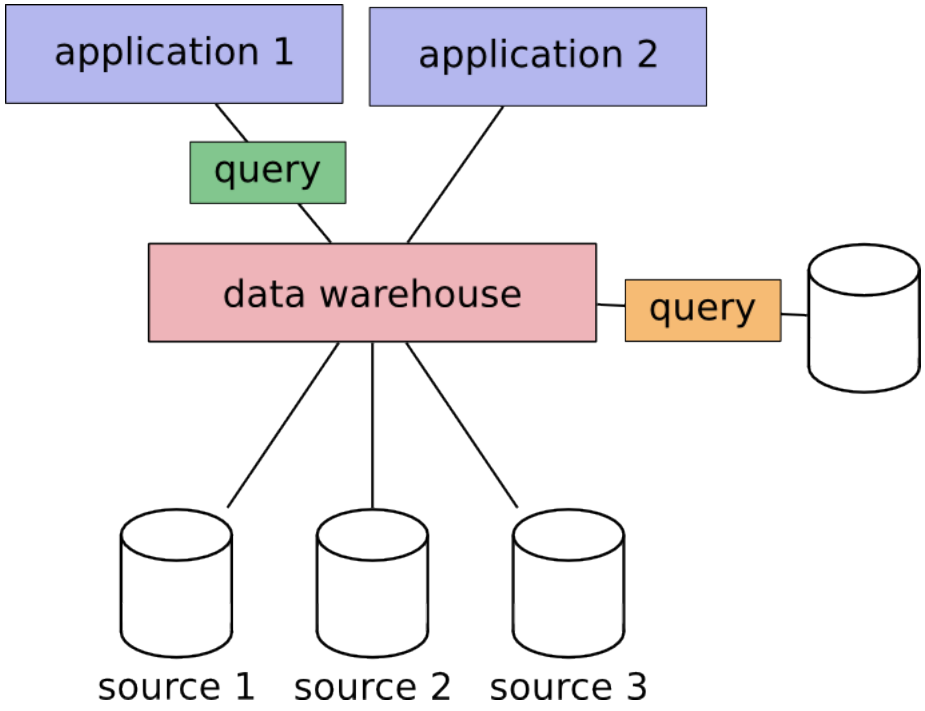
Data Warehousing

Datenfluss

- ETL-Prozess
  - Extraktion
  - Transformation
  - Laden
- Redundante Datenhaltung
- Periodischer Datenimport
- Aktualität

Anfragebearbeitung

- Zentrale Anfragebearbeitung mit lokaler Optimierung



Funktioniert für RDF und Linked Data im Prinzip genauso

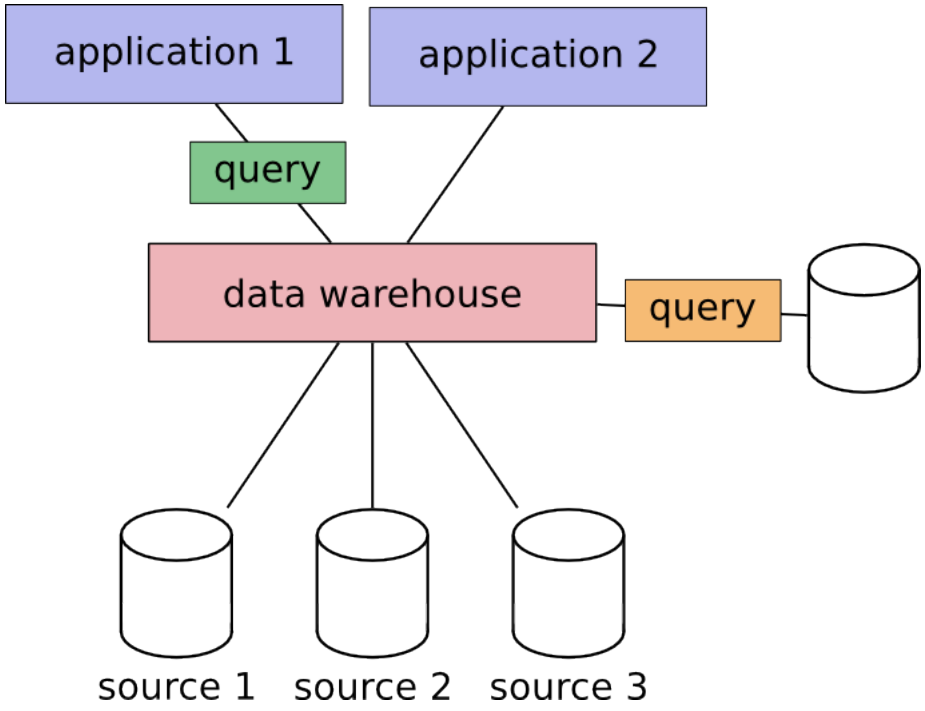
Data Warehousing

Datenfluss

- ETL-Prozess
  - Extraktion
  - Transformation
  - Laden
- Redundante Datenhaltung
- Periodischer Datenimport
- Aktualität

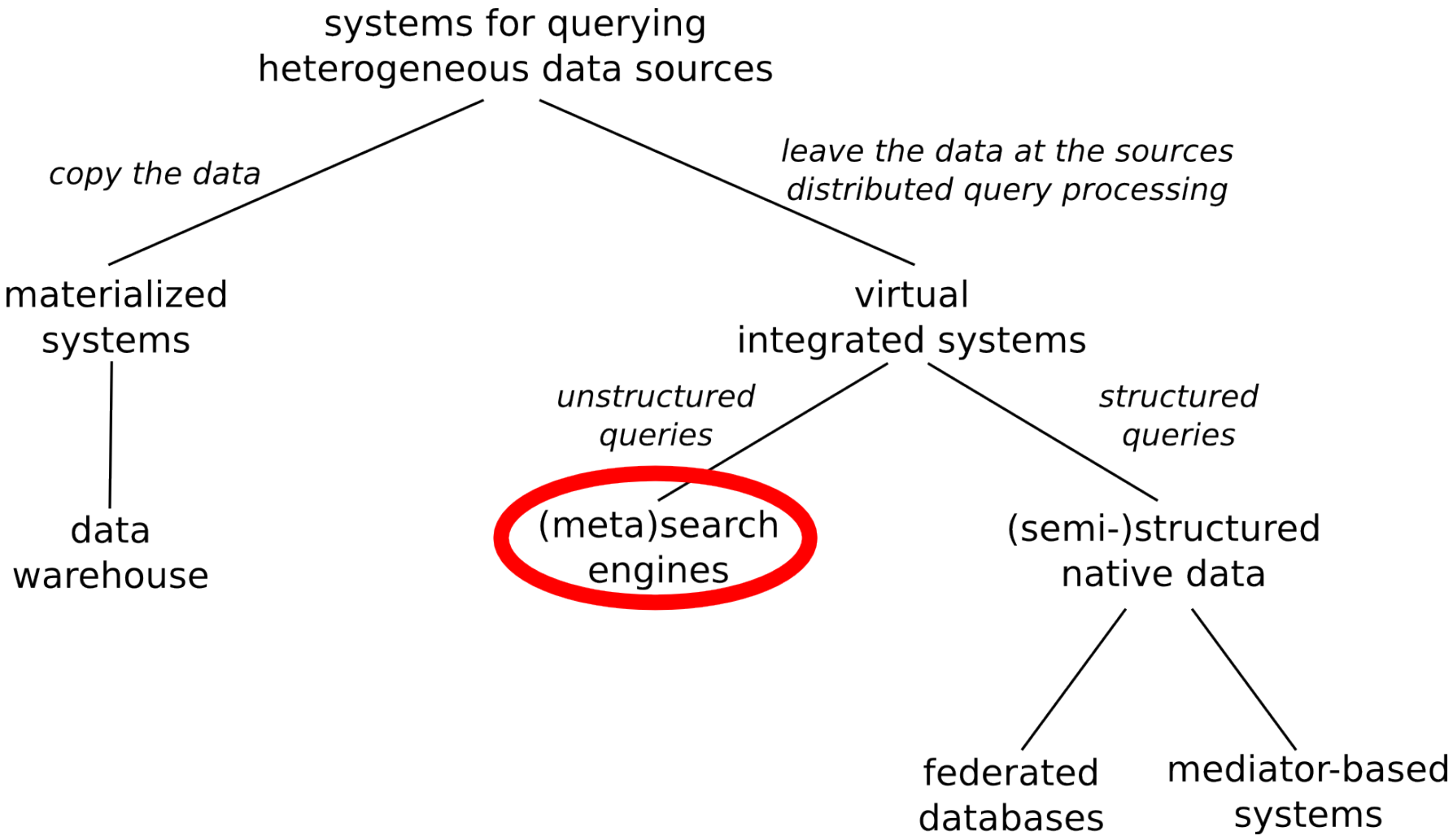
Anfragebearbeitung

- Zentrale Anfragebearbeitung mit lokaler Optimierung

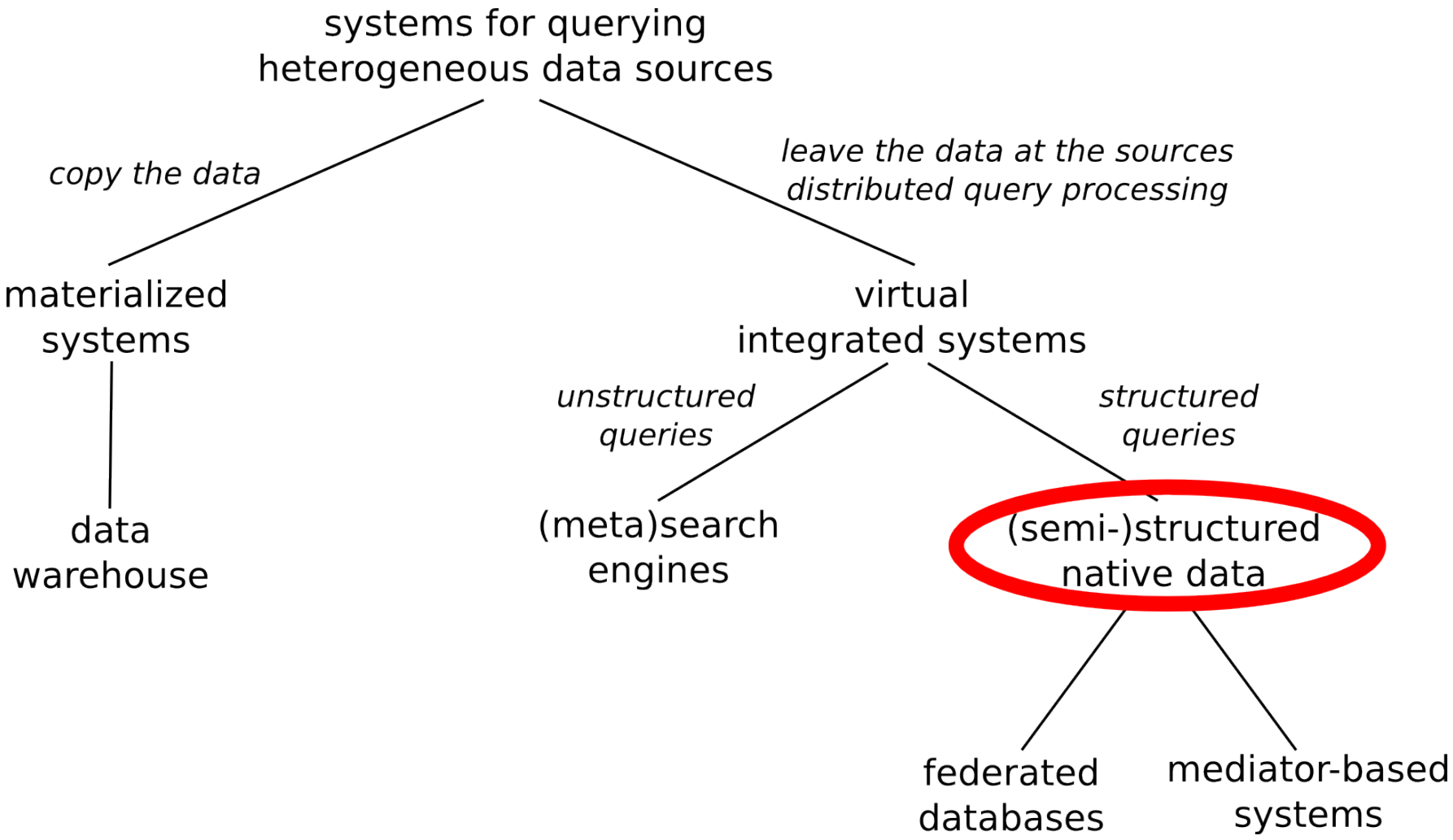


Funktioniert für RDF und Linked Data im Prinzip genauso

Datenintegrationssysteme [DD99]



Datenintegrationssysteme [DD99]



Föderierte und Mediator-Wrapper-Systeme

Gemeinsamkeiten

- Zusammenschluss eigenständiger Datenquellen
- Ein globales Interface

Föderierte Systeme

- Aktive Unterstützung und Kooperation
- Datenquellen überwinden Heterogenität (Datenmodell, Anfragesprache, etc.)

Mediator-Wrapper-Systeme

- Mediator hat Statistiken
- Wrapper überwindet Heterogenität
- Keine zusätzliche Kooperation der Quellen (Autonomie)

Föderierte und Mediator-Wrapper-Systeme

Gemeinsamkeiten

- Zusammenschluss eigenständiger Datenquellen
- Ein globales Interface

Föderierte Systeme

- Aktive Unterstützung und Kooperation
- Datenquellen überwinden Heterogenität (Datenmodell, Anfragesprache, etc.)

Mediator-Wrapper-Systeme

- Mediator hat Statistiken
- Wrapper überwindet Heterogenität
- Keine zusätzliche Kooperation der Quellen (Autonomie)

Föderierte und Mediator-Wrapper-Systeme

Terminologie in der Semantic-Web-Community

- Virtual integration
- Federated systems
- Federator
- Mediator/wrapper
- Federated query processing
- ...

Föderierte und Mediator-Wrapper-Systeme

Terminologie in der Semantic-Web-Community

- Virtual integration
- Federated systems
- Federator
- Mediator/wrapper
- Federated query processing
- ...

Föderierte und Mediator-Wrapper-Systeme

Terminologie in der Semantic-Web-Community

- Virtual integration
- Federated systems
- Federator
- Mediator/wrapper
- Federated query processing
- ...

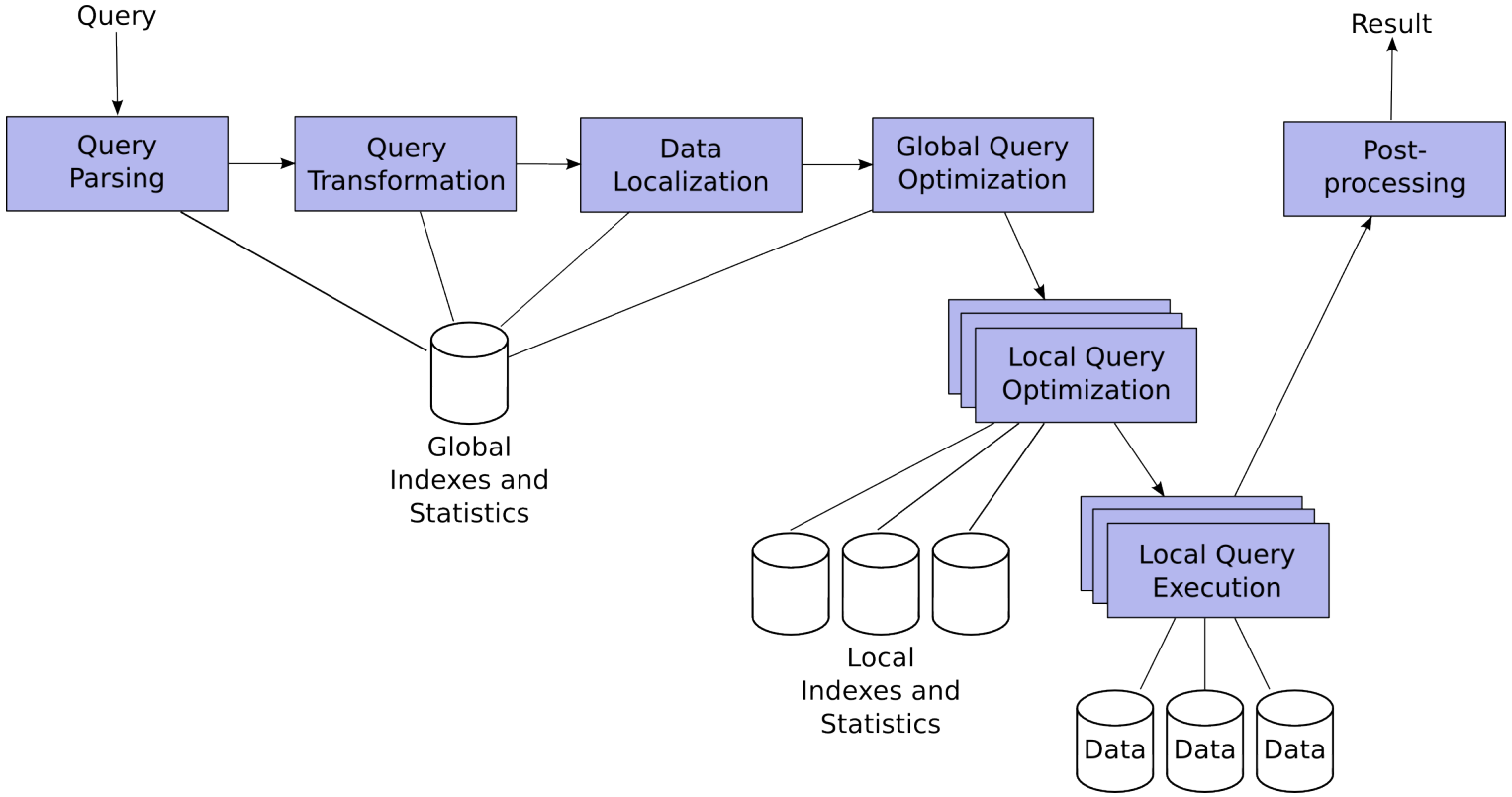
Alles wird mehr oder weniger gleich verwendet

Verteilte Anfragebearbeitung und Linked-Open-Data

Verteilte Anfragebearbeitung für Linked-Open-Data

- Entspricht in etwa der virtuellen Datenintegration
- Vieles ähnlich zu verteilten Datenbanken
- Strukturierte Anfragen: SPARQL

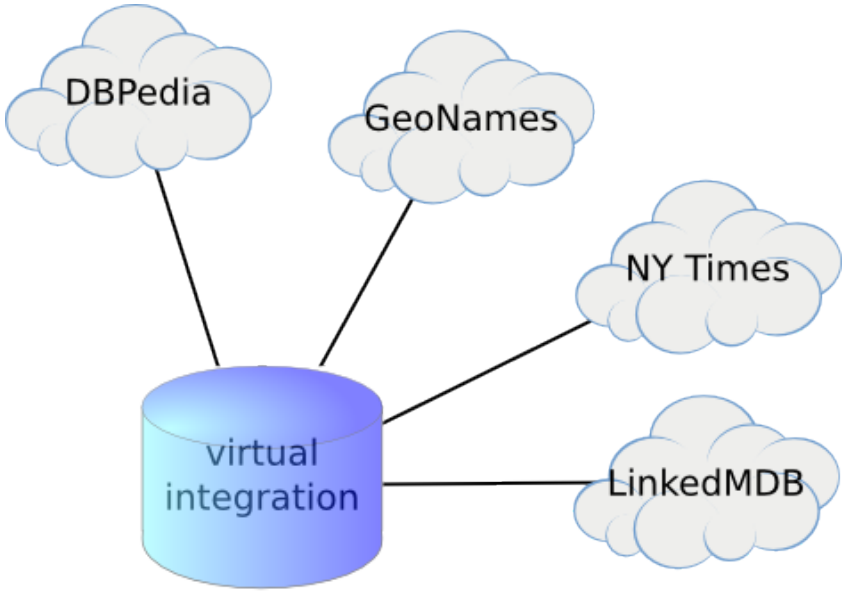
Grundlegende Schritte



Naive verteilte Anfragebearbeitung

Szenario

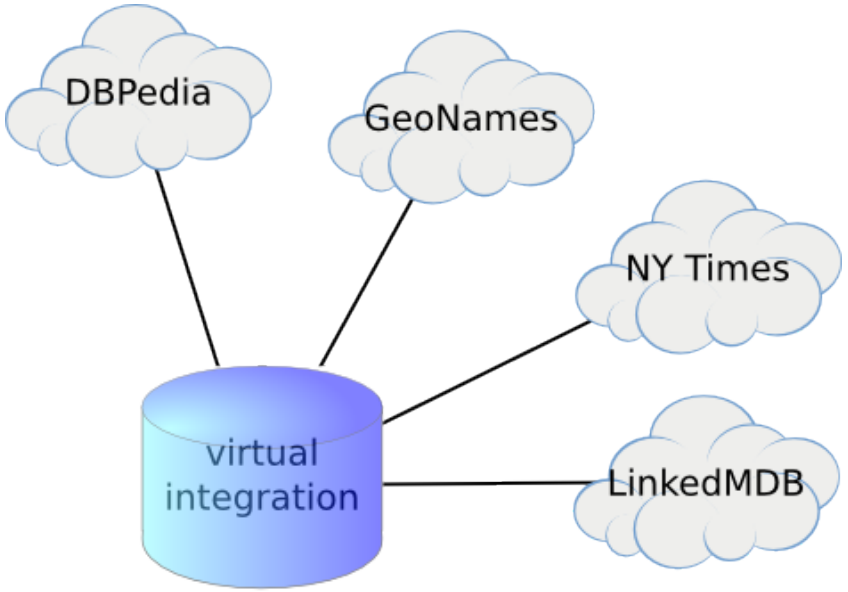
- SPARQL-Endpoints
- Keine Indexe



Naive verteilte Anfragebearbeitung

Szenario

- SPARQL-Endpoints
- Keine Indexe



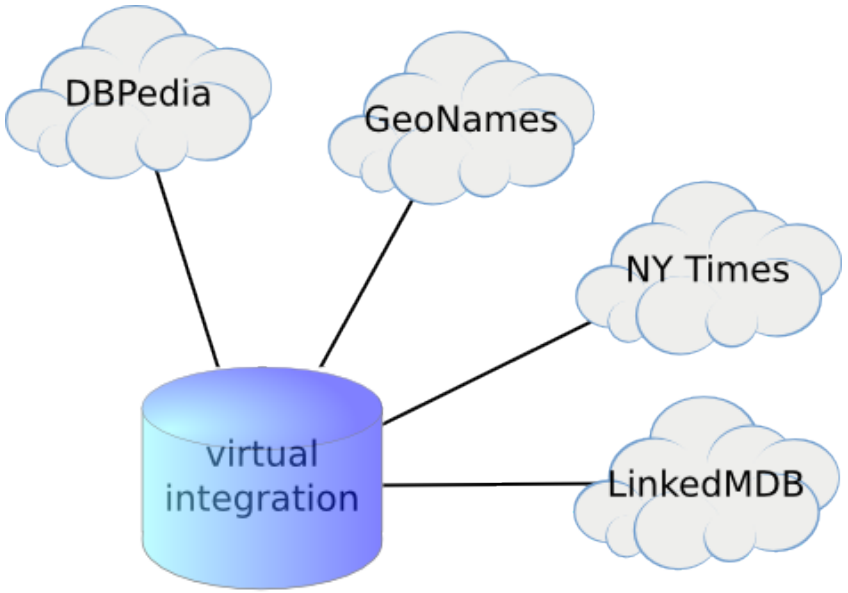
Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Naive verteilte Anfragebearbeitung

Beispielanfrage

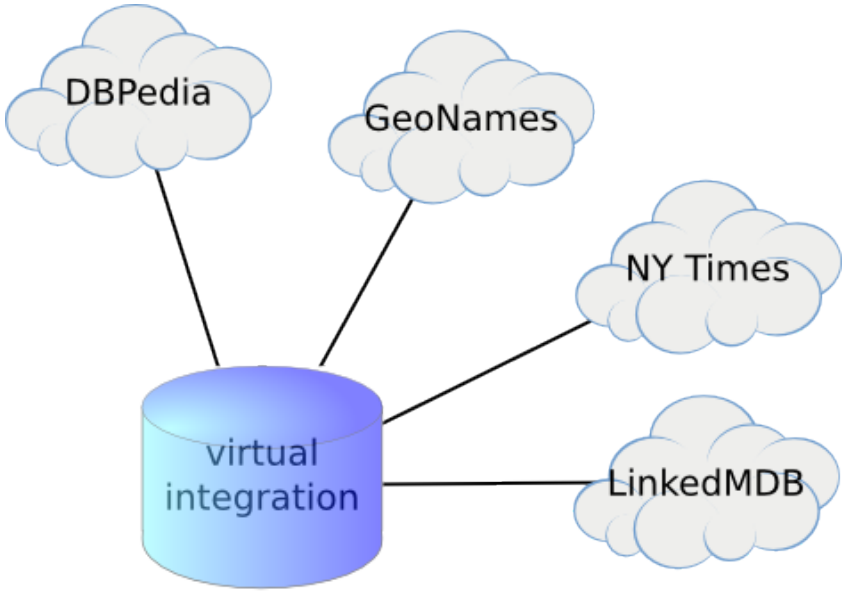
```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```



Naive verteilte Anfragebearbeitung

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

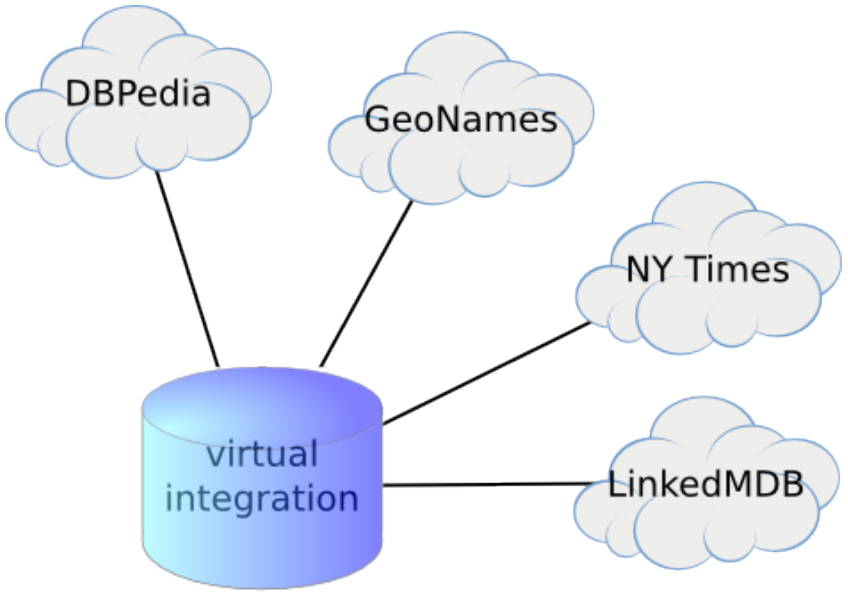


- Sende Nachricht mit Triple-Pattern an alle 4 Quellen
  $\rightarrow$ 4 Requests
- Empfange 200 Mappings für ?Country und ?Capital
  e.g., ?Country=ex:Germany, ?Capital=ex:Berlin

Naive verteilte Anfragebearbeitung

Beispielanfrage

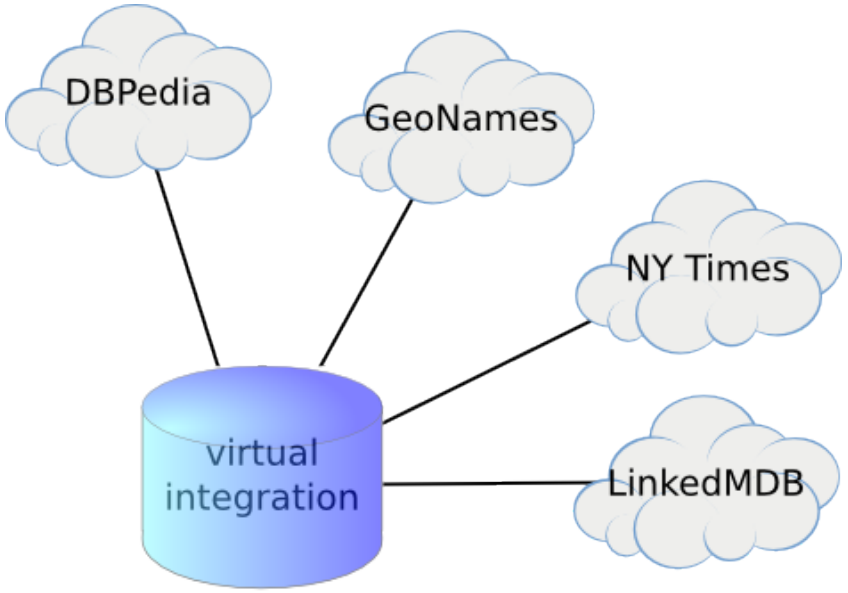
```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```



Naive verteilte Anfragebearbeitung

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

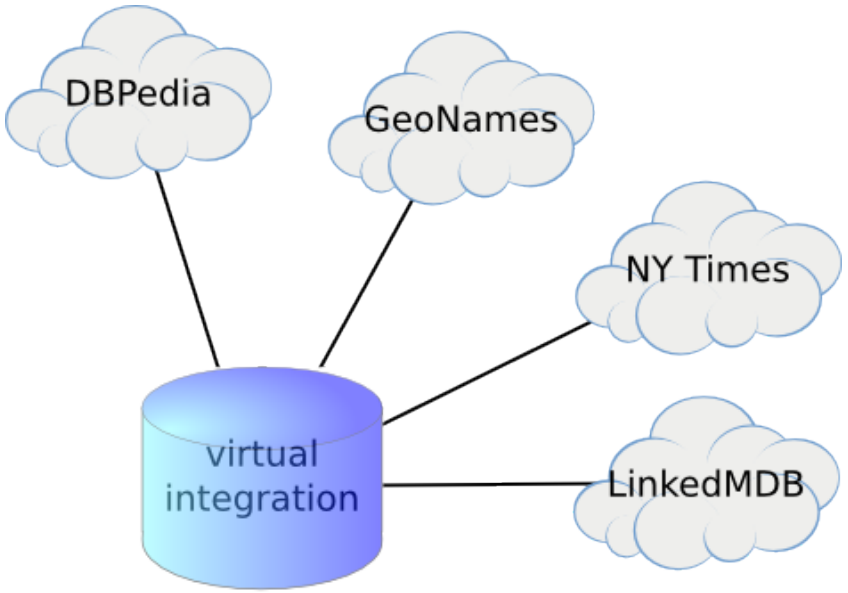


- Benutze Ergebnisse für 2. Triple-Statement (Nested-Loop)
- 200×4 Requests
  e.g., SELECT ?CountryPop WHERE {ex:Germany ex:population ?CountryPop .}
- 150 Mappings

Naive verteilte Anfragebearbeitung

Beispielanfrage

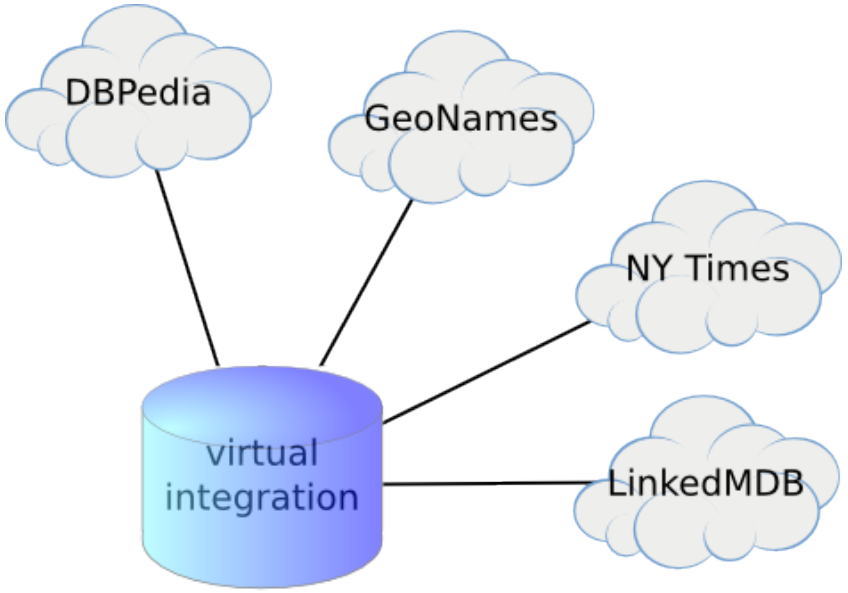
```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```



Naive verteilte Anfragebearbeitung

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```



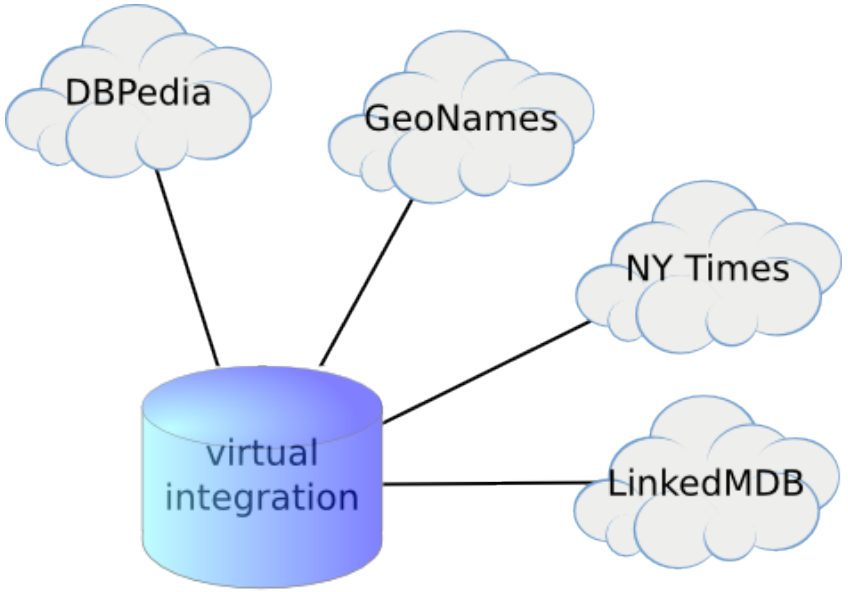
- Benutze Ergebnisse für 3. Triple-Statement (Nested-Loop)
- 150×4 Requests

  e.g., SELECT ?CapitalPop WHERE {ex:Berlin ex:population ?CapitalPop .}

Naive verteilte Anfragebearbeitung

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```



Insgesamt:

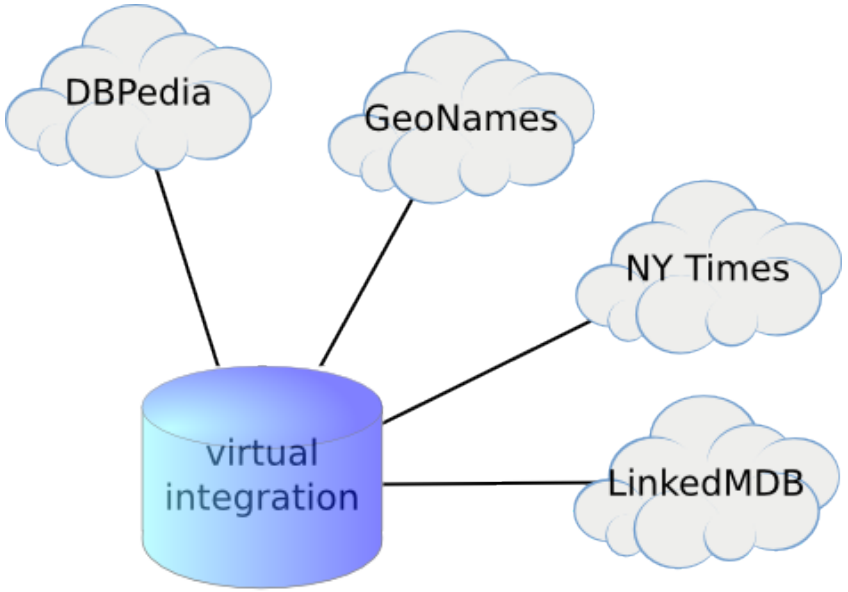
$4 + 200 \times 4 + 150 \times 4 = 1404$

Requests

Naive verteilte Anfragebearbeitung

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```



Insgesamt:

$4 + 200 \times 4 + 150 \times 4 = 1404$

Requests

Viele (unnötige) Requests an Datenquellen

Anfrageoptimierung

Heuristik

- Minimiere Kommunikationskosten

Methoden

- Vereinfachung und Filter Pushing
- Source Selection
- Remote Subqueries
- Optimierung der Joinreihenfolge
- Operator-Implementierungen

Anfrageoptimierung

Heuristik

- Minimiere Kommunikationskosten

Methoden

- Vereinfachung und Filter Pushing
- Source Selection
- Remote Subqueries
- Optimierung der Joinreihenfolge
- Operator-Implementierungen

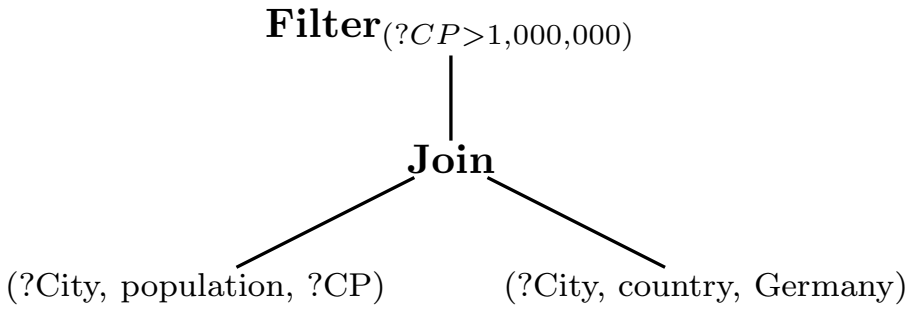
Vereinfachung und Filter Pushing

Normalisierung und Vereinfachung

- Widersprüchliche Prädikate
- Prädikate, die immer false/true ergeben
- Irrelevante Variablen
- etc.

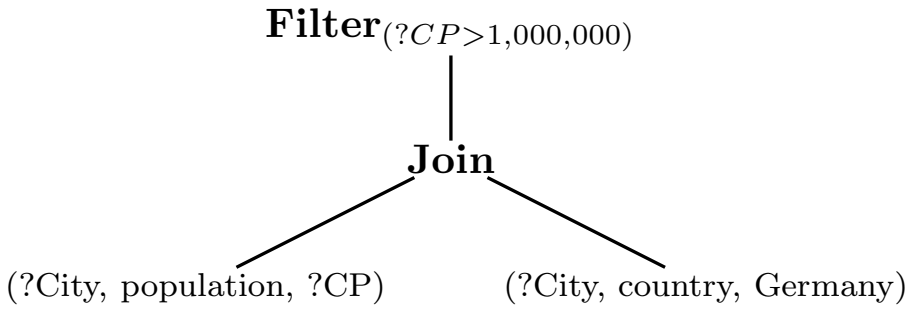
Vereinfachung und Filter Pushing

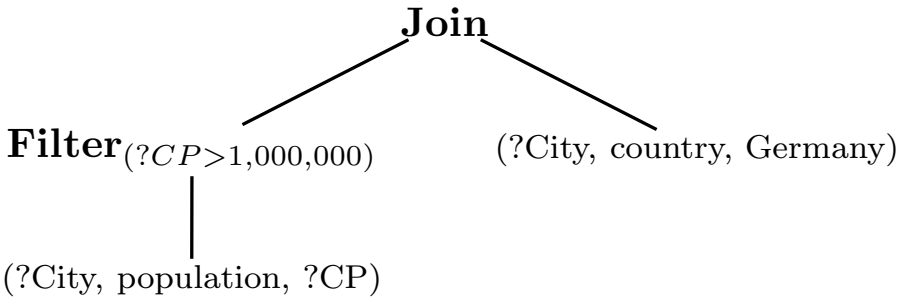
Frühe Evaluierung von Filtern reduziert die Größe der Zwischenergebnisse



Vereinfachung und Filter Pushing

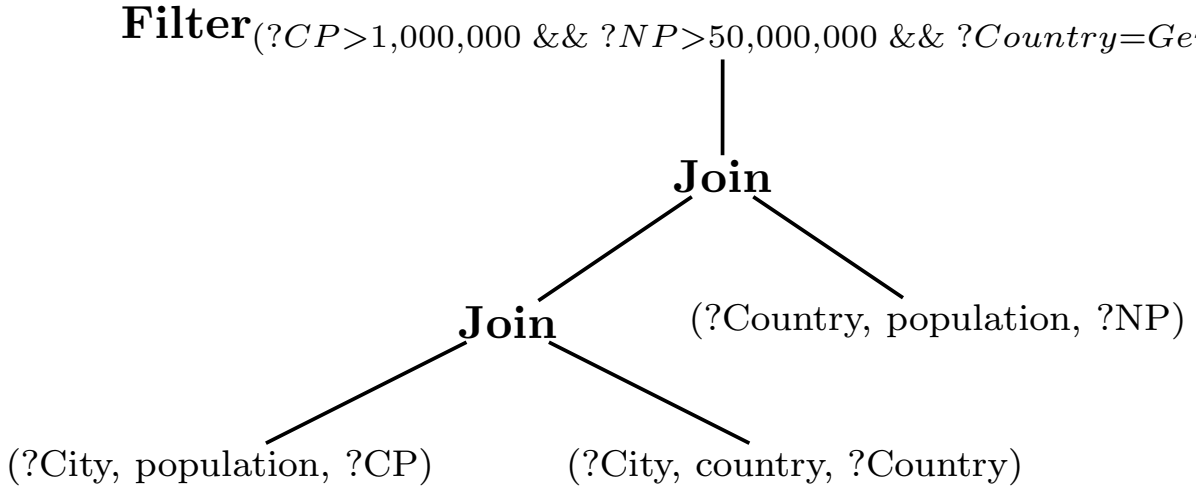
Frühe Evaluierung von Filtern reduziert die Größe der Zwischenergebnisse





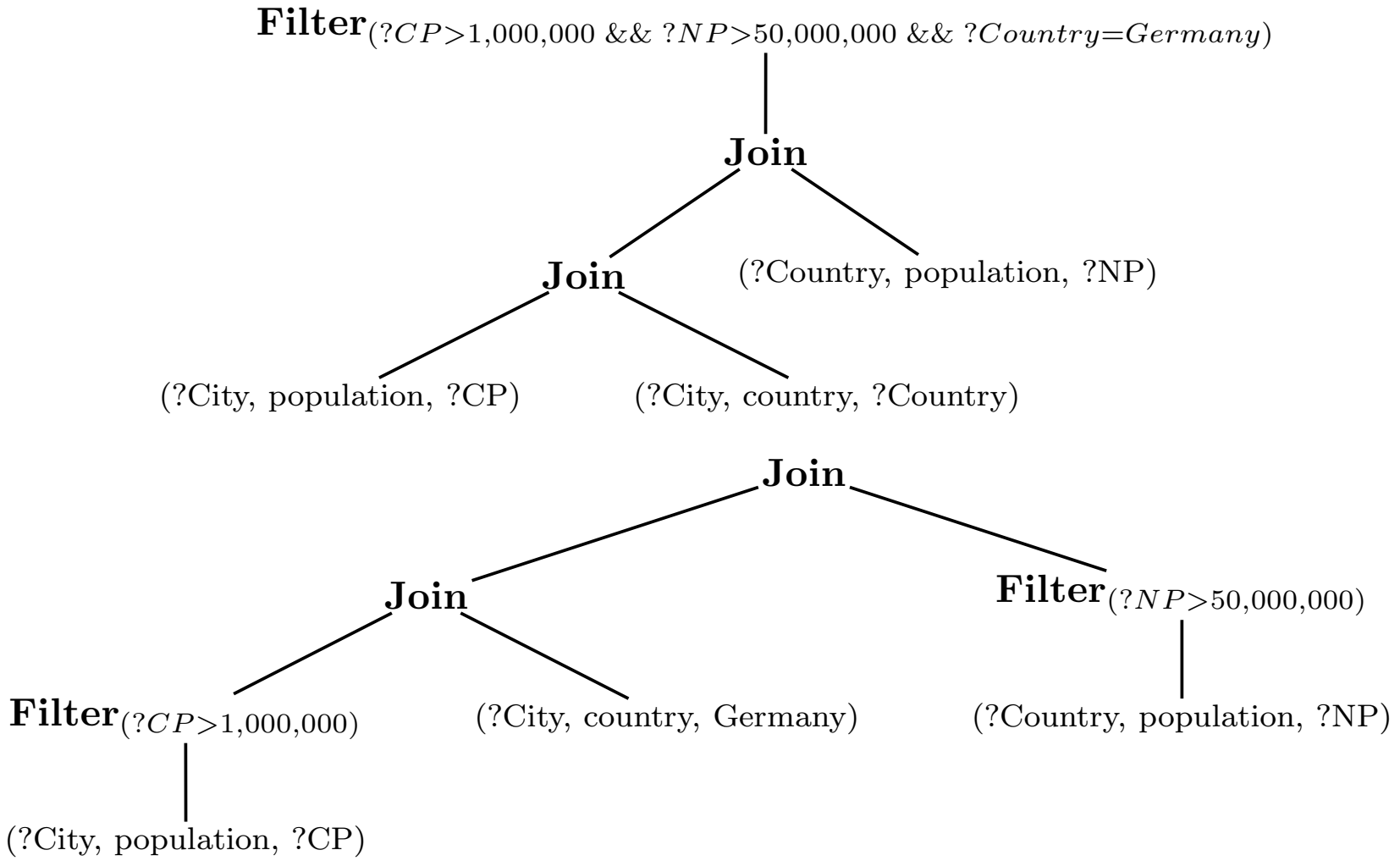
Vereinfachung und Filter Pushing

Frühe Evaluierung von Filtern reduziert die Größe der Zwischenergebnisse



Vereinfachung und Filter Pushing

Frühe Evaluierung von Filtern reduziert die Größe der Zwischenergebnisse



Source Selection

Ziel

- Identifikation von Quellen mit für die Anfrage relevanten Daten

Ansätze

- Keine zusätzlichen Informationen
- SPARQL ASK Requests und Caching
- Statistiken und Indexe

Source Selection

Ziel

- Identifikation von Quellen mit für die Anfrage relevanten Daten

Ansätze

- Keine zusätzlichen Informationen
- SPARQL ASK Requests und Caching
- Statistiken und Indexe

Indexe und Statistiken

- Keyword-Indexes
- Schema-Indexe
- URI indexe
- Häufige Pfade
- Service-Level Descriptions
- VoID Statistics (Vocabulary of Interlinked Datasets)
- Histogramme

Source Selection

Ziel

- Identifikation von Quellen mit für die Anfrage relevanten Daten

Ansätze

- Keine zusätzlichen Informationen
- SPARQL ASK Requests und Caching
- Statistiken und Indexe

Anwendbarkeit

Es kommt sehr stark auf den Grad der Kooperation der Quellen an!

Source Selection mit SPARQL ASK

Keine Kooperation der Quellen nötig

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Für jedes Triple-Pattern jeweils eine Anfrage an jede Quelle
Antwort: true/false

```
ASK {
  ?Country ex:capital ?Capital .
}
```

Source Selection mit SPARQL ASK

Keine Kooperation der Quellen nötig

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Für jedes Triple-Pattern jeweils eine Anfrage an jede Quelle
Antwort: true/false

```
ASK {
  ?Country ex:capital ?Capital .
}
```

```
ASK {
  ?Country ex:population ?CountryPop .
}
```

Source Selection mit SPARQL ASK

Keine Kooperation der Quellen nötig

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Für jedes Triple-Pattern jeweils eine Anfrage an jede Quelle
Antwort: true/false

```
ASK {
  ?Country ex:capital ?Capital .
}
```

```
ASK {
  ?Capital ex:population ?CapitalPop .
}
```

```
ASK {
  ?Country ex:population ?CountryPop .
}
```

Source Selection mit VoID Descriptions

Beispiel DBpedia[2] – Kooperation der Quellen nötig

- Prefixes

```
@prefix owl: http://www.w3.org/2002/07/owl#.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix dbpo: <http://dbpedia.org/ontology/> .
...
```

- General Information
- Basic Statistics
- Predicate Statistics
- Class Statistics

[2] http://code.google.com/p/fbench/source/browse/trunk/EvalBenchmark/suites/SPLENDID/void/dbpedia3.5.1_subset-void.n3?r=119

Source Selection mit VoID Descriptions

Beispiel DBpedia[2] – Kooperation der Quellen nötig

- Prefixes
- General Information
  `void:sparqlEndpoint <http://dbpedia.org/sparql> ;`
  `...`
- Basic Statistics
- Predicate Statistics
- Class Statistics

[2] http://code.google.com/p/fbench/source/browse/trunk/EvalBenchmark/suites/SPLENDID/void/dbpedia3.5.1_subset-void.n3?r=119

Source Selection mit VoID Descriptions

Beispiel DBpedia[2] – Kooperation der Quellen nötig

- Prefixes
- General Information
- Basic Statistics

```
void:triples "43620475" xsd:integer ;
void:entities "2222456" xsd:integer ;
void:properties "1063" xsd:integer ;
void:distinctSubjects "9495865" xsd:integer ;
void:distinctObjects "13636604" xsd:integer ;
```

- Predicate Statistics
- Class Statistics

[2] http://code.google.com/p/fbench/source/browse/trunk/EvalBenchmark/suites/SPLENDID/void/dbpedia3.5.1_subset-void.n3?r=119

Source Selection mit VoID Descriptions

Beispiel DBpedia[2] – Kooperation der Quellen nötig

- Prefixes
- General Information
- Basic Statistics
- Predicate Statistics

```
void:propertyPartition [
void:property dbpo:aSide ;
void:triples "1600" xsd:integer ;
void:distinctSubjects "1552" xsd:integer ;
void:distinctObjects "1554" xsd:integer
] , [
void:property dbpo:abbreviation ;
void:triples "1144" xsd:integer ;
void:distinctSubjects "1141" xsd:integer ;
void:distinctObjects "1096" xsd:integer
] , [
...
];
```

- Class Statistics

[2] subset-void.n3?r=119

Source Selection mit VoID Descriptions

Beispiel DBpedia[2] – Kooperation der Quellen nötig

- Prefixes
- General Information
- Basic Statistics
- Predicate Statistics
- Class Statistics

```
void:classPartition [
void:class dbpo:Activity ;
void:entities "1234" xsd:integer
] , [
void:class dbpo:Actor ;
void:entities "37898" xsd:integer
] , [
...
]
```

[2] http://code.google.com/p/fbench/source/browse/trunk/EvalBenchmark/suites/SPLENDID/void/dbpedia3.5.1_subset-void.n3?r=119

Remote Subqueries

So viel wie möglich von den SPARQL-Endpunkten berechnen lassen und Größe von Zwischenergebnissen minimieren

- Parallele Ausführung
- Verringerte Netzwerklast
- Lastverteilung
- Bessere Antwortzeit

Optimierungen

- Exclusive Join Groups
- Identifiziere Teilanfragen, die mehrmals im Anfragegraph enthalten sind

Remote Subqueries

So viel wie möglich von den SPARQL-Endpunkten berechnen lassen und Größe von Zwischenergebnissen minimieren

- Parallele Ausführung
- Verringerte Netzwerklast
- Lastverteilung
- Bessere Antwortzeit

Optimierungen

- Exclusive Join Groups
- Identifiziere Teilanfragen, die mehrmals im Anfragegraph enthalten sind

Exclusive Join Groups

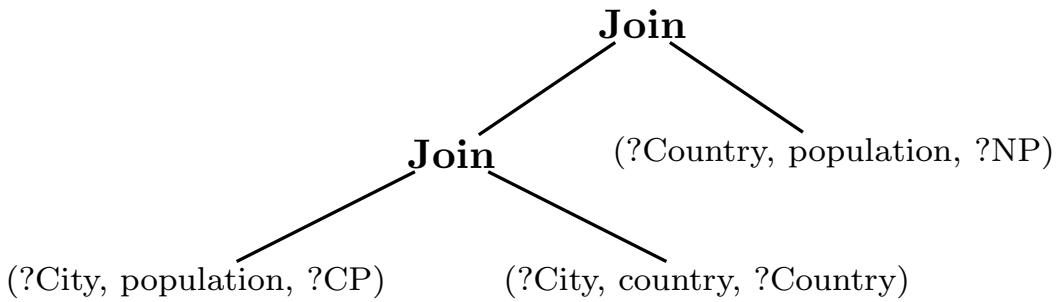
Exclusive Join Groups

- Gruppen von Triple-Statements mit nur einer relevanten Quelle
- Kombiniere diese Triple-Statements in einer Teilanfrage

Exclusive Join Groups

Exclusive Join Groups

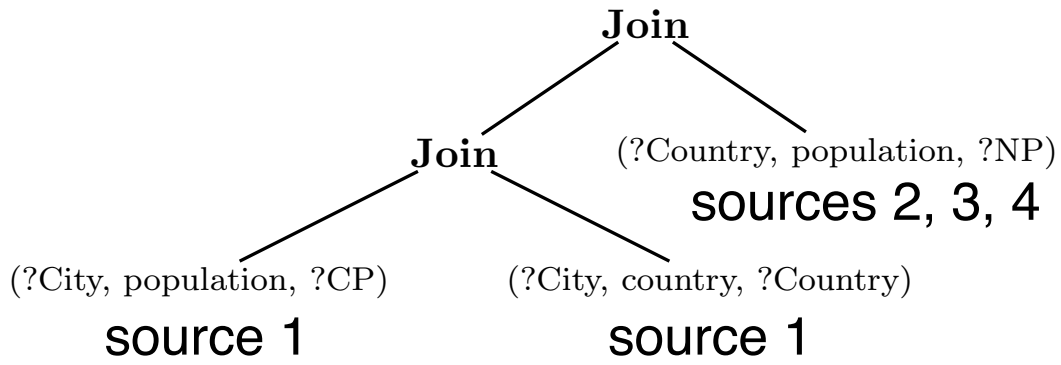
- Gruppen von Triple-Statements mit nur einer relevanten Quelle
- Kombiniere diese Triple-Statements in einer Teilanfrage



Exclusive Join Groups

Exclusive Join Groups

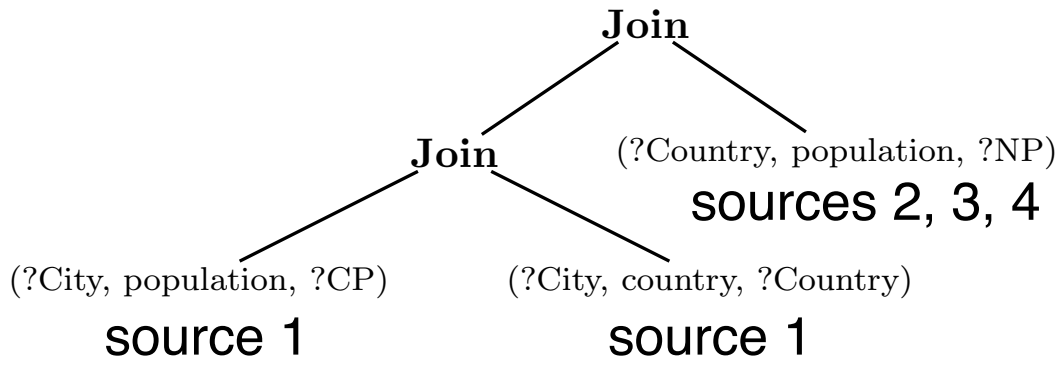
- Gruppen von Triple-Statements mit nur einer relevanten Quelle
- Kombiniere diese Triple-Statements in einer Teilanfrage



Exclusive Join Groups

Exclusive Join Groups

- Gruppen von Triple-Statements mit nur einer relevanten Quelle
- Kombiniere diese Triple-Statements in einer Teilanfrage



Ein Join kann direkt an Source 1 berechnet werden (1 Anfrage)

Joinreihenfolge

Kriterien

- Kleine Zwischenergebnisse, hohe Selektivität
- Exclusive Join Groups
- Variable Counting
- Kardinalitätsabschätzungen mit Statistiken

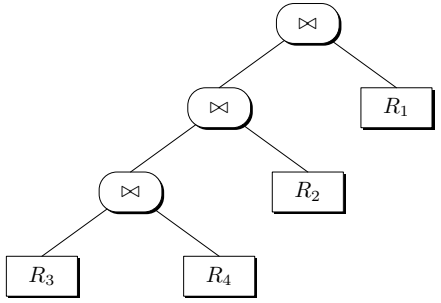
Joinreihenfolge

Kriterien

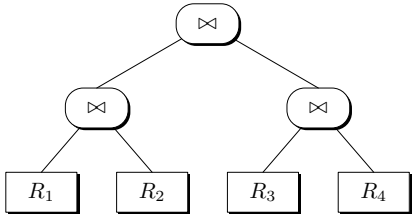
- Kleine Zwischenergebnisse, hohe Selektivität
- Exclusive Join Groups
- Variable Counting
- Kardinalitätsabschätzungen mit Statistiken

Standardtechniken

- Lineare Bäume vs. buschige Bäume
- Kostenmodell mit Statistiken
- Dynamic Programming



Linearer Joinbaum



Buschiger Joinbaum

Kosten- und Kardinalitätsabschätzungen

Kostenmodell für Kommunikationskosten (verteilte Datenbanken)

Beispiel:

- Source S1 relevant für Triple-Statement q_1
- Source S2 relevant für Triple-Statement q_2

Kosten Coordinator Join:

$$c = card(q_1) \cdot c_t + card(q_2) \cdot c_t + 2 \cdot c_m$$

Kosten Coordinator Semi-Join mit Request pro Binding:

$$c = card(q_1) \cdot c_t + card(q_1) \cdot c_m + card(q_2') \cdot c_t$$

Kostenfaktoren:

- c_t: Kosten für den Transfer eines Triples
- c_m: Kosten für dem Transfer einer Nachricht
- $card(q_2')$: Join-Partner an Source 2 für Tuple von Source 1

Kosten- und Kardinalitätsabschätzungen

Kostenmodell für Kommunikationskosten (verteilte Datenbanken)

Beispiel:

- Source S1 relevant für Triple-Statement q_1
- Source S2 relevant für Triple-Statement q_2

Kosten Coordinator Join:

$$c = card(q_1) \cdot c_t + card(q_2) \cdot c_t + 2 \cdot c_m$$

Kosten Coordinator Semi-Join mit Request pro Binding:

$$c = card(q_1) \cdot c_t + card(q_1) \cdot c_m + card(q_2') \cdot c_t$$

Kostenfaktoren:

- c_t: Kosten für den Transfer eines Triples
- c_m: Kosten für dem Transfer einer Nachricht
- $card(q_2')$: Join-Partner an Source 2 für Tuple von Source 1

Kosten- und Kardinalitätsabschätzungen

Kardinalitätsabschätzungen mit VoID-Statistiken [GS11]

Exakte Kardinalitäten sind gegeben für

- Gebundene Predicates (?,p,?)
  VoID Predicate Statistics
- Prädikat `rdf:type` mit gebundenem Object (?,rdf:type,o)
  VoID Class Statistics

Abschätzungen für alles andere

Beispiel (s,p,?)

- $card(s,p,?) = \frac{card(?,p,?)}{\text{dist. subjects for p}}$; beides in VoID Predicate Statistics

Beispiel (s,?,?)

- $card(s,?,?) = \frac{\text{triple in source}}{\text{dist. subjects in source}}$; beides in VoID General Statistics

Abschätzungen für Joins

Kosten- und Kardinalitätsabschätzungen

Kardinalitätsabschätzungen mit VoID-Statistiken [GS11]

Exakte Kardinalitäten sind gegeben für

- Gebundene Predicates (?,p,?)
  VoID Predicate Statistics
- Prädikat `rdf:type` mit gebundenem Object (?,rdf:type,o)
  VoID Class Statistics

Abschätzungen für alles andere

Beispiel (s,p,?)

- $card(s,p,?) = \frac{card(?,p,?)}{\text{dist. subjects for p}}$; beides in VoID Predicate Statistics

Beispiel (s,?,?)

- $card(s,?,?) = \frac{\text{triple in source}}{\text{dist. subjects in source}}$; beides in VoID General Statistics

Abschätzungen für Joins

Operator-Implementierungen

Join-Implementierungen

- Coordinator Join
  - Daten von Quellen downloaden
  - Join lokal berechnen
- Remote Join
  - Exclusive Join Groups
- Coordinator Semi-Join
  - Evaluiere ein Triple-Pattern
  - Benutze Teilergebnisse, um weitere Triple von den Sourcen anzufragen
  - Single vs. Block Requests
  - Berechne Join am Koordinator
- Independent Join Groups
  - Erweiterung Coordinator Semi-Join
  - Reduktion der Nachrichten
  - Mehrere Joins parallel

Operator-Implementierungen

Join-Implementierungen

- Coordinator Join
  - Daten von Quellen downloaden
  - Join lokal berechnen
- Remote Join
  - Exclusive Join Groups
- Coordinator Semi-Join
  - Evaluiere ein Triple-Pattern
  - Benutze Teilergebnisse, um weitere Triple von den Sourcen anzufragen
  - Single vs. Block Requests
  - Berechne Join am Koordinator
- Independent Join Groups
  - Erweiterung Coordinator Semi-Join
  - Reduktion der Nachrichten
  - Mehrere Joins parallel

Operator-Implementierungen

Join-Implementierungen

- Coordinator Join
  - Daten von Quellen downloaden
  - Join lokal berechnen
- Remote Join
  - Exclusive Join Groups
- Coordinator Semi-Join
  - Evaluiere ein Triple-Pattern
  - Benutze Teilergebnisse, um weitere Triple von den Sourcen anzufragen
  - Single vs. Block Requests
  - Berechne Join am Koordinator
- Independent Join Groups
  - Erweiterung Coordinator Semi-Join
  - Reduktion der Nachrichten
  - Mehrere Joins parallel

Operator-Implementierungen

Join-Implementierungen

- Coordinator Join
  - Daten von Quellen downloaden
  - Join lokal berechnen
- Remote Join
  - Exclusive Join Groups
- Coordinator Semi-Join
  - Evaluiere ein Triple-Pattern
  - Benutze Teilergebnisse, um weitere Triple von den Sourcen anzufragen
  - Single vs. Block Requests
  - Berechne Join am Koordinator
- Independent Join Groups
  - Erweiterung Coordinator Semi-Join
  - Reduktion der Nachrichten
  - Mehrere Joins parallel

Operator-Implementierungen

Join-Implementierungen

- Coordinator Join
  - Daten von Quellen downloaden
  - Join lokal berechnen
- Remote Join
  - Exclusive Join Groups
- Coordinator Semi-Join
  - Evaluiere ein Triple-Pattern
  - Benutze Teilergebnisse, um weitere Triple von den Sourcen anzufragen
  - Single vs. Block Requests
  - Berechne Join am Koordinator
- Independent Join Groups
  - Erweiterung Coordinator Semi-Join
  - Reduktion der Nachrichten
  - Mehrere Joins parallel

Coordinator Semi-Join mit Blöcken

Coordinator Semi-Join

- Naive Variante
  - Ein Binding (Variablenwert) pro Request
- Effizientere Variante [SHH+11]
  - Mehrere Bindings pro Request (Blöcke)

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing

...
```

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage Block Coordinator Join:

```
(ex:Germany, ex:population, ?Population)
UNION
(ex:USA, ex:population, ?Population)
UNION
(ex:China, ex:population, ?Population)
```

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage Block Coordinator Join:

```
(ex:Germany, ex:population, ?Population)
UNION
(ex:USA, ex:population, ?Population)
UNION
(ex:China, ex:population, ?Population)
```

Ergebnisse zu S2:

```
?Population=308,745,538
?Population=81,757,600
?Population=1,339,724,852
```

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage Block Coordinator Join:

```
(ex:Germany, ex:population, ?Population)
UNION
(ex:USA, ex:population, ?Population)
UNION
(ex:China, ex:population, ?Population)
```

Ergebnisse zu S2:

```
?Population=308,745,538
?Population=81,757,600
?Population=1,339,724,852
```

Problem:

- Zufällige Reihenfolge der Ergebnisse

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage Block Coordinator Join:

```
(ex:Germany, ex:population, ?Population)
UNION
(ex:USA, ex:population, ?Population)
UNION
(ex:China, ex:population, ?Population)
```

Ergebnisse zu S2:

```
?Population=308,745,538
?Population=81,757,600
?Population=1,339,724,852
```

Lösung:

- Variablen umbenennen

Coordinator Semi-Join mit Blöcken

Beispiel

```
S1: ?Country ex:capital ?Capital
S2: ?Country ex:population ?CountryPop
S3: ?Capital ex:population ?CapitalPop
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage Block Coordinator Join:

```
(ex:Germany, ex:population, ?Population_1)
UNION
(ex:USA, ex:population, ?Population_2)
UNION
(ex:China, ex:population, ?Population_3)
```

Ergebnisse zu S2:

```
?Population_2=308,745,538
?Population_1=81,757,600
?Population_3=1,339,724,852
```

Independent Join Groups [SHH+11]

Gruppiere *unabhängige* Triple-Statements zu einer Teilanfrage

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Independent Join Groups [SHH+11]

Gruppiere *unabhängige* Triple-Statements zu einer Teilanfrage

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Independent Join Groups [SHH+11]

Gruppiere *unabhängige* Triple-Statements zu einer Teilanfrage

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing

...
```

Independent Join Groups [SHH+11]

Gruppiere *unabhängige* Triple-Statements zu einer Teilanfrage

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing

...
```

Teilanfrage zur Evaluierung von S2 und S3:

```
(ex:Germany, ex:population, ?CountryPop_0)
UNION
(ex:Berlin, ex:population, ?CapitalPop_1)
```

Independent Join Groups [SHH+11]

Gruppiere *unabhängige* Triple-Statements zu einer Teilanfrage

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage zur Evaluierung von S2 und S3:

```
(ex:Germany, ex:population, ?CountryPop_0)
UNION
(ex:Berlin, ex:population, ?CapitalPop_1)
```

Ergebnisse zu S2 und S3:

```
?CapitalPop_1=3,500,000
?CountryPop_0=83,000,000 ...
```

Independent Join Groups [SHH+11]

Gruppiere *unabhängige* Triple-Statements zu einer Teilanfrage

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Ergebnisse zu S1:

```
?Country=ex:Germany, ?Capital=ex:Berlin
?Country=ex:USA, ?Capital=ex:WashingtonDC
?Country=ex:China, ?Capital=ex:Beijing
...
```

Teilanfrage zur Evaluierung von S2 und S3:

```
(ex:Germany, ex:population, ?CountryPop_0)
UNION
(ex:Berlin, ex:population, ?CapitalPop_1)
```

Ergebnisse zu S2 und S3:

```
?CapitalPop_1=3,500,000
?CountryPop_0=83,000,000 ...
```

Verbesserung: Blöcke von Bindings statt Einzelanfragen

SPARQL-Erweiterungen

Strategien bisher

- SPARQL 1.0
- Keine Kommunikation der Quellen untereinander
- Alles läuft über Koordinator

Erweiterungen

- SPARQL 1.1 BINDINGS Operator
- SPARQL 1.1 SERVICE Operator

SPARQL-Erweiterungen

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Beispielanfrage mit SERVICE Operator

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  SERVICE <http://example.org/exampleEndpoint>
     { ?Capital ex:population ?CapitalPop }
}
```

SPARQL-Erweiterungen

Beispielanfrage

```
SELECT ?Country ?Capital
       ?CountryPop ?CapitalPop WHERE {
  ?Country ex:capital ?Capital .
  ?Country ex:population ?CountryPop .
  ?Capital ex:population ?CapitalPop .
}
```

Teilanfrage zur Evaluierung von S2:

```
SELECT ?Population_1 ?Population_2
       ?Population_3 WHERE {
  { ex:Germany ex:population ?Population_1 }
  UNION
  { ex:USA ex:population ?Population_2 }
  UNION
  { ex:China ex:population ?Population_3 }
}
```

Beispielanfrage mit BINDINGS Operator

```
SELECT ?Country ?CountryPop WHERE {
  ?Country ex:population ?CountryPop .
  BINDINGS ?Country {
  (ex:Germany)
  (ex:USA)
  (ex:China)
  }
```

Zusammenfassung

	Data Warehouses	**Suchmaschinen**	**Lookup-basierte Anfragebearbeitung**	**Verteilte Anfragebearbeitung**
Source Interface	hptsl. RDF-Dumps	alle	Dereferenz. URIs	SPARQL-Endpoints
Live-Zugriff	nein	nein	ja	ja
Hilfsdaten	lok. Statistiken	gecrawlter Index	–	Index & Statistiken
Antwortzeit	schnell	schnell	medium	medium
Durchsatz	schnell	schnell	langsam	medium
Aktualität	schlecht	medium	hoch	medium/hoch

In Anlehnung an [HL10]

Zusammenfassung

- Es gibt verschiedene Zugriffsarten für Linked Open Data
  SPARQL-Endpoints, Dereferenzieren von URIs, RDF dumps
- Es gibt verschiedene Arten auf die Daten zuzugreifen und Anfragen zu bearbeiten
  Browsing, Suchmaschinen, unstrukturierte und strukturierte Anfragen, Lookup-basierte Verfahren, verteilte Anfragebearbeitung
- Optimierungstechniken ähnlich zu verteilten Datenbanken
  Kostenmodell, Kardinalitäten, Operatorimplementierungen
- Optimierungspotential wird durch die Quellen bestimmt
  Indexe, SPARQL 1.0 / 1.1
- Aktualität vs. Geschwindigkeit

Zusammenfassung

- Es gibt verschiedene Zugriffsarten für Linked Open Data
  SPARQL-Endpoints, Dereferenzieren von URIs, RDF dumps
- Es gibt verschiedene Arten auf die Daten zuzugreifen und Anfragen zu bearbeiten
  Browsing, Suchmaschinen, unstrukturierte und strukturierte Anfragen, Lookup-basierte Verfahren, verteilte Anfragebearbeitung
- Optimierungstechniken ähnlich zu verteilten Datenbanken
  Kostenmodell, Kardinalitäten, Operatorimplementierungen
- Optimierungspotential wird durch die Quellen bestimmt
  Indexe, SPARQL 1.0 / 1.1
- Aktualität vs. Geschwindigkeit

Zusammenfassung

- Es gibt verschiedene Zugriffsarten für Linked Open Data
  SPARQL-Endpoints, Dereferenzieren von URIs, RDF dumps
- Es gibt verschiedene Arten auf die Daten zuzugreifen und Anfragen zu bearbeiten
  Browsing, Suchmaschinen, unstrukturierte und strukturierte Anfragen, Lookup-basierte Verfahren, verteilte Anfragebearbeitung
- Optimierungstechniken ähnlich zu verteilten Datenbanken
  Kostenmodell, Kardinalitäten, Operatorimplementierungen
- Optimierungspotential wird durch die Quellen bestimmt
  Indexe, SPARQL 1.0 / 1.1
- Aktualität vs. Geschwindigkeit

Zusammenfassung

- Es gibt verschiedene Zugriffsarten für Linked Open Data
  SPARQL-Endpoints, Dereferenzieren von URIs, RDF dumps
- Es gibt verschiedene Arten auf die Daten zuzugreifen und Anfragen zu bearbeiten
  Browsing, Suchmaschinen, unstrukturierte und strukturierte Anfragen, Lookup-basierte Verfahren, verteilte Anfragebearbeitung
- Optimierungstechniken ähnlich zu verteilten Datenbanken
  Kostenmodell, Kardinalitäten, Operatorimplementierungen
- Optimierungspotential wird durch die Quellen bestimmt
  Indexe, SPARQL 1.0 / 1.1
- Aktualität vs. Geschwindigkeit

Zusammenfassung

- Es gibt verschiedene Zugriffsarten für Linked Open Data
  SPARQL-Endpoints, Dereferenzieren von URIs, RDF dumps
- Es gibt verschiedene Arten auf die Daten zuzugreifen und Anfragen zu bearbeiten
  Browsing, Suchmaschinen, unstrukturierte und strukturierte Anfragen, Lookup-basierte Verfahren, verteilte Anfragebearbeitung
- Optimierungstechniken ähnlich zu verteilten Datenbanken
  Kostenmodell, Kardinalitäten, Operatorimplementierungen
- Optimierungspotential wird durch die Quellen bestimmt
  Indexe, SPARQL 1.0 / 1.1
- Aktualität vs. Geschwindigkeit

Literatur I

[DD99] Ruxandra Domenig and Klaus R. Dittrich. An overview and classification of mediated query systems. *SIGMOD Record*, 28(3):63–72, 1999.

[GS11] Olaf Görlitz and Steffen Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *COLD Workshop '11*, 2011.

[Har11] Olaf Hartig. Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In *ESWC (1)*, pages 154–169, 2011.

[HBF09] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. Executing SPARQL Queries over the Web of Linked Data. In *The Semantic Web - ISWC 2009*, volume 5823, pages 293–309. Springer Berlin / Heidelberg, 2009.

[HL10] Olaf Hartig and Andreas Langegger. A Database Perspective on Consuming Linked Data on the Web. *Datenbank-Spektrum*, 10(2):57–66, 2010.

[LT10] Günter Ladwig and Thanh Tran. Linked Data Query Processing Strategies. In *International Semantic Web Conference (ISWC'10)*, pages 453–469, 2010.

Literatur II

[LT11] Günter Ladwig and Thanh Tran. SIHJoin: querying remote and local linked data. In *ESWC'11*, pages 139–153, 2011.

[SHH+11] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *ISWC 2011*, pages 601–616. 2011.

[UHK+11] Jürgen Umbrich, Katja Hose, Marcel Karnstedt, Andreas Harth, and Axel Polleres. Comparing data summaries for processing live queries over linked data. *World Wide Web*, 14:495–544, 2011. 10.1007/s11280-010-0107-z.