Web of Data

Iztok Savnik, Famnit, UP

April, 2018.

Outline

1) Graph data model (RDF)
2) Popular graph databases on Web
3) Semantic Web
4) Linked data and applications
5) big3store

Graph data model (RDF + RDFS)

Graph data model

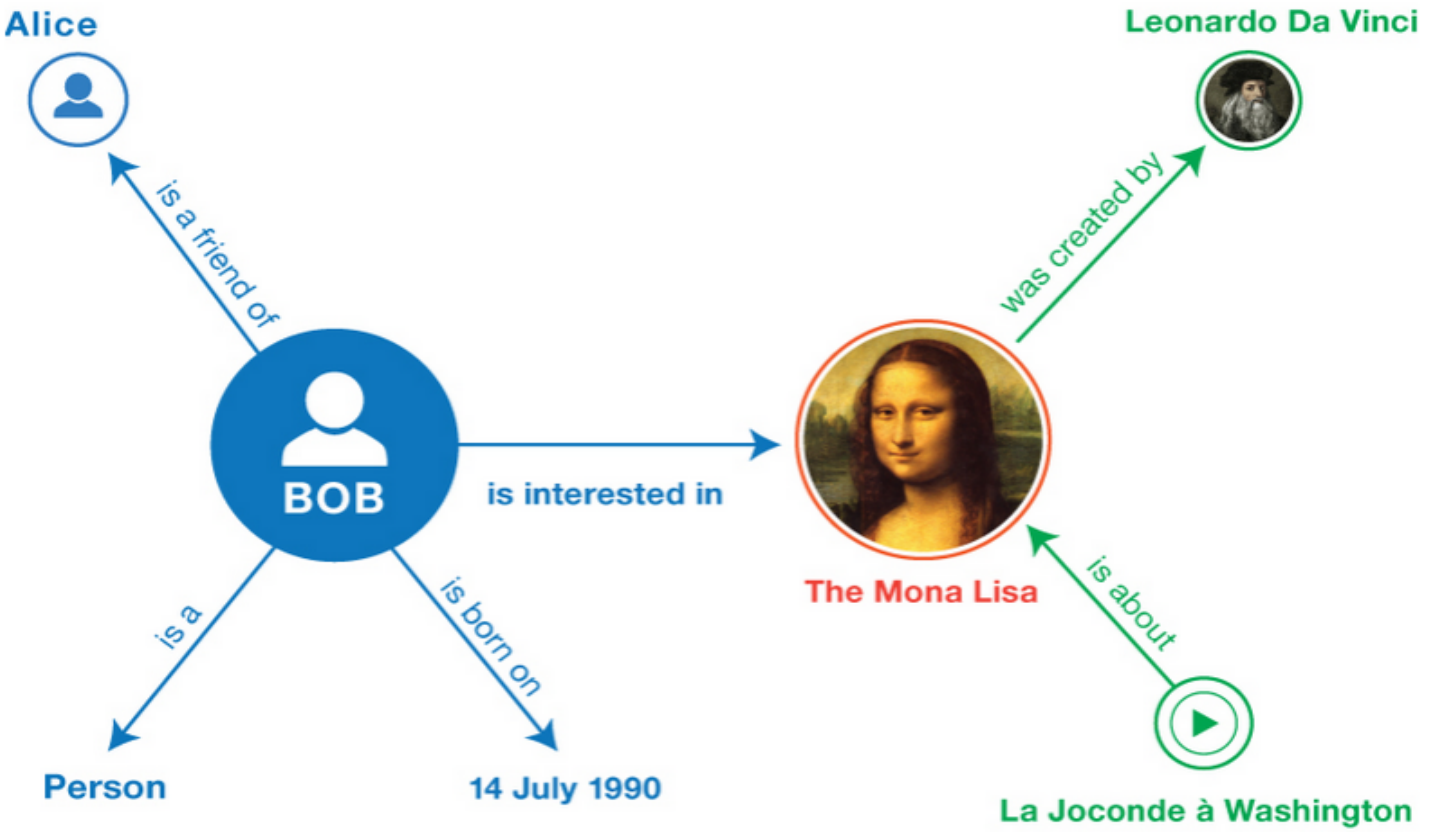
- Graph database
  - Database that uses graphs for the representation of data and queries
- Vertexes
  - Represent things, persons, concepts, classes, ...
- Arcs
  - Represent properties, relationships, associations, ...
  - Arcs have labels !

RDF

- Resource Description Framework
  - Tim Berners Lee, 1998-2009
  - This is movement !
- What is behind ?
  - Graphs
  - Triples (3)
  - Semantic data models
  - Human associative memory (psychology)
  - Associative neural networks
  - Hopfield Network

RDF

```
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa>
```



RDF syntax

- N3, TVS
- Turtle
- TriG
- N-Triples
- RDF/XML
- RDF/JSON

Name spaces

- Using short names for URL-s
  - Long names are tedious
- Simple but strong concept
- Defining name space:

prefix rdf:, namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#

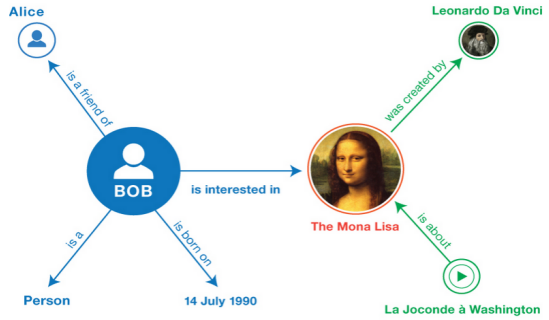
prefix rdfs:, namespace URI: http://www.w3.org/2000/01/rdf-schema#

prefix dc:, namespace URI: http://purl.org/dc/elements/1.1/

prefix owl:, namespace URI: http://www.w3.org/2002/07/owl#

prefix ex:, namespace URI: http://www.example.org/ (or http://www.example.com/)

prefix xsd:, namespace URI: http://www.w3.org/2001/XMLSchema#



N-Triples

<http://example.org/bob#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/Person> .
<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/knows> <http://example.org/alice#me> .
<http://example.org/bob#me> <http://schema.org/birthDate> "1990-07-04"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/topic_interest> <http://www.wikidata.org/entity/Q12418> .
<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/title> "Mona Lisa" .
<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/creator> <http://dbpedia.org/resource/Leonardo_da_Vinci> .
<http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619> <http://purl.org/dc/terms/subject> <

Turtle

```
01  BASE   <http://example.org/>
02  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
03  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
04  PREFIX schema: <http://schema.org/>
05  PREFIX dcterms: <http://purl.org/dc/terms/>
06  PREFIX wd: <http://www.wikidata.org/entity/>
07
08  <bob#me>
09      a foaf:Person ;
10      foaf:knows <alice#me> ;
11      schema:birthDate "1990-07-04"^^xsd:date ;
12      foaf:topic_interest wd:Q12418 .
13
14  wd:Q12418
15      dcterms:title "Mona Lisa" ;
16      dcterms:creator <http://dbpedia.org/resource/Leonardo_da_Vinci> .
17
18  <http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619>
19      dcterms:subject wd:Q12418 .
```

Additional RDF Constructs

- Complex values
  - Bags, lists, trees, graphs
- Empty nodes
- Types of atomic values
- Types of nodes
- Reification

RDF Schema

- RDFS
- Knowledge representation language
  - Not just graph any more !
  - AI Frames, Object Model
- Small dictionary for RDFS
  - rdfs:class, rdfs:subClassOf, rdfs:type
  - rdfs:property, rdfs:subPropertyOf
  - rdfs:domain, rdfs:range

RDFS Concepts

Construct	Syntactic form	Description
Class (a class)	**C** rdf:type rdfs:Class	**C** (a resource) is an RDF class
Property (a class)	**P** rdf:type rdf:Property	**P** (a resource) is an RDF property
type (a property)	**I** rdf:type **C**	**I** (a resource) is an instance of **C** (a class)
subClassOf (a property)	**C1** rdfs:subClassOf **C2**	**C1** (a class) is a subclass of **C2** (a class)
subPropertyOf (a property)	**P1** rdfs:subPropertyOf **P2**	**P1** (a property) is a sub-property of **P2** (a property)
domain (a property)	**P** rdfs:domain **C**	domain of **P** (a property) is **C** (a class)
range (a property)	**P** rdfs:range **C**	range of **P** (a property) is **C** (a class)

Classes

ex:MotorVehicle rdf:type rdfs:Class .
ex:PassengerVehicle rdf:type rdfs:Class .
ex:Van rdf:type rdfs:Class .
ex:Truck rdf:type rdfs:Class .
ex:MiniVan rdf:type rdfs:Class .

ex:PassengerVehicle rdfs:subClassOf ex:MotorVehicle .
ex:Van rdfs:subClassOf ex:MotorVehicle .
ex:Truck rdfs:subClassOf ex:MotorVehicle .

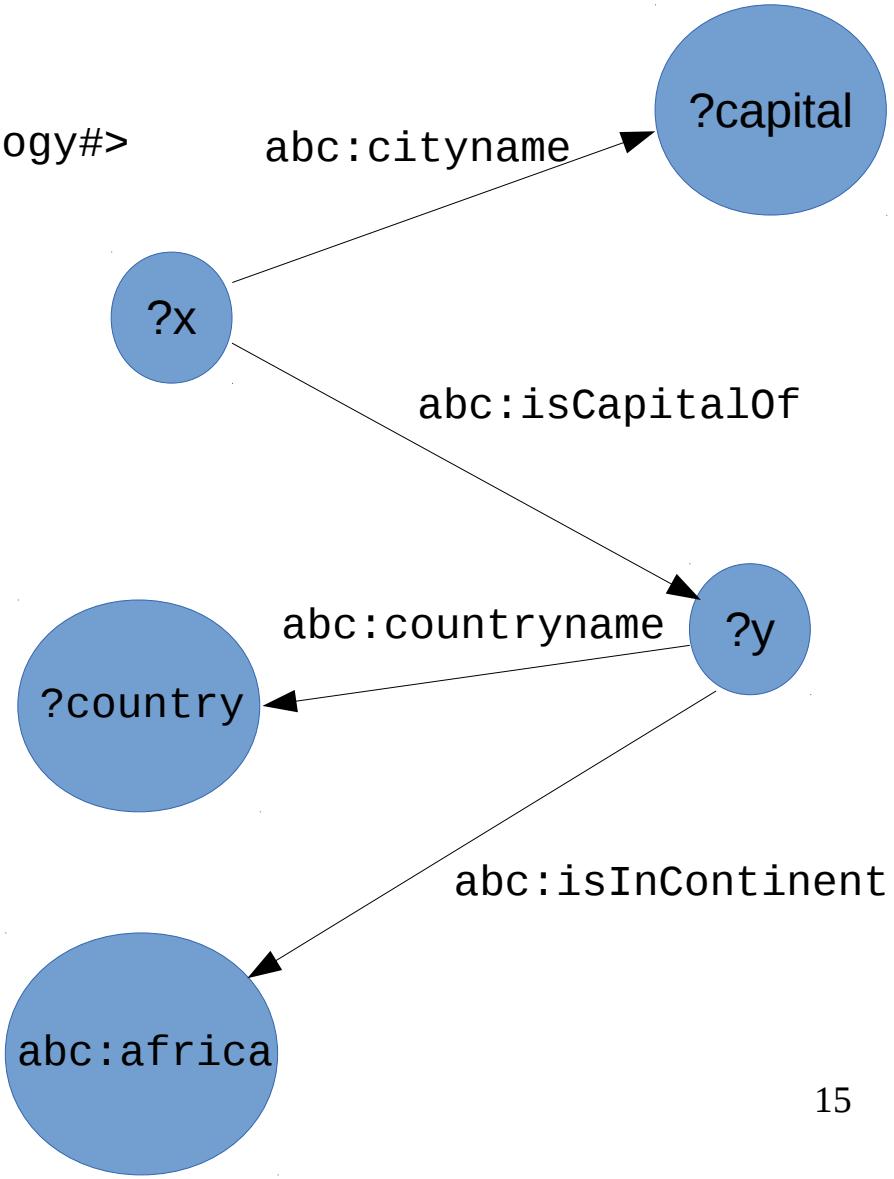
ex:MiniVan rdfs:subClassOf ex:Van .
ex:MiniVan rdfs:subClassOf ex:PassengerVehicle .

SPARQL

- SPARQL Protocol and RDF Query Language
- SPARQL query
  - Graph can include variables in place of constants
- Operations
  - JOIN (natural, left-join)
  - AND, FILTER, UNION, OPTIONAL
- Commercial DBMS-s
  - Implement RDF and SPARQL

Example SPARQL query

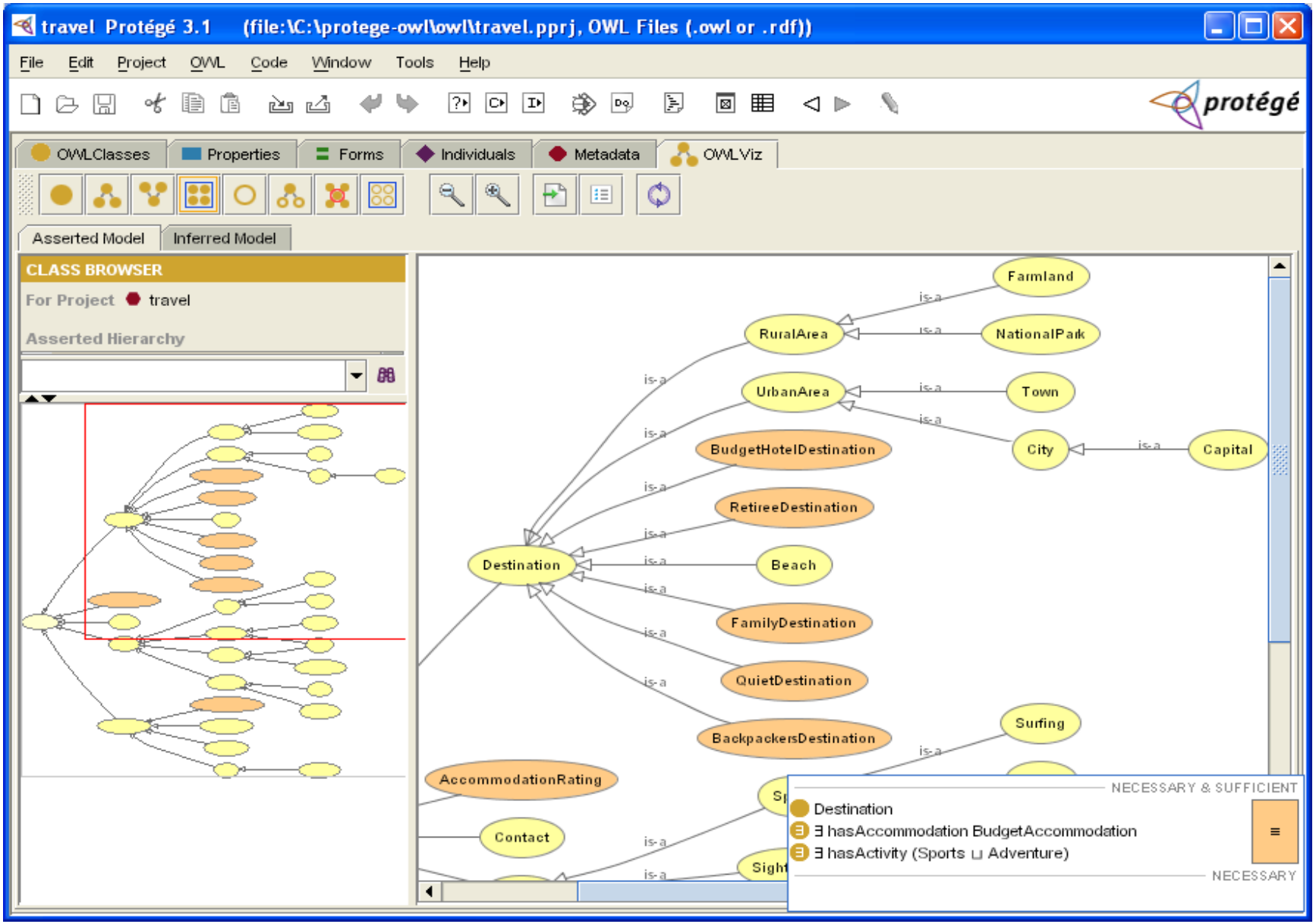
```
PREFIX
   abc: <http://mynamespace.com/exampleOntology#>
SELECT ?capital ?country
WHERE { ?x abc:cityname ?capital.
        ?y abc:countryname ?country.
        ?x abc:isCapitalOf ?y.
        ?y abc:isInContinent abc:africa. }
```



Logic - OWL

- Ontology language
  - Knowledge representation + Logic
- Based on description logic
  - Fragments of predicate calculus
  - Hierarchy of DL languages
- OWL reasoners
  - FaCT++, HermiT, RacerPro, Pellet, ...

Protégé


travel Protégé 3.1 (file:\C:\protege-owl\owl\travel.pprj, OWL Files (.owl or .rdf))
File Edit Project OWL Code Window Tools Help
protégé
OWLClasses Properties Forms Individuals Metadata OWLViz
Asserted Model Inferred Model
CLASS BROWSER
For Project travel
Asserted Hierarchy
Farmland
RuralArea
NationalPark
UrbanArea
Town
City
Capital
BudgetHotelDestination
RetireeDestination
Destination
Beach
FamilyDestination
QuietDestination
BackpackersDestination
Surfing
AccommodationRating
Contact
is-a
NECESSARY & SUFFICIENT
Destination
∃ hasAccommodation BudgetAccommodation
∃ hasActivity (Sports ⊔ Adventure)
NECESSARY

Popular graph databases on Web

Terminology

- Linked data
  - Linked Open Data
- Open data
- Graph databases
- Knowledge bases
- Knowledge graphs

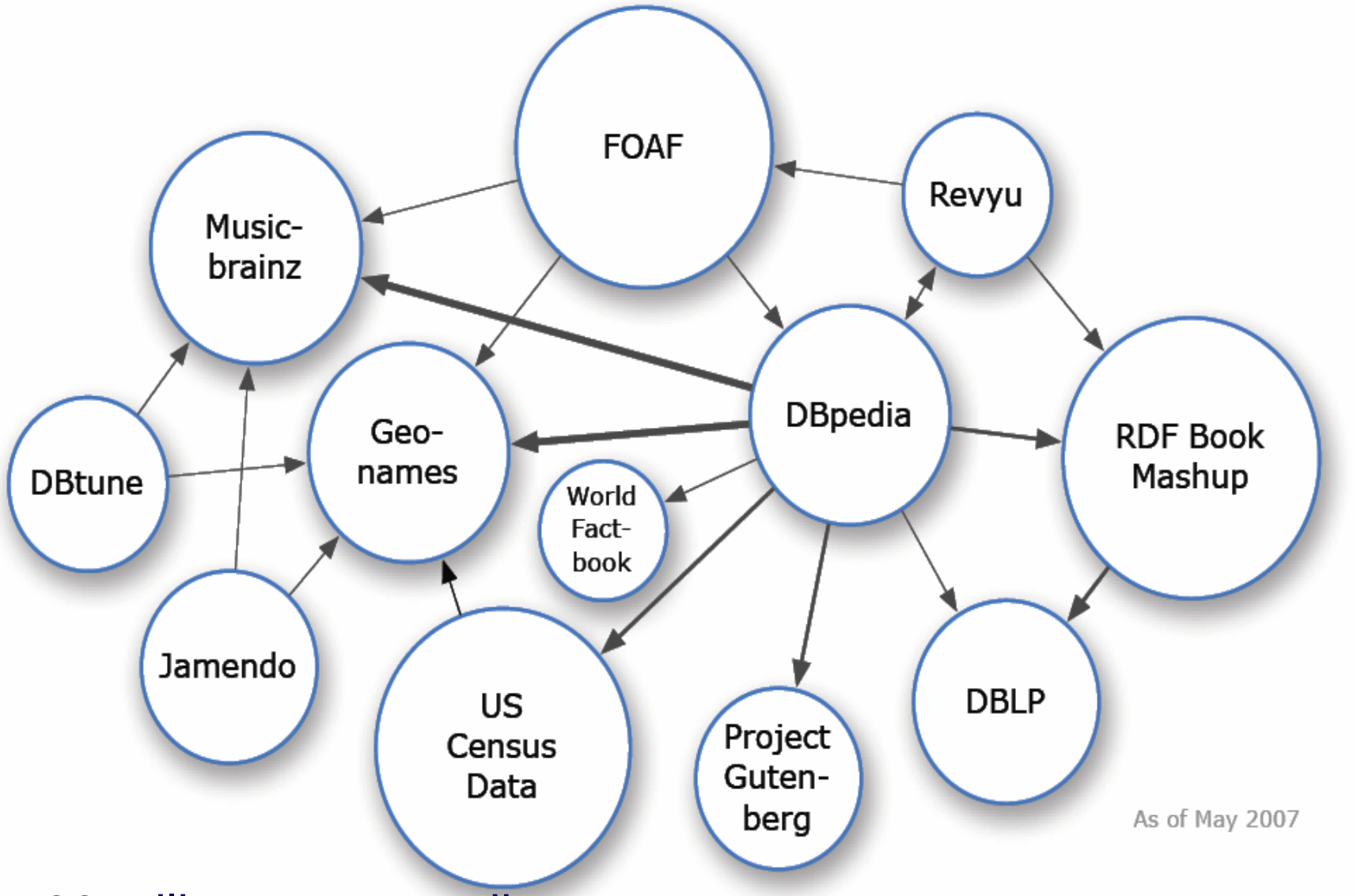
Wordnet

- Princeton's large lexical database of English.
  - Cognitve synonims: synsets
    - 117,000 synsets
  - Synsets are linked by:
    - conceptual-semantic relationships, and
    - lexical relationships.
    - Include definitions of synsets.
  - Main relationships:
    - Synonymy, hyponymy (ISA), meronymy (part-whole), antonymy

Linked Open Data

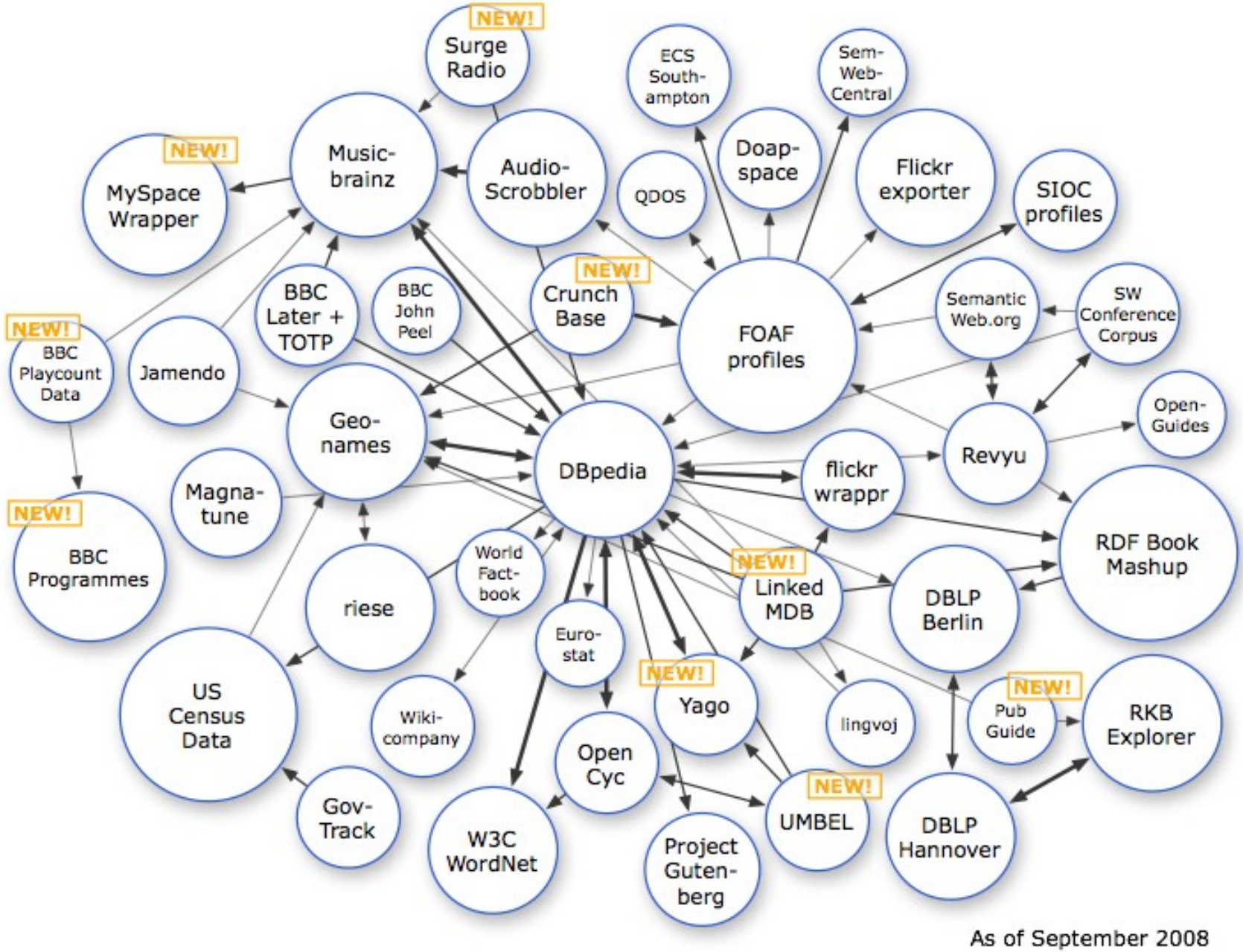
- Datasets are represented in RDF
  - Wikipedia, Wikibooks, Geonames, MusicBrainz, WordNet, DBLP bibliography
- Number of triples: 33 Giga (10^9) (2011)
- Governments:
  - USA, UK, Japan, Austria, Belgium, France, Germany, ...
- Active community
  - http://en.wikipedia.org/wiki/Open_Data
  - http://www.w3.org/LOD

LOD zbirke na spletu: Maj 2007

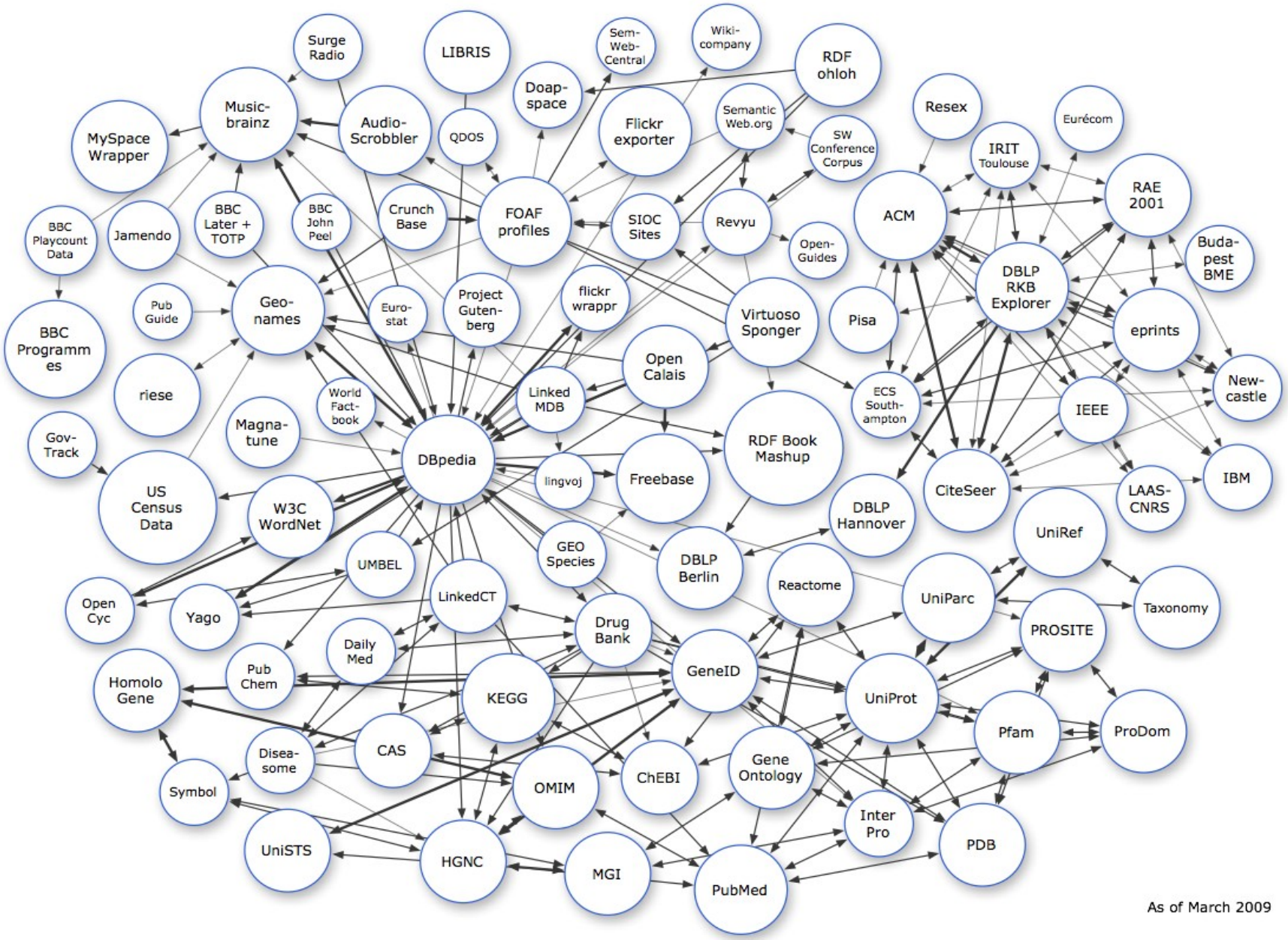


- Nad 500 miljonov RDF trojic
- Okoli 120,000 RDF povezav med pod.viri

LOD zbirke na spletu: September 2008

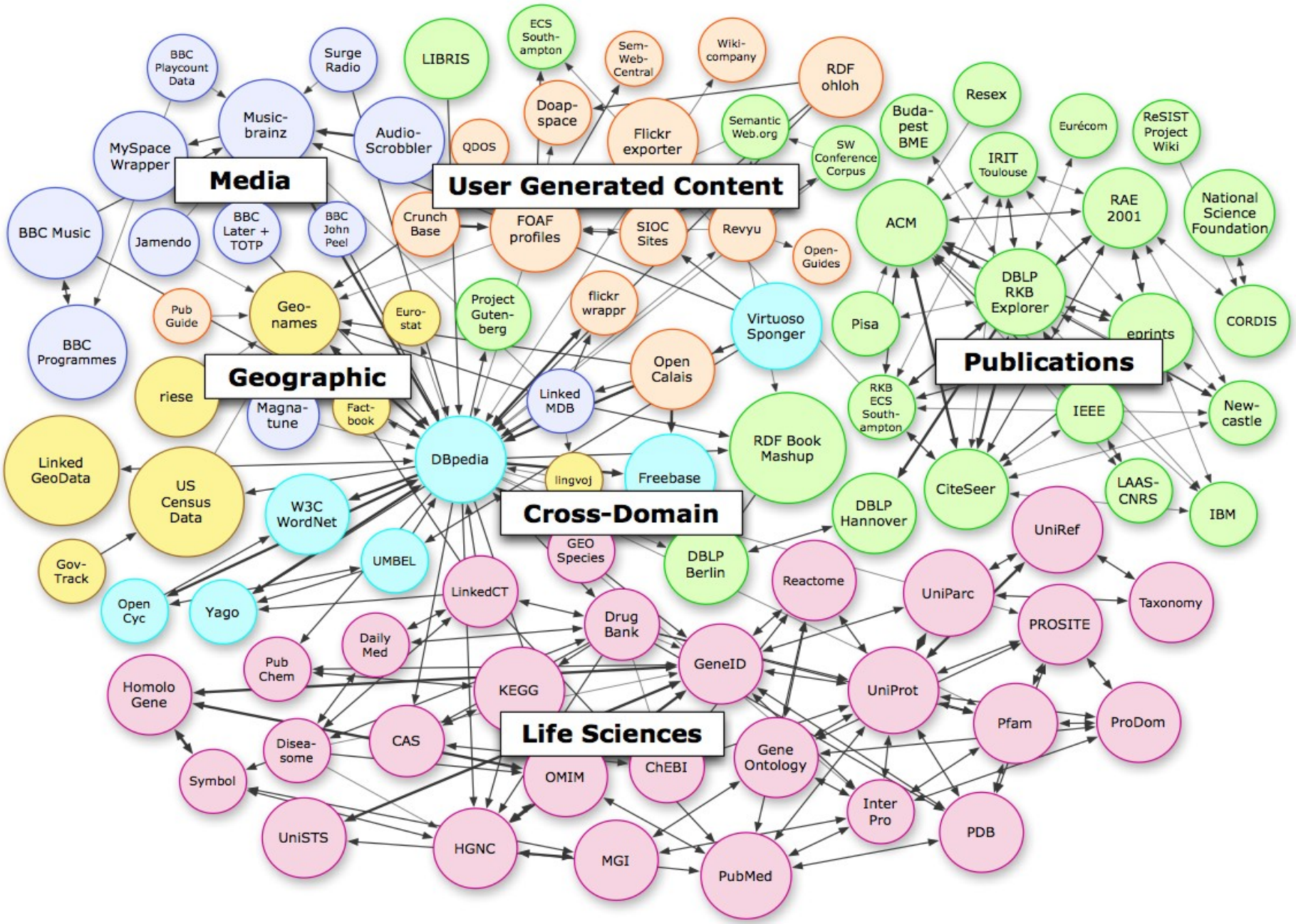


LOD zbirke na spletu: Marec 2009

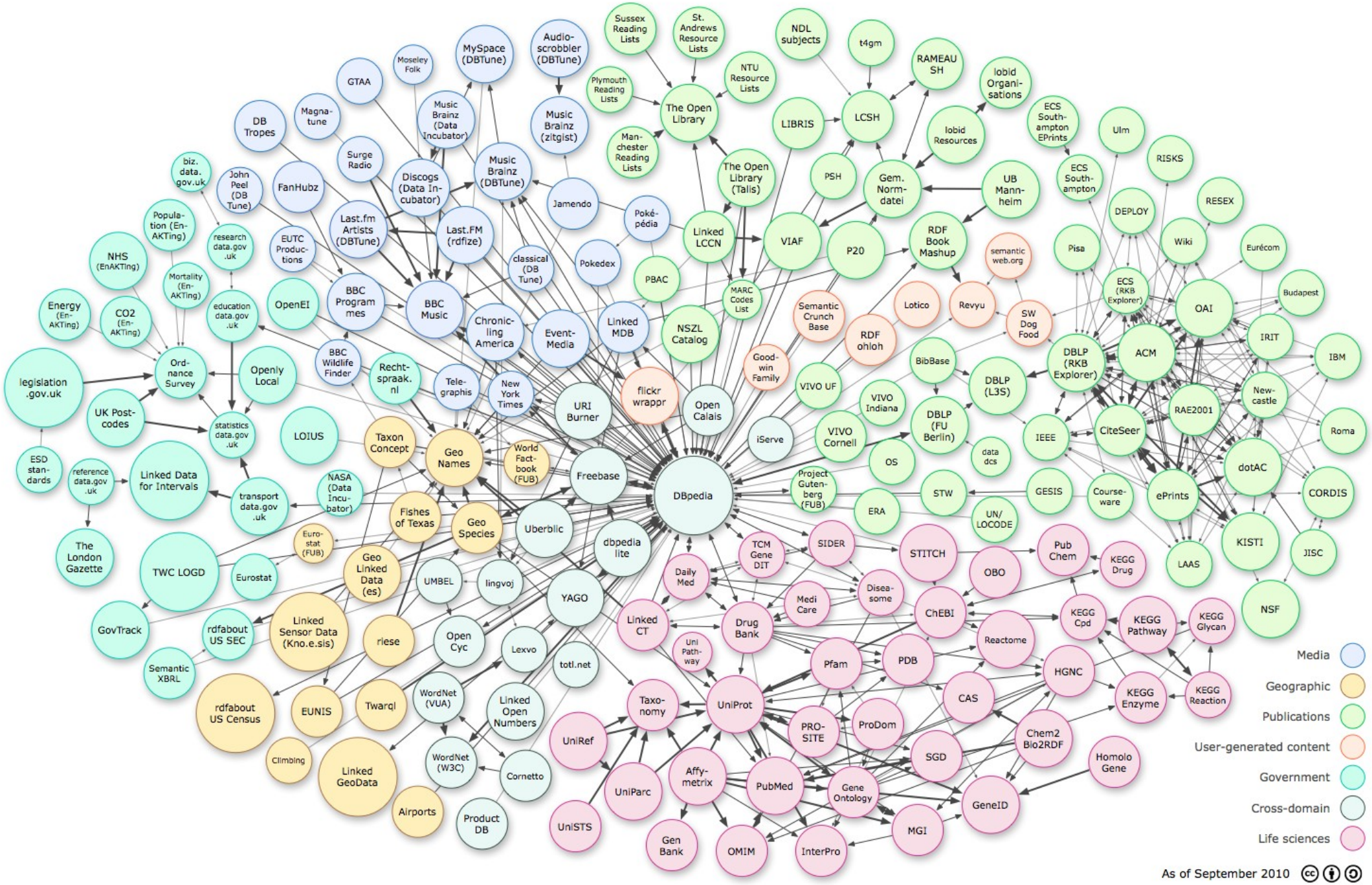


As of March 2009

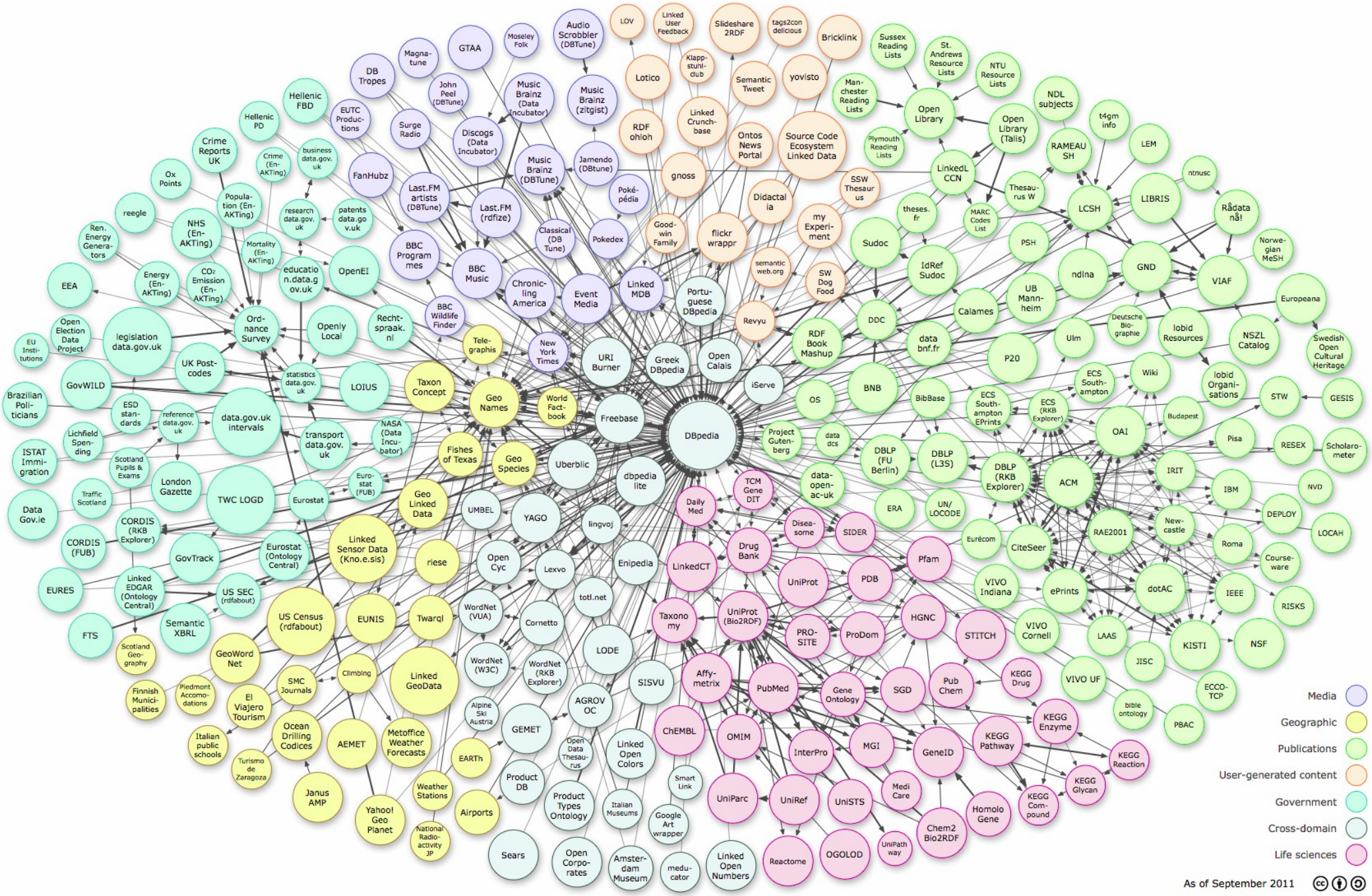
LOD zbirke na spletu: Julij 2009



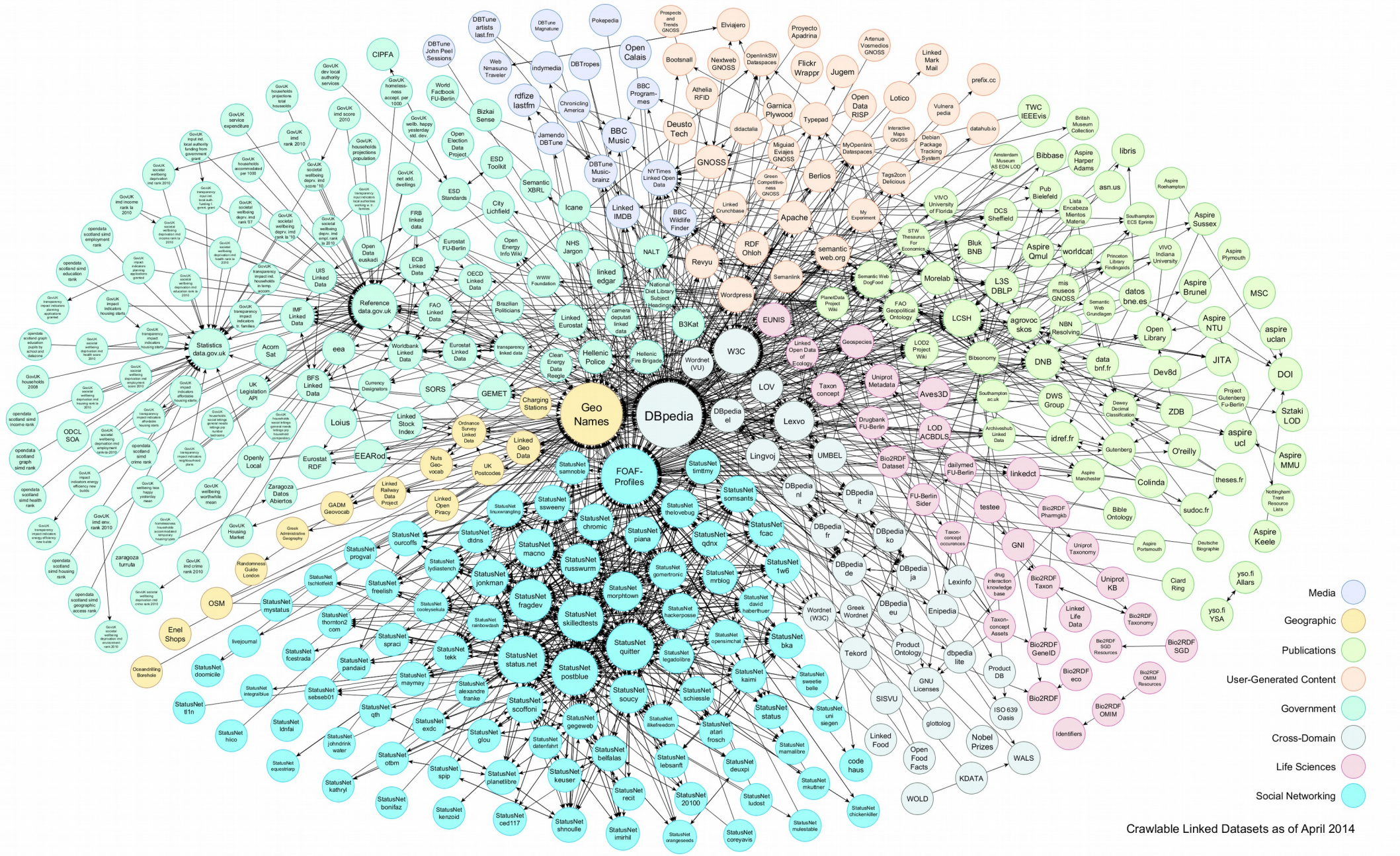
LOD zbirke na spletu: Sept 2010



LOD zbirke na spletu: Sept 2011



LOD Cloud, 2014



LOD Cloud, 2018

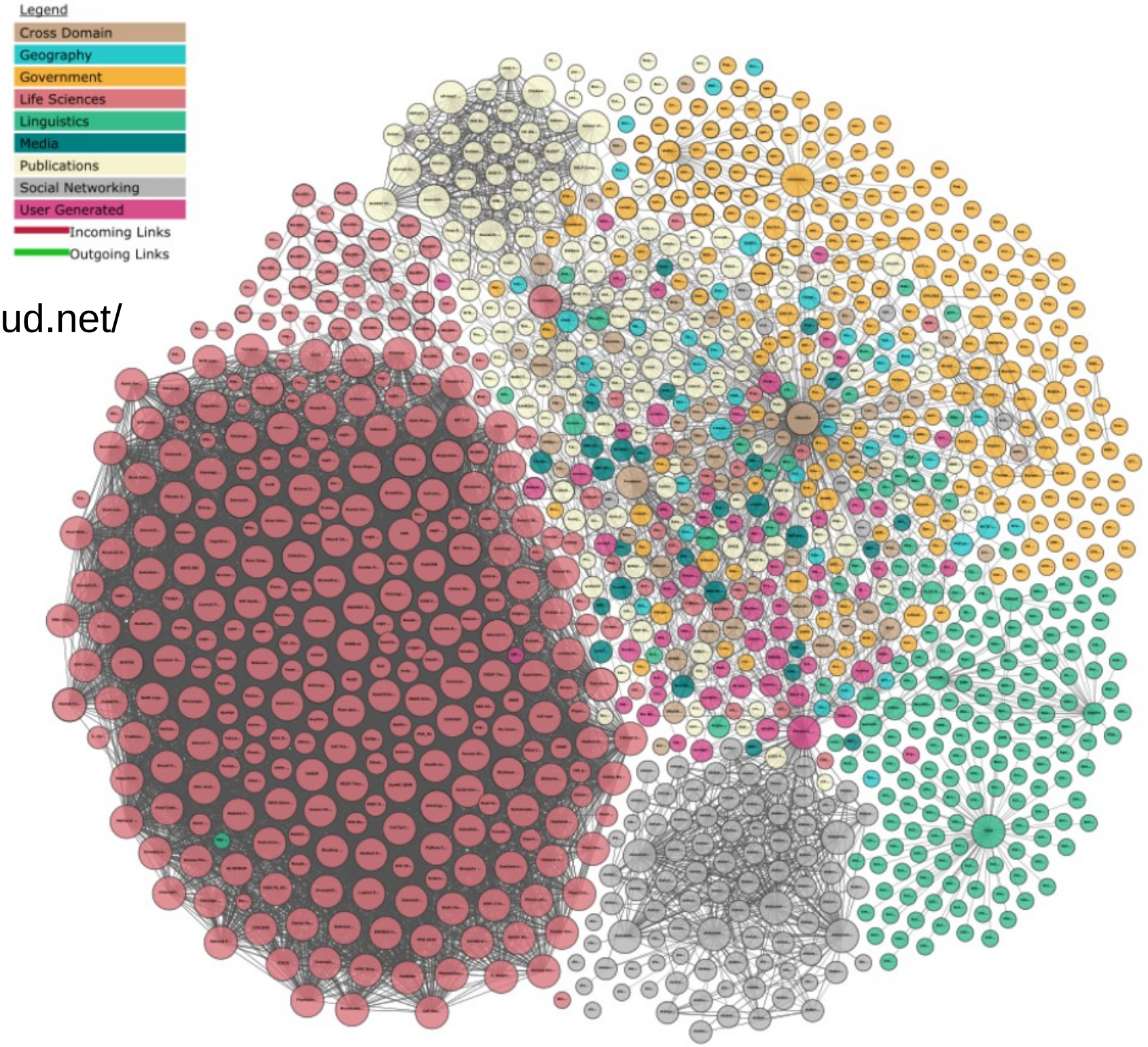
Basic Statistics

Criterion	Average	Min	Max	Median	Total
Triples	67,544.15	0	47,054,407	337.0	192,230,648
Entities	18,105.28	0	9,319,918	80.0	54,225,309
Literals	30,137.45	0	31,476,008	166.0	90,261,655
Blanks	3,554.83	0	3,565,513	0.0	10,646,711
Blanks as subject	1,742.85	0	1,910,532	0.0	5,219,831
Blanks as object	1,812.01	0	3,564,789	0.0	5,426,969
Subclasses	1.6	0	2,000	0.0	4,779
Typed subjects	7,387.12	0	6,990,722	39.0	22,124,421
Labeled subjects	1,219.97	0	1,440,595	0.0	3,653,811
Average properties per entity	4.98	0.0	91.16	3.71	
Average string length typed	13.28	0.0	436.0	0.0	
Average string length untyped	391.77	0.0	181,576.0	10.0	
Average class hierarchy depth	3.24	1	9	None	
Links	15,379.59	0	13,252,430	57.0	46,061,873
Average property hierarchy depth	1.5	1	3	None	
Vocabularies	4.27	1	18	3.0	12,110
Classes	4.36	1	330	3.0	10,384
Properties	17.58	1	254	16.0	49,916

9960 datasets

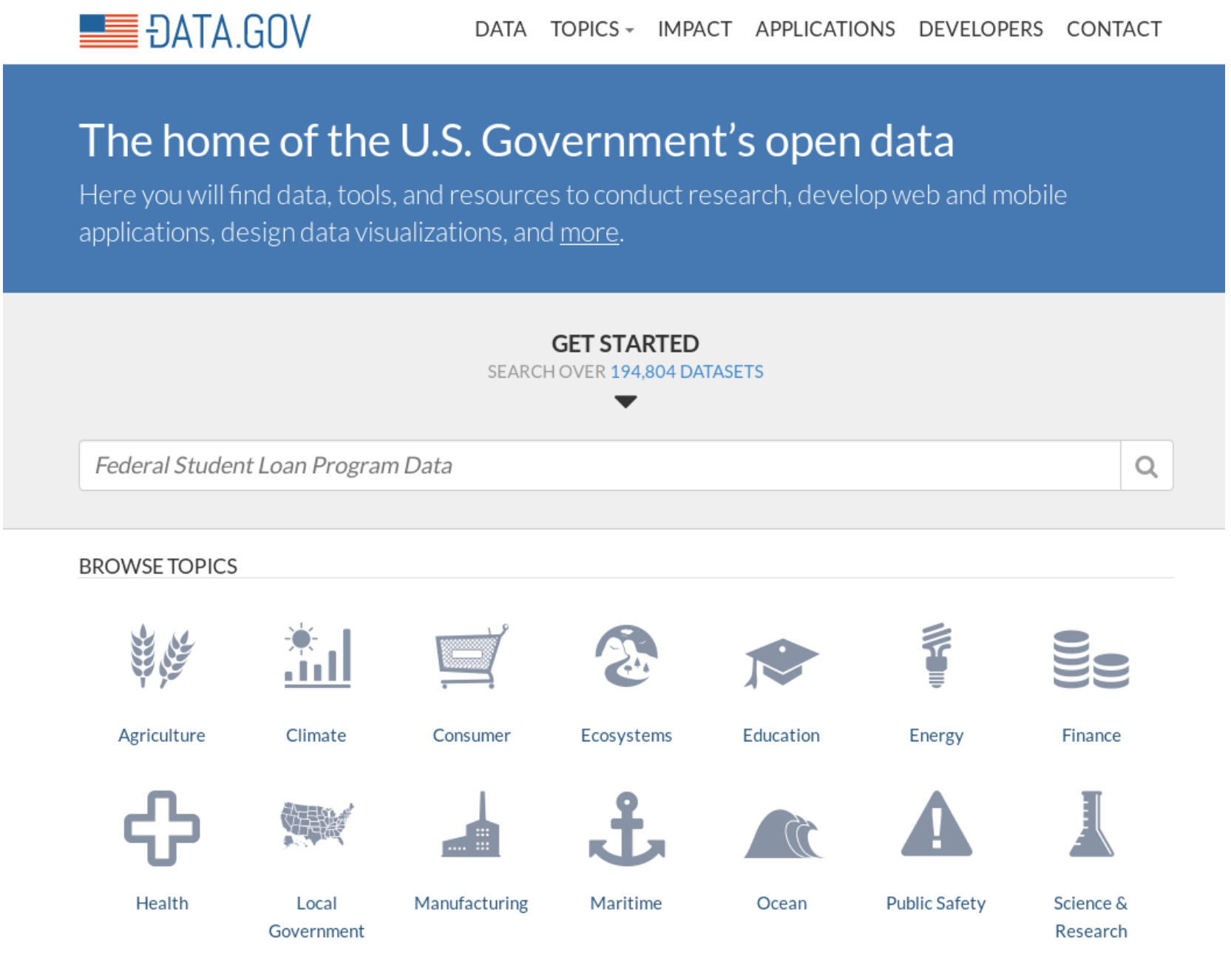
149,423,660,620 triples from **2973 datasets** (192,230,648 triples from **2838 dumps**, 149,231,429,972 from **151 datasets via SPARQL**)

Problems with **6971 datasets** (70.1%): **6578 dumps having errors**, **393 SPARQL endpoints with errors**

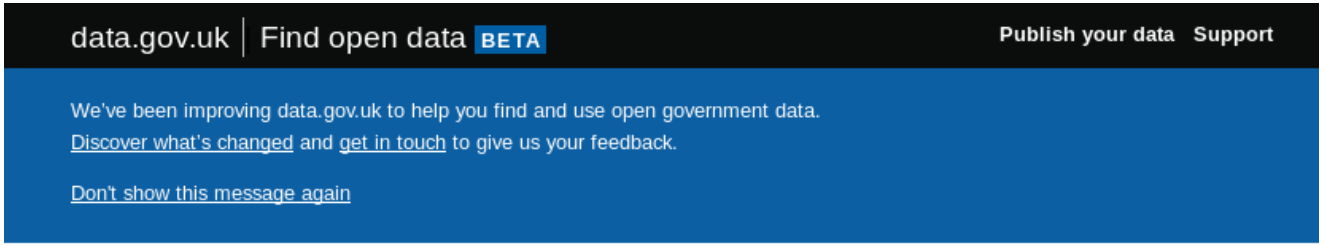


http://lod-cloud.net/

Open Data


DATA.GOV
DATA
TOPICS
IMPACT
APPLICATIONS
DEVELOPERS
CONTACT
The home of the U.S. Government's open data
Here you will find data, tools, and resources to conduct research, develop web and mobile applications, design data visualizations, and more.
GET STARTED
SEARCH OVER 194,804 DATASETS
Federal Student Loan Program Data
BROWSE TOPICS
Agriculture
Climate
Consumer
Ecosystems
Education
Energy
Finance
Health
Local Government
Manufacturing
Maritime
Ocean
Public Safety
Science & Research

Open Data



Find open data

Find data published by central government, local authorities and public bodies to help you build products and services

Business and economy
Small businesses, industry, imports, exports and trade

Crime and justice
Courts, police, prison, offenders, borders and immigration

Defence
Armed forces, health and safety, search and rescue

Education
Students, training, qualifications and the National Curriculum

Environment
Weather, flooding, rivers, air quality, geology and agriculture

Government
Staff numbers and pay, local councillors and department business plans

Government spending
Includes all payments by government departments over £25,000

Health
Includes smoking, drugs, alcohol, medicine performance and hospitals

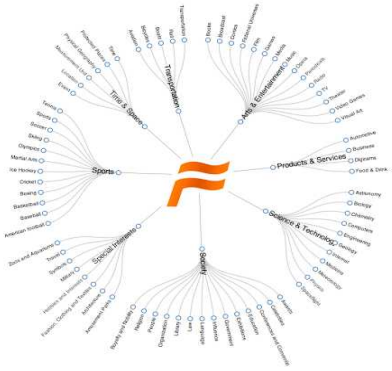
Mapping
Addresses, boundaries, land ownership, aerial photographs, seabed and land terrain

Society
Employment, benefits, household finances, poverty and population

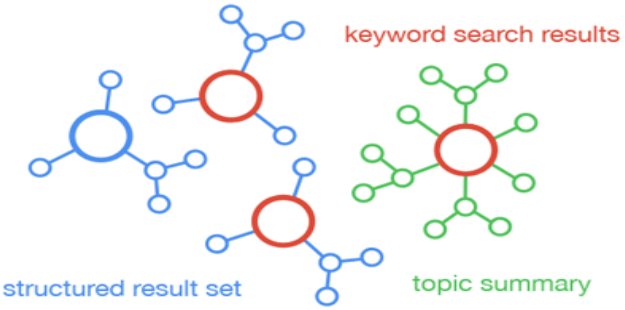
Towns and cities
Includes housing, urban planning, leisure, waste and energy, consumption

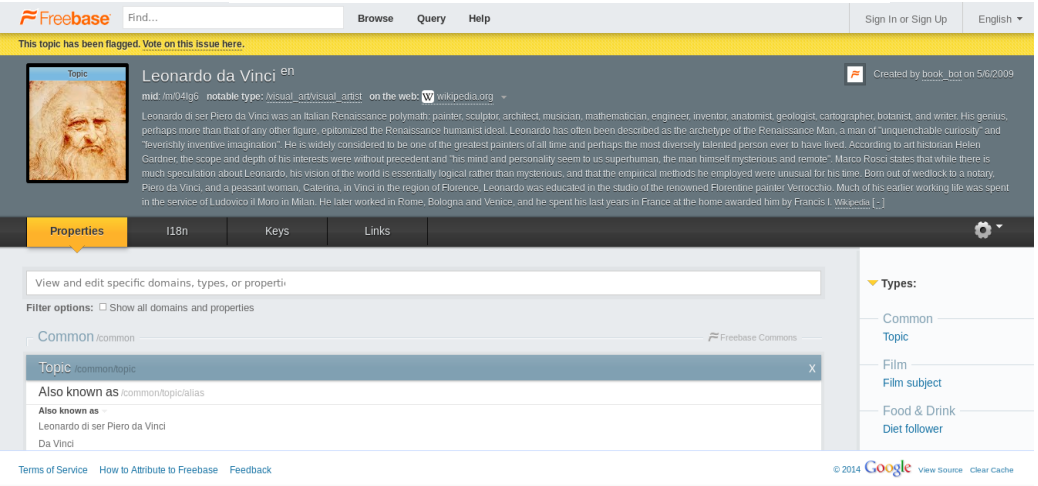
Transport
Airports, roads, freight, electric vehicles, parking, buses and footpaths

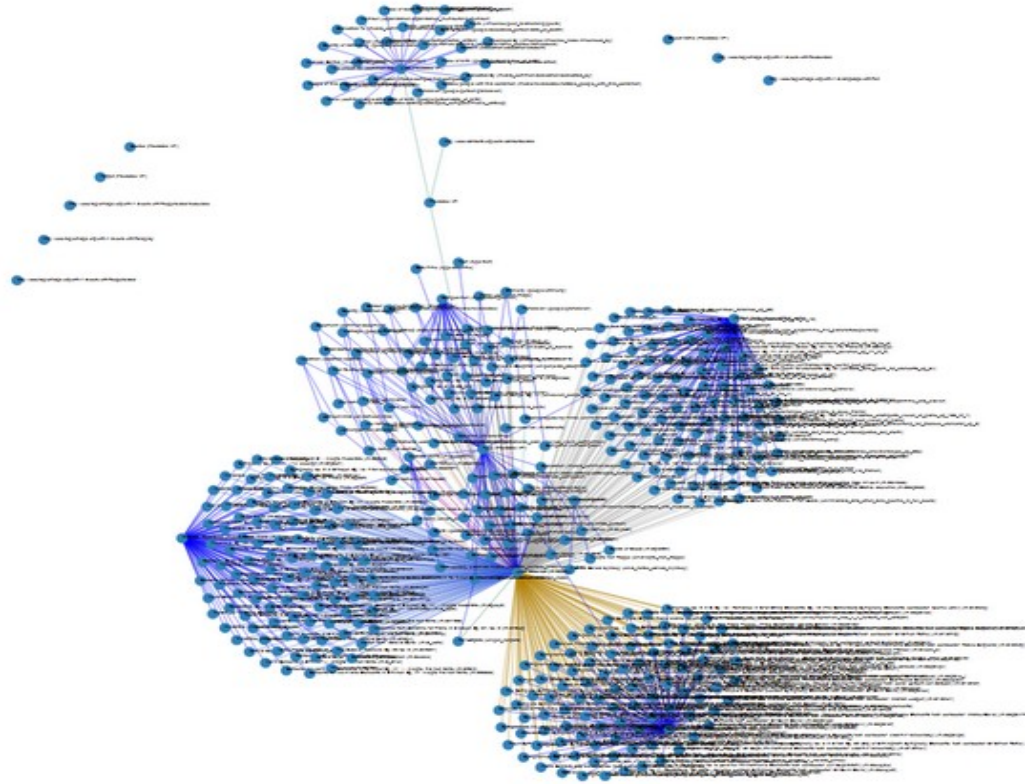
Freebase



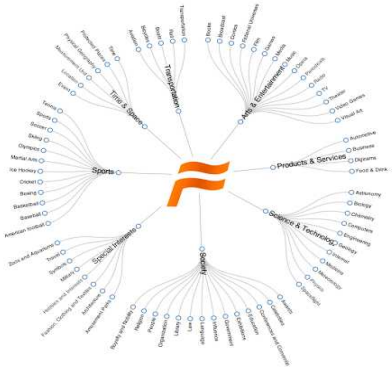
- Free, knowledge graph:
  - people, places and things,
  - 3,041,722,635 facts, 49,947,845 topics
- Semantic search engines are here !







Freebase



- Based on graphs:
  - nodes, links, types, properties, namespaces
- Google use of Freebase
  - Knowledge graph
  - Words become concepts
  - Semantic questions
  - Semantic associations
  - Browsing knowledge
  - Knowledge engine
- Available in RDF

Knowledge graph

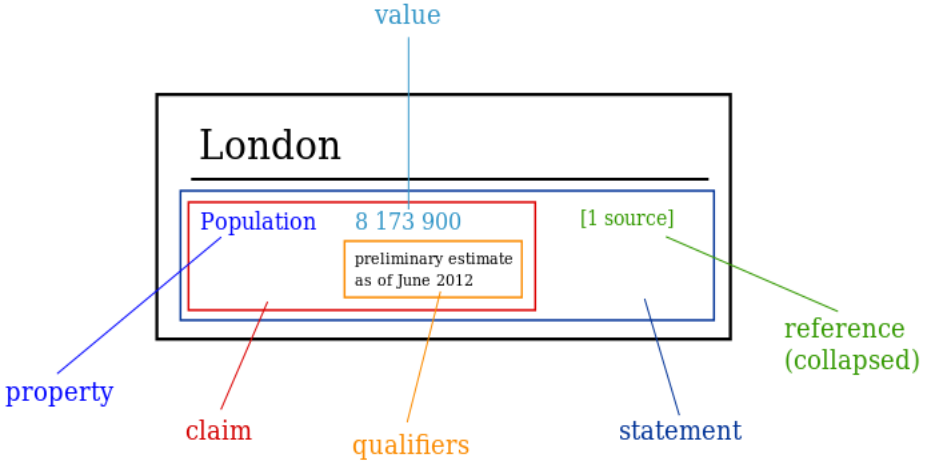
- Google’s Knowledge Graph
  - 70 billion facts, oct 2016
  - Box to the right of search results, since 2012
  - Google Assistant and Google Home voice queries
- Knowledge Vault, Google, 2014
  - Initiative to succeed the capabilities of the Knowledge Graph
    - … to deal with facts, automatically gathering and merging information from across the Internet into a knowledge base capable of answering direct questions, such as "Where was Madonna born"

YAGO

- 10 Mega (10^6) concepts
  - 120M facts about these entities
  - Max Planc Institute, Informatik
  - Accuracy of 95%
- Includes:
  - Wikipedia, WordNet, GeoNames
  - Links Wordnet to Wikipedia taxonomy (350K concepts)
  - Anchored in time and space

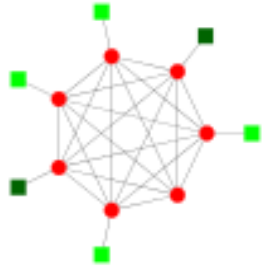
Wikidata

- Free knowledge base with 46,769,977 items
  - 14,913,910 - 2015
- Collecting structured data
- Properties of
  - person, organization,

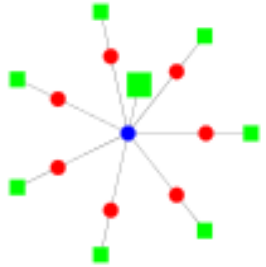


Former system:
interwiki links
between all languages

Phase 1 of Wikidata:
links of all languages
to one central point

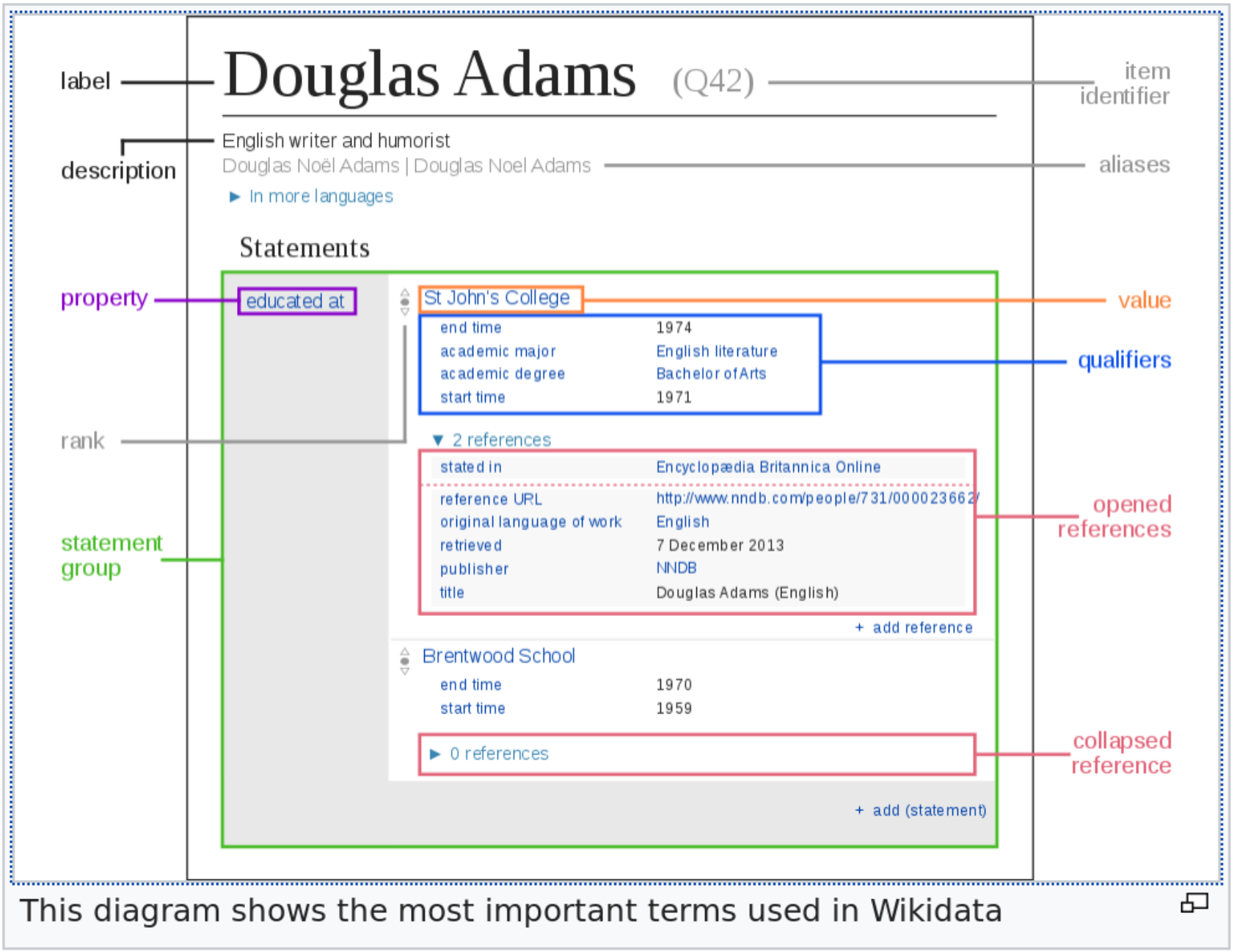


Former system:
Independent information about
infoboxes in all languages



Phase 2 of Wikidata:
Information for infoboxes
of all languages
on one central point

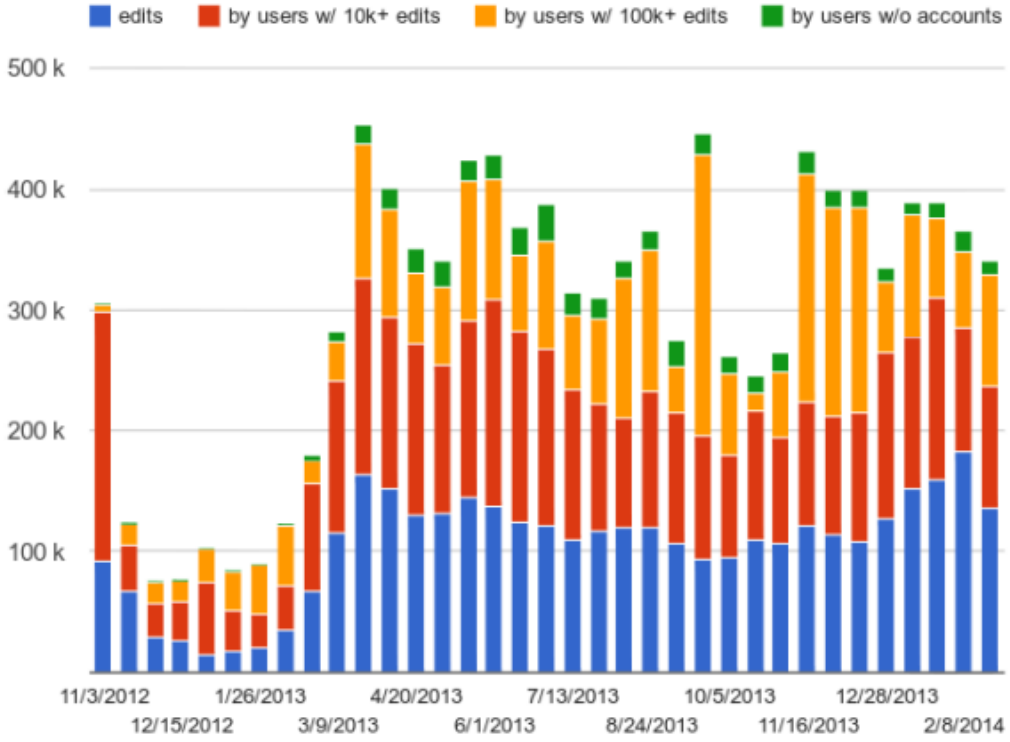
Wikidata

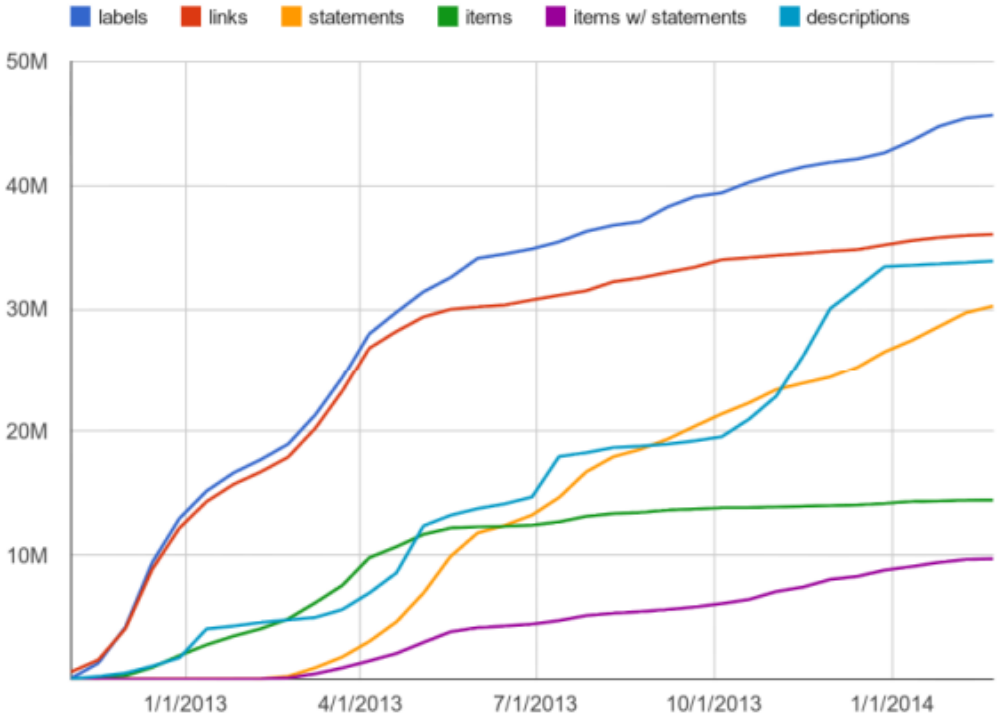


This diagram shows the most important terms used in Wikidata

Wikidata

- 2015 - Free knowledge base with 14,550,852 items



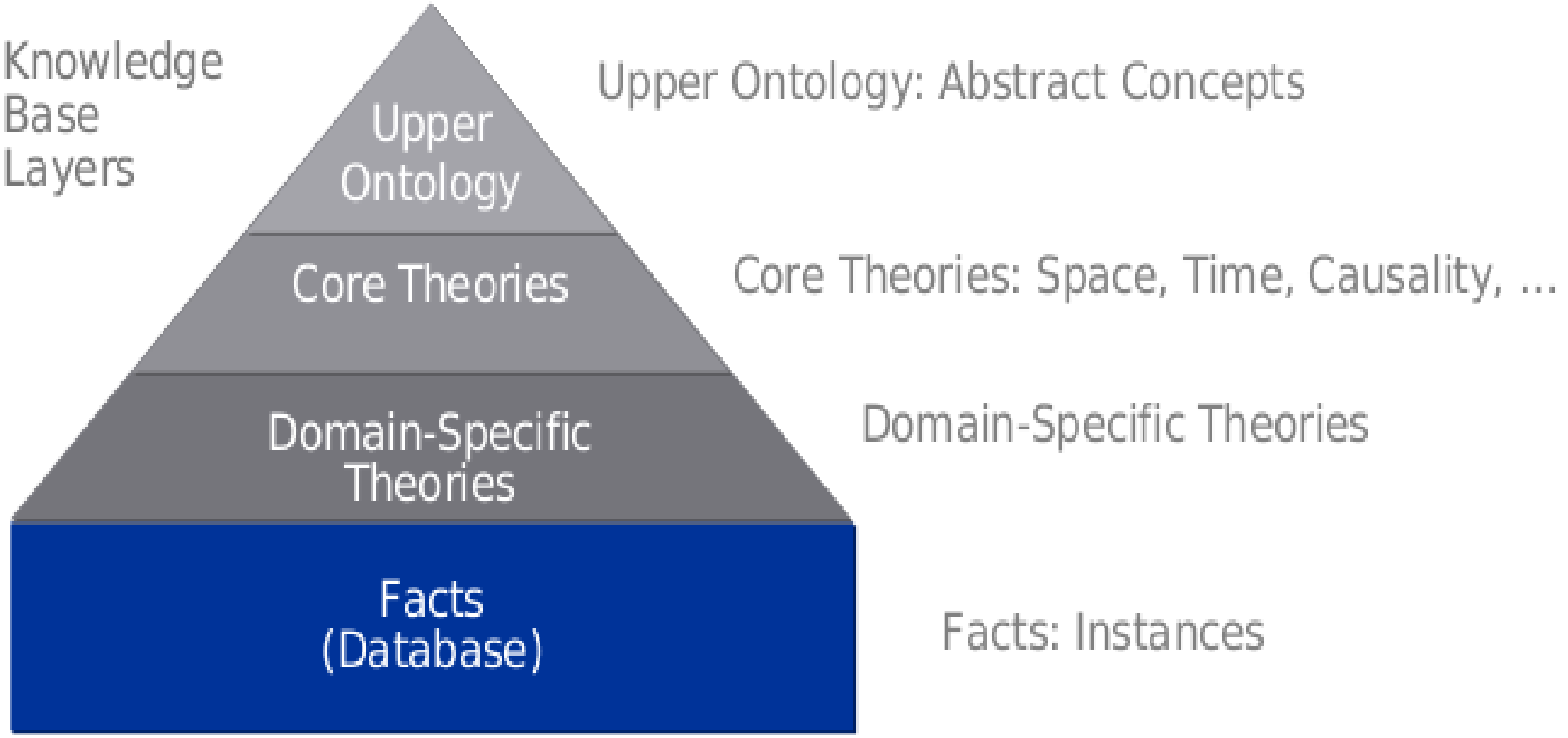


Cycorp®
Home of smarter solutions

Cyc - knowledge base

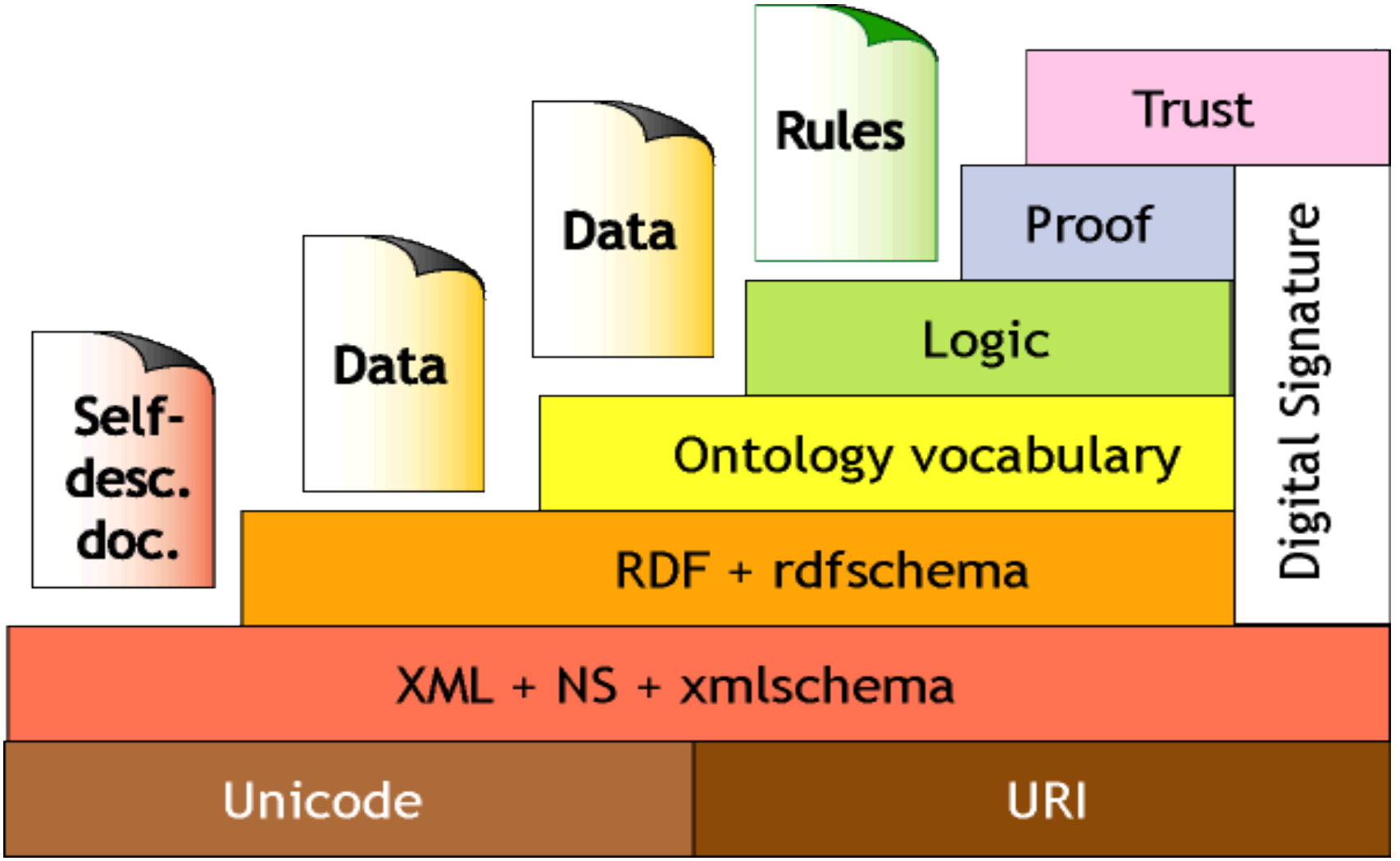
- Knowledge base
  - Doug Lenat
  - Conceptual networks (ontologies)
  - Higher ontology, basic theories, specific theories
  - Predefined semantic relationships
  - 500.000 terms, including about 17.000 types of relations, and about 7.000.000 assertions relating these terms
- Common sense reasoner
  - Based on predicate calculus
  - Rule-based reasoning

Cyc


Knowledge Base Layers
Upper Ontology
Core Theories
Domain-Specific Theories
Facts (Database)
Upper Ontology: Abstract Concepts
Core Theories: Space, Time, Causality, ...
Domain-Specific Theories
Facts: Instances

Semantic Web

Stolp semantičnega spleta



Sintaksa in semantika

- Sintaksa: struktura podatkov
- Semantika: pomen podatkov
- Dva pogoja potrebna za skupno delo:
  - Skupna sintaksa: programi lahko razčlenjujejo podatke
  - Skupen način razumevanja pomena: programi lahko uporabljajo podatke.

XML

- XML: eXtensible Mark-up Language
- XML dokumenti so napisani z uporabniško definiranimi značkami
- Oznake so uporabljene za izražanje “pomena” delov podatkov

XML - osnovni gradniki

- Elementi
  - Značke – opisujejo podatkovni objekt
- Atributi (prilastki)
  - Opisujejo lastnosti objektov - elementov
- Entitete
  - Deli skupnega teksta
- Komentarji

XML - primer

```
<?xml version="1.0"?>
<!-- File Name: Inventory.xml -->

<inventory>
        <book genre=“comp” id=“b724”>
                <title>Database Management Systems</title>
                <author><firstname>Raghu</firstname>
                    <lastname>Ramakrishnan</lastname>
                </author>
                <publisher>McGraw Hill</publisher>
                <year>2000</year>
        </book>
        …
</inventory>
```

XML - osnovna pravila

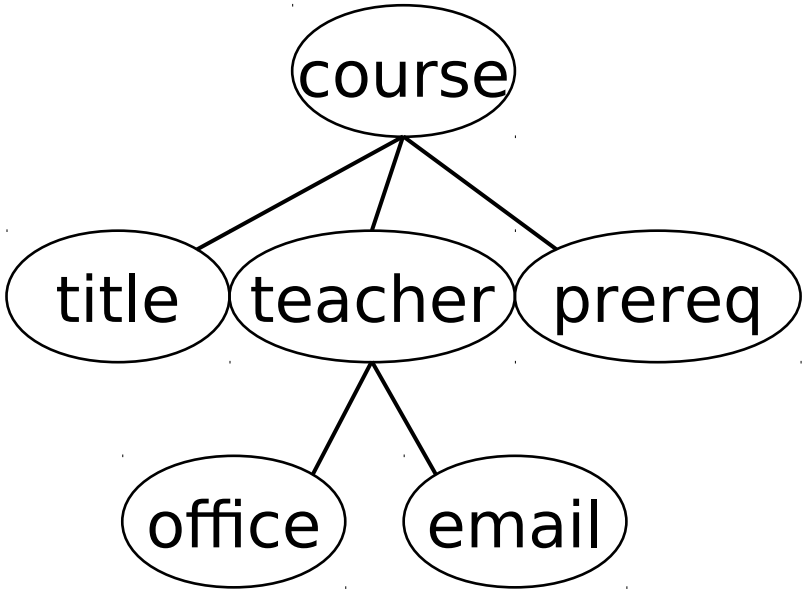
- Vsebuje deklaracijo dokumenta
- Vsaka začetna značka mora imeti tudi pripadajočo končno značko
- Strukture morajo biti pravilno gnezdene
- Vrednosti atributov so navedene v narekovajih
- Na najvišjem nivoju je lahko le en element – korenski element (*S*)

XML

- XML: dokument = označeno drevo
- vozlišče = oznaka + atributi/vrednosti + vsebina

```
<course date="...">
  <title>...</title>
  <teacher>
    <office>...</office>
    <email>...</email>
  </teacher>
  <room>...</room>
  <prereq>...</prereq>
</course>
```

=



DTD

- Document Type Descriptor
- Podedovan od SGML
- Podoben DB shemi, čeprav ni zares …
- BNF slovnica (prim. kontekstno neodvisni jezik)
- Definicija elementov specifičnega XML jezika

DTD - primer

<!element book (title,author*,publisher,year) >

<!element title #PCDATA >

<!element publisher #PCDATA >

<!element year #PCDATA >

<!element author (firstname,lastname,address?,age?) >

<!attlist book id ID #required >

<!attlist book genre CDATA #required >

…

XML

- XML Schema = slovnica za opis dreves in podatkovnih tipov
- Lahko uporabljamo XML za predstavitev semantike?

XML in pomen

```
<Predator>
      …
</Predator>
```

- Predator: srednje višine, velika vztrajnost, letalna naprava.
- Predator: tisti, ki terorizira, razbija in uničuje še posebno za neko korist.
- Predator: organizem, ki živi na osnovi ulova drugih organizmov.
- ...

Omejitve pri opisu pomena

- XML ne opisuje:
  1. Besednjak specifičen za neko domeno
  2. Ontološki primitivi za modeliranje podatkov
- Zahteva dogovor glede 1 in 2
- Uporabna rešitev za lokalne projekte:
  - Agenti v manjšem stabilnem okolju
  - Strani na manjšem in stabilnem intranet
- Ni primerno za predstavitev spletnih virov

RDF

- RDF je podatkovni model
  - Model je neodvisen od domene, aplikacije in se lahko internacionalizira
  - Model lahko vidimo kot usmerjen, označen graf ali kot objekten model (objekti/atributi/vrednosti)
- RDF podatkovni model je abstrakten, konceptualni nivo neodvisen od XML
  - XML je lahko sintaksa za RDF in ne del RDF
  - RDF podatki se lahko sploh ne pojavijo v XML obliki

Asociativni model

- Splošen model

- RDF – Resource Description Language
  - RDF je podatkovni model !
  - Opisi temeljijo na trojicah: asociacije
- Lahko opišemo kompleksne vrednosti
  - Kolekcije, vreče, sestavljene objekte, …

RDF model

RDF model = množica RDF trojic

trojica = izraz (stavek)

`(subjekt, predikiat, objekt)`

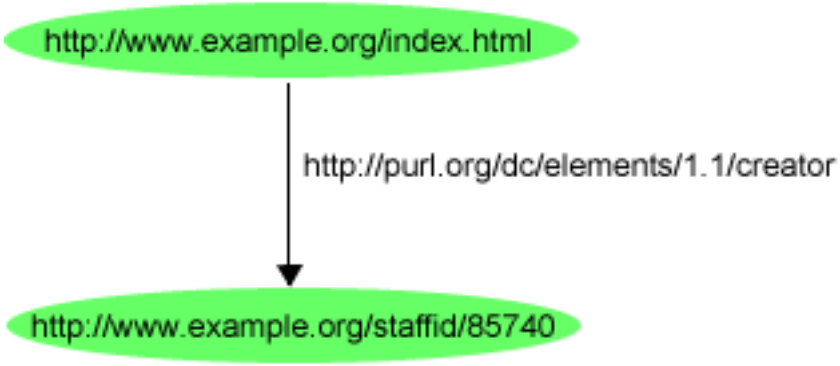
- subjekt = vir
- predikat = lastnost (vira)
- objekt = vrednost (lastnosti)

`http://www.example.org/index.html` has a **creator** whose value is **`John Smith`**

RDF model

- Subjekt: `http://www.example.org/index.html`
- Predikat: `http://purl.org/dc/elements/1.1/creator`
- Objekt: `http://www.example.org/staffid/85740`

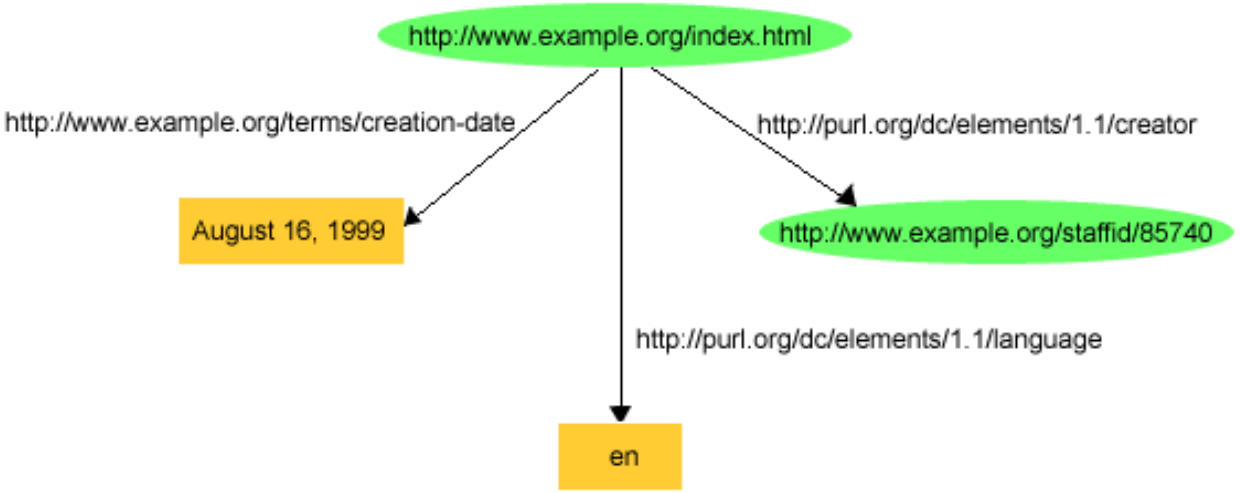
- Vsaka trojica ustreza eni povezavi v grafu



Primer RDF opisa

```
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/creator> <http://www.example.org/staffid/85740> .
<http://www.example.org/index.html> <http://www.example.org/terms/creation-date> "August 16, 1999" .
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/language> "en" .
```

- Uporabljamo polne poti !
- Koncepti so ovali
- Literali so pravokotniki



RDF sintaksa

RDF model = graf z označenimi povezavami
= množica trojic

- Grafična notacija (graf)
- Notacija (neformalna) na osnovi trojic, npr.:

```
(subject, predicate, object)
```

- Notacija:
  - N3, TVS
  - Turtle, TriG, N-Triples
  - RDF/XML, RDF/JSON

Prostori imen

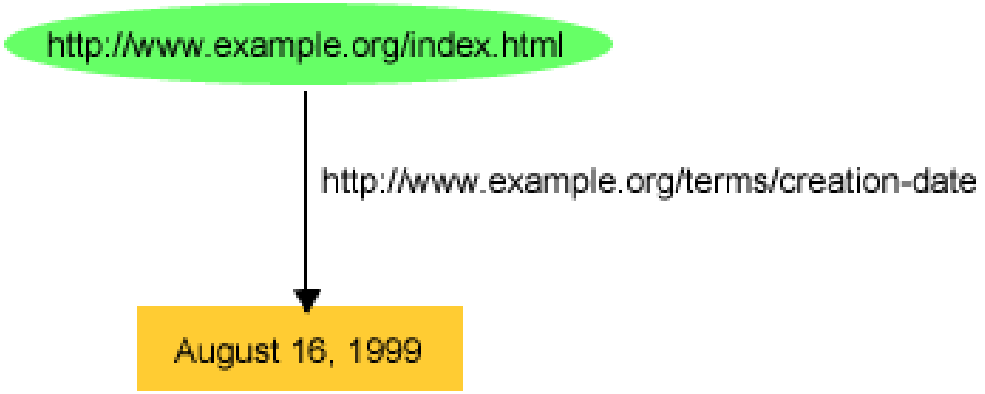
- Uporabljali bomo okrajšana imena URL naslovov
- Imena so definirana na določenih URL-jih
- Imena, ki jih uporabljamo so predpone

```
prefix rdf:, namespace URI: http://www.w3.org/1999/02/22-rdf-syntax-ns#
prefix rdfs:, namespace URI: http://www.w3.org/2000/01/rdf-schema#
prefix dc:, namespace URI: http://purl.org/dc/elements/1.1/
prefix owl:, namespace URI: http://www.w3.org/2002/07/owl#
prefix ex:, namespace URI: http://www.example.org/ (or http://www.example.com/)
prefix xsd:, namespace URI: http://www.w3.org/2001/XMLSchema#
```

- Poglejmo si zdaj okrajšan zapis primerov

En stavek

```
ex:index.html exterms:creation-date "August 16, 1999" .
```




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">

<rdf:Description rdf:about="http://www.example.org/index.html">
     <exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>

</rdf:RDF>
```

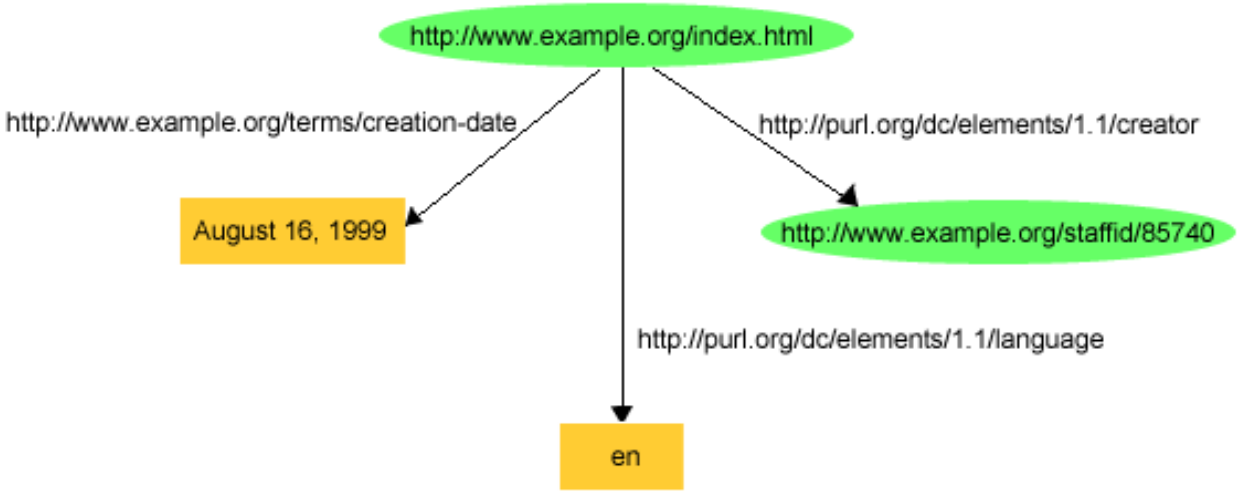
Dva stavka

```
ex:index.html exterms:creation-date "August 16, 1999" .
ex:index.html dc:language "en" .
```

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:exterms="http://www.example.org/terms/">

<rdf:Description rdf:about="http://www.example.org/index.html">
<exterms:creation-date>August 16, 1999</exterms:creation-date>
</rdf:Description>

<rdf:Description rdf:about="http://www.example.org/index.html">
<dc:language>en</dc:language>
</rdf:Description>

</rdf:RDF>
```

```
ex:index.html dc:creator exstaff:85740 .
ex:index.html exterms:creation-date "August 16, 1999" .
ex:index.html dc:language "en" .
```

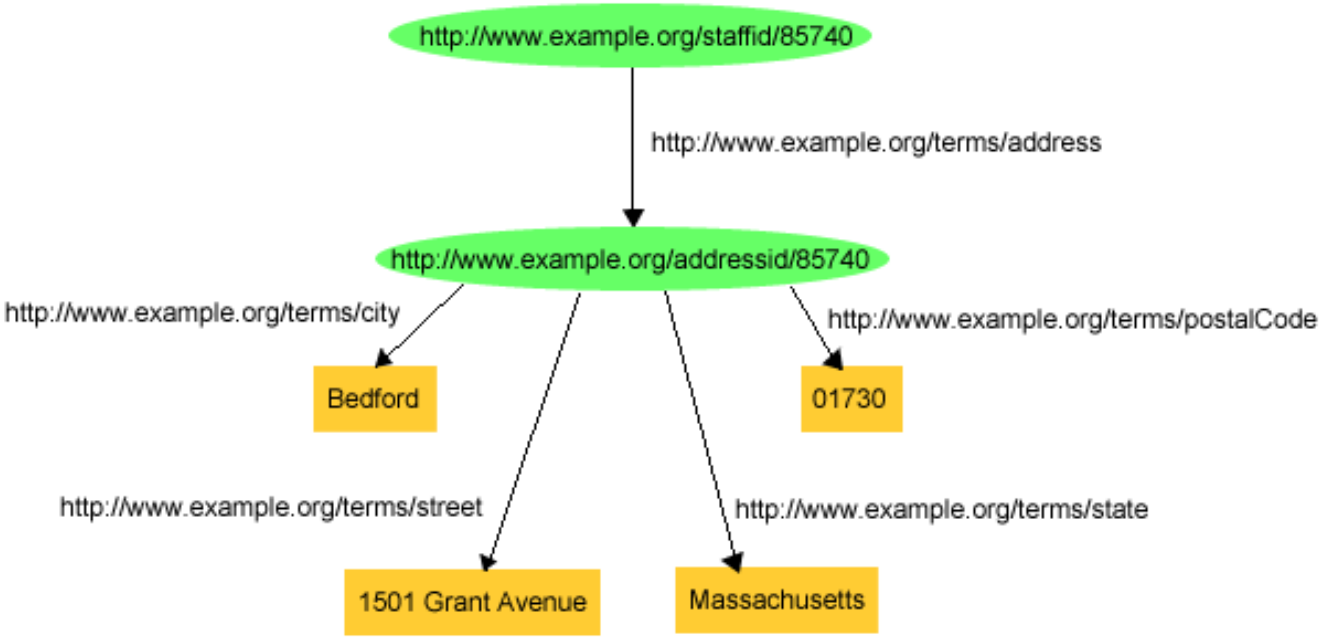
Trije stavki




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns:dc="http://purl.org/dc/elements/1.1/"
                 xmlns:exterms="http://www.example.org/terms/">

<rdf:Description rdf:about="http://www.example.org/index.html">
   <exterms:creation-date>August 16, 1999</exterms:creation-date>
   <dc:language>en</dc:language>
   <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
</rdf:Description>

</rdf:RDF>
```

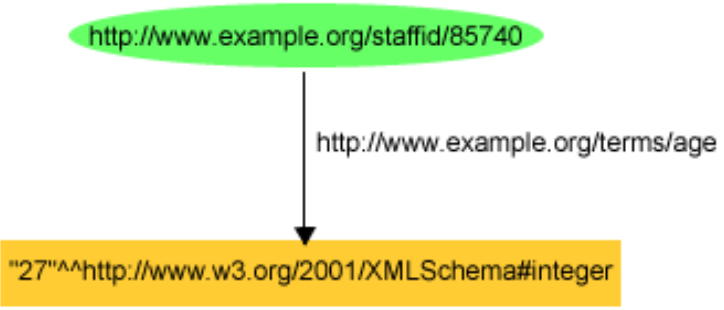
Strukturirane vrednosti

```
exstaff:85740 exterms:address exaddressid:85740 .
exaddressid:85740 exterms:street "1501 Grant Avenue" .
exaddressid:85740 exterms:city "Bedford" .
exaddressid:85740 exterms:state "Massachusetts" .
exaddressid:85740 exterms:postalCode "01730" .
```



Tip skalarja

- Skalarju definiramo tip
- 27 pomeni celo število in ne znaka “2” in “7”

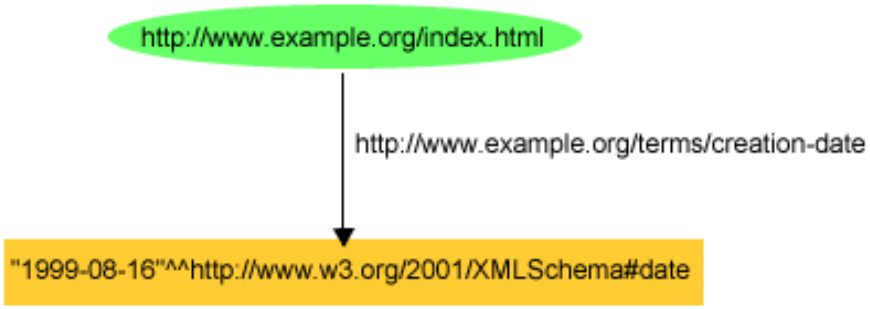



```
exstaff:85740 exterms:age "27"^^xsd:integer .
```

```
<http://www.example.org/staffid/85740> <http://www.example.org/terms/age>
    "27"^^<http://www.w3.org/2001/XMLSchema#integer> .
```

- RDF nima vgrajenih tipov
- Tipi so definirani izven RDF: datatype URI
- XML Schema enostavni tipi
- `xsd:integer, xsd:float, xsd:double, xsd:boolean, xsd:boolean, xsd: date, …`

Tip skalarja




```
ex:index.html exterms:creation-date "1999-08-16"^^xsd:date .
```

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:exterms="http://www.example.org/terms/">

<rdf:Description rdf:about="http://www.example.org/index.html">
   <exterms:creation-date rdf:datatype=
   "http://www.w3.org/2001/XMLSchema#date">1999-08-16 </exterms:creation-date>
</rdf:Description>

</rdf:RDF>
```

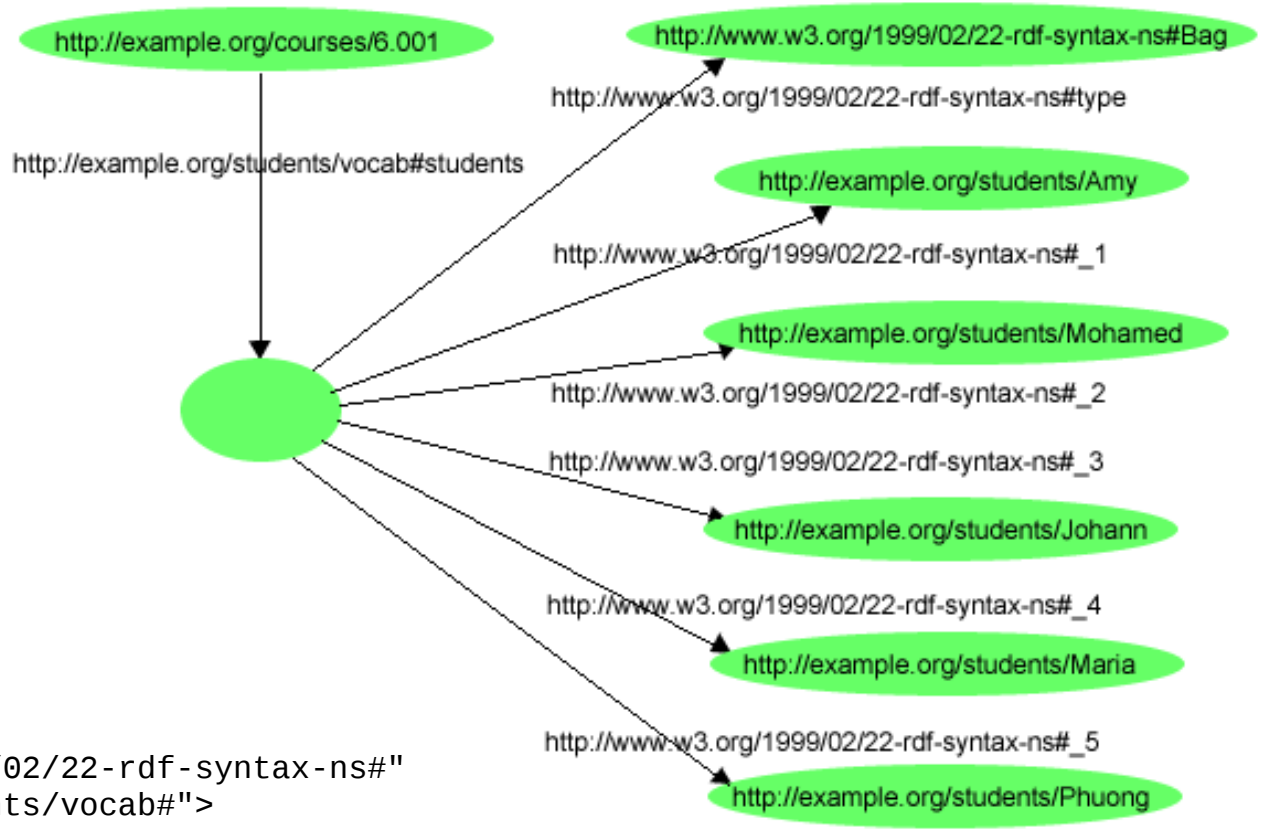
Uporaba ID

- Do zdaj smo uporabljali `rdf:about` za specifikacijo objekta, ki ga opisujemo
  - Opisovanje virov
- Včasih želimo opisati objekt, ki ga ni moč opisati z URI referenco
  - Primer: katalog objektov na določenem naslovu
- Podobno XML ID: unikaten znotraj osnovnega URI

Kontejnerji

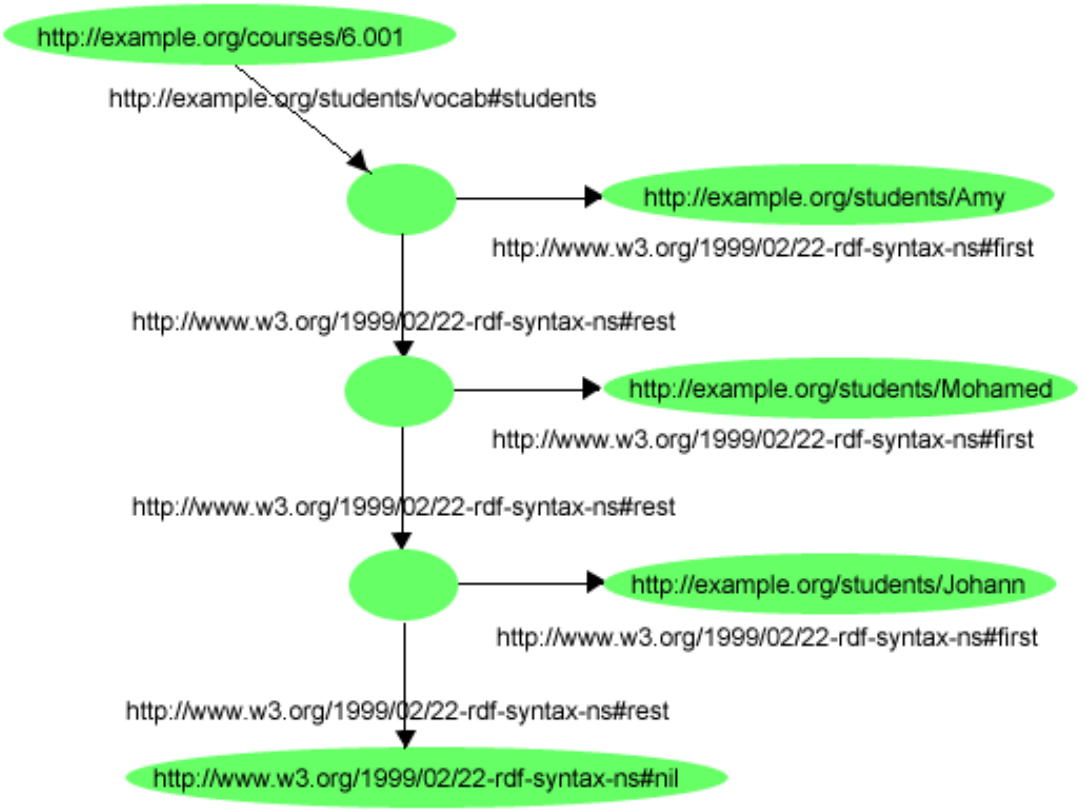
- Kontejnerji omogočajo grupiranje virov (ali besed)
- Napišemo lahko izjave o kontejnerju (kot celota) ali individualno o njegovih članih
- Poznamo različne tipe kontejnerjev
  - Vreča (rdf:Bag) – neurejena kolekcija
  - Zaporedje (rdf:Seq) – urejena kolekcija (= “sekvenca”)
  - Alternative (rdf:Alt) – predstavlja alternative
- Dvojniki so dovoljeni (ni mehanizma za zagotavljanje unikatnosti vrednosti)

Vreča




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

<rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students>
     <rdf:Bag>
             <rdf:li rdf:resource="http://example.org/students/Amy"/>
             <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
             <rdf:li rdf:resource="http://example.org/students/Johann"/>
                  <rdf:li rdf:resource="http://example.org/students/Maria"/>
                  <rdf:li rdf:resource="http://example.org/students/Phuong"/>
     </rdf:Bag>
    </s:students>
</rdf:Description>

</rdf:RDF>
```

Kolekcija




```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:s="http://example.org/students/vocab#">

<rdf:Description rdf:about="http://example.org/courses/6.001">
   <s:students rdf:parseType="Collection">
    <rdf:Description rdf:about="http://example.org/students/Amy"/>
    <rdf:Description rdf:about="http://example.org/students/Mohamed"/>
    <rdf:Description rdf:about="http://example.org/students/Johann"/>
   </s:students>
</rdf:Description>

</rdf:RDF>
```

RDF shema

RDFS = RDF shema

- Definira majhen slovar za RDF:
  - `rdfs:class, rdfs:subClassOf, rdfs:type`
  - `rdfs:property, rdfs:subPropertyOf`
  - `rdfs:domain, rdfs:range`
- Ustreza množici RDF predikatov:

  ⇒ meta-nivo

  ⇒ poseben vnaprej definiran pomen

Razredi

ex:MotorVehicle rdf:type rdfs:Class .
ex:PassengerVehicle rdf:type rdfs:Class .
ex:Van rdf:type rdfs:Class .
ex:Truck rdf:type rdfs:Class .
ex:MiniVan rdf:type rdfs:Class .

ex:PassengerVehicle rdfs:subClassOf ex:MotorVehicle .
ex:Van rdfs:subClassOf ex:MotorVehicle .
ex:Truck rdfs:subClassOf ex:MotorVehicle .

ex:MiniVan rdfs:subClassOf ex:Van .
ex:MiniVan rdfs:subClassOf ex:PassengerVehicle .

XML/RDF hierarchija vozil

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd http://www.w3.org/2001/XMLSchema#">]>

<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xml:base="http://example.org/schemas/vehicles">

<rdf:Description rdf:ID="MotorVehicle">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="PassengerVehicle">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Truck">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
    <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>
```

RDFS lastnosti

Slovar za lastnosti:

- RDF razred:
  - `rdfs:Property` – definicija lastnosti

- RDF schema lastnosti:
  - `rdfs:domain` – definicija prve komponente lastnosti
  - `rdfs:range` – definicija druge komponente lastnosti
  - `rdfs:subPropertyOf` - izraža ISA relacijo med lastnostmi

Opisovanje lastnosti

- Opisujemo lastnosti razredov podobno kot razrede v programskih jezikih
- Vse lastnosti so definirane kot primerki `rdf:Property`
- Ni omejitev glede številnosti lastnosti!
- Pri definiciji zaloge vrednosti lahko uporabljamo tudi osnovne tipe npr. `xsd:integer`

Primer

- Lastnost `ex:author` ima domeno `ex:Book` in zalogo vrednosti razred `ex:Person`

```
ex:Book rdf:type rdfs:Class .
ex:Person rdf:type rdfs:Class .
ex:author rdf:type rdf:Property .
ex:author rdfs:domain ex:Book .
ex:author rdfs:range ex:Person .
```

- Lastnost ima lahko več kot eno zalogo vrednosti:

```
ex:hasMother rdfs:range ex:Female .
ex:hasMother rdfs:range ex:Person .
```

Lastnosti vozil

- Avto je registrirala oseba:

```
<rdf:Property rdf:ID="registeredTo">
   <rdfs:domain rdf:resource="#MotorVehicle"/>
   <rdfs:range rdf:resource="#Person"/>
</rdf:Property>
```

- Razdalja med sedeži:

```
<rdf:Property rdf:ID="rearSeatLegRoom">
   <rdfs:domain rdf:resource="#PassengerVehicle"/>
   <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>
```

Pod-lastnosti

- Primarni voznik je podlastnost voznika:

```
ex:driver rdf:type rdf:Property .
ex:primaryDriver rdf:type rdf:Property .
ex:primaryDriver rdfs:subPropertyOf ex:driver .

<rdf:Property rdf:ID="driver">
   <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
   <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

Linked data and applications

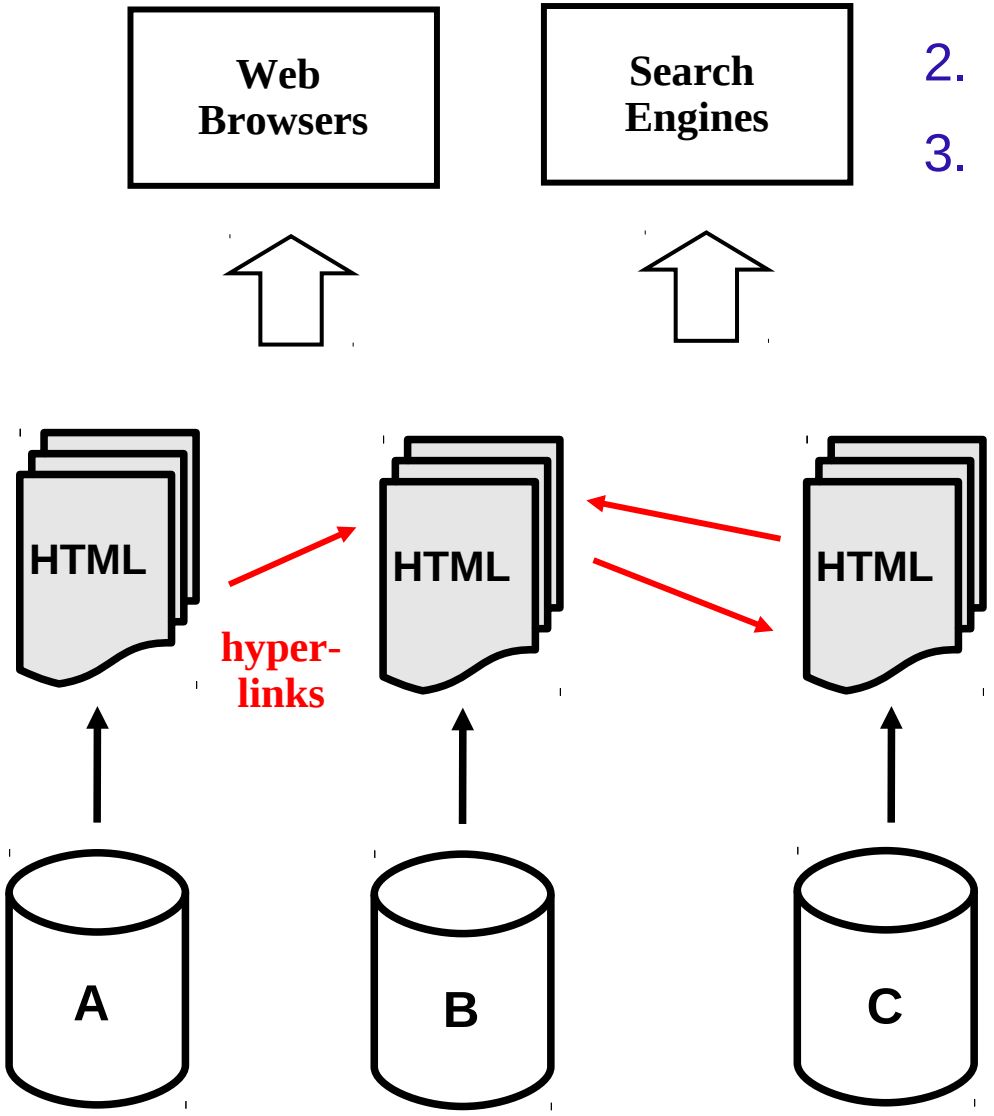
Linked data

1. Od spleta dokumentov do spleta podatkov
   - Spletni API-ji in Linked Data
2. Linked Data implementacija na spletu
   - Kateri podatki obstajajo?
3. Aplikacije
   - Kaj se dogaja s podatki?
4. Naslednji koraki
   - Kaj manjka?

Klasični splet

En sam globalen inform. prostor

1. URL za:
   - Globalni unikatni IDs
   - Poizvedovalni mehanizmi
2. HTML kot skupna oblika vsebine
3. Hyper-povezave



Problem in rešitev

Problem

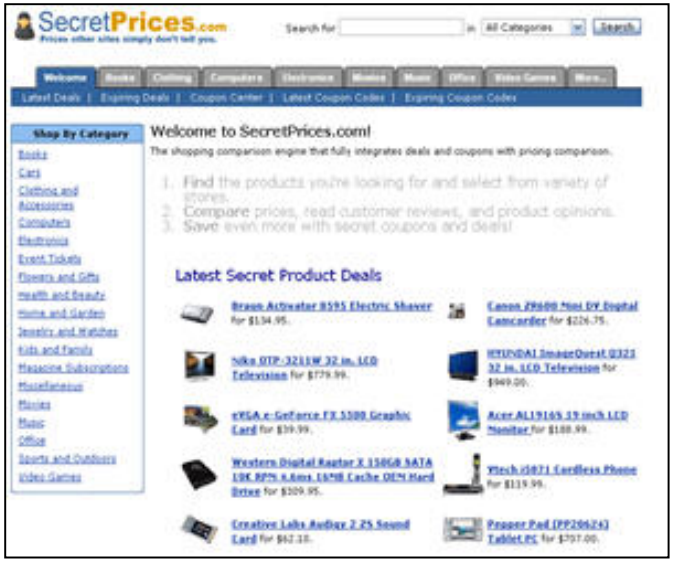
Ker je vsebina spleta zelo šibko strukturirana, aplikacije težko implementirajo pametne operacije.

Rešitev

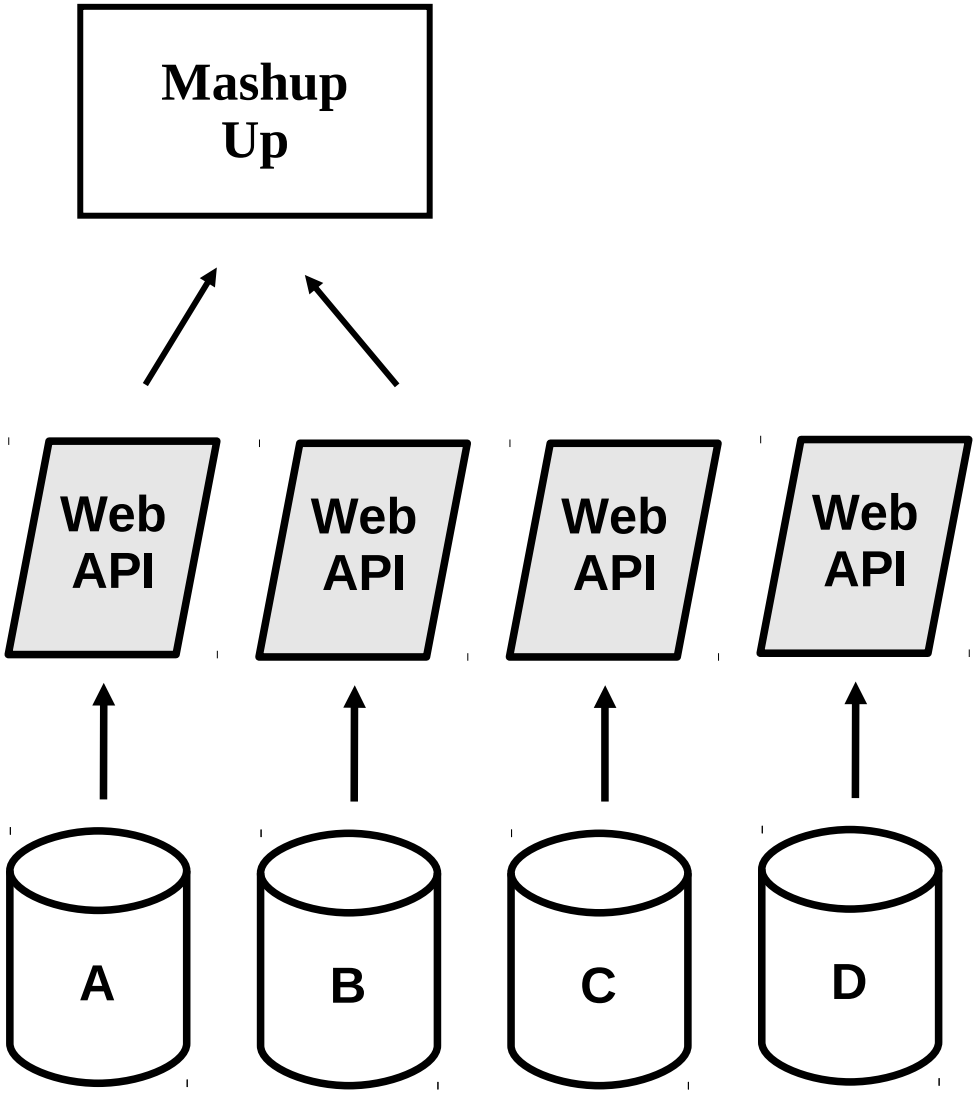
Povečaj strukturo vsebine spleta.

Spletni API-ji in prepletene storitve

programmableweb



Spletni API-ji in mashup-i



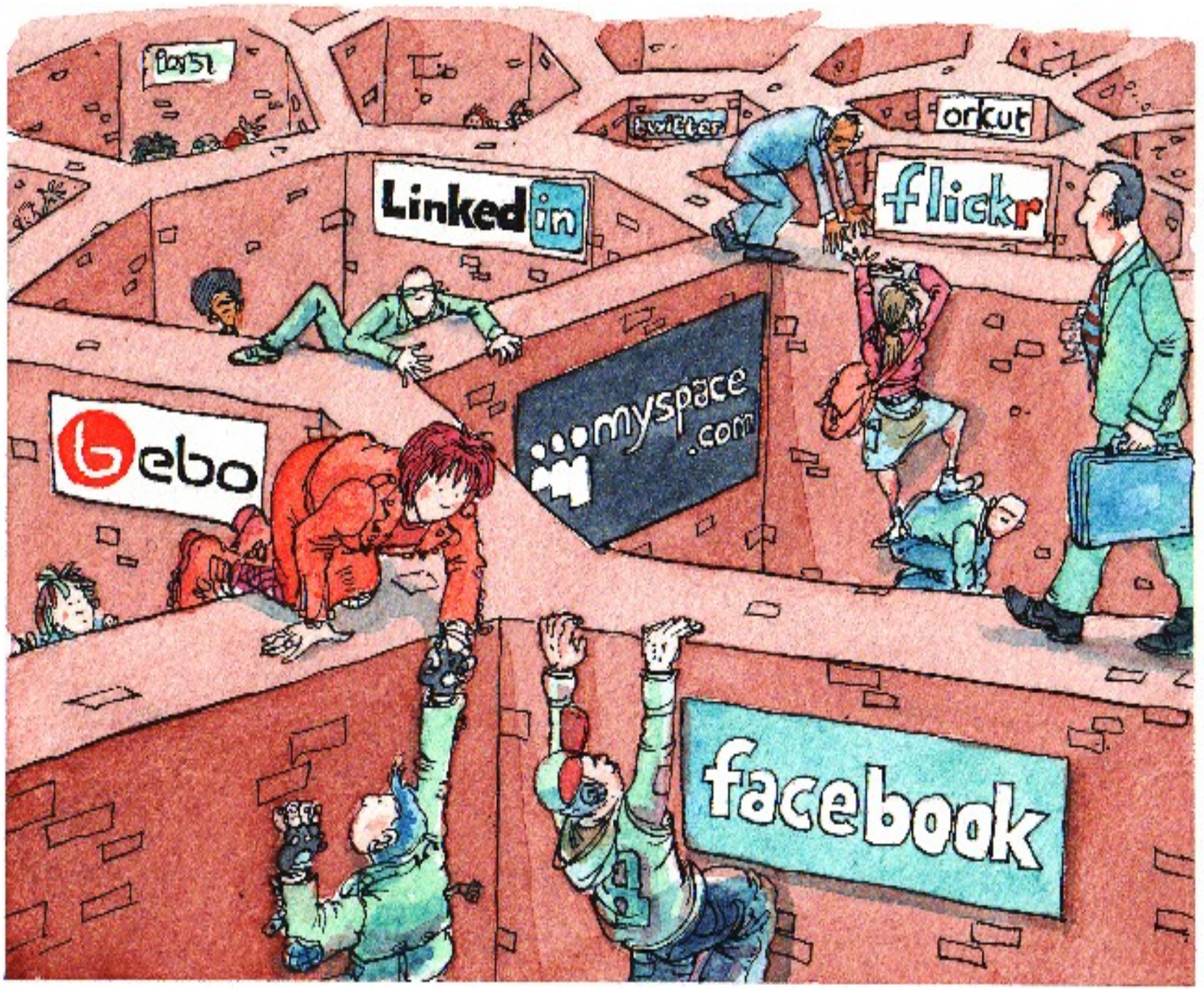
Slabe lastnosti:

0) API-ji nudijo privatne vmesnike.

1) Mashup-i temeljijo na fiksni množici podatkovnih virov.

2) Ne moremo definirati povezav med podatkovnimi objekti.

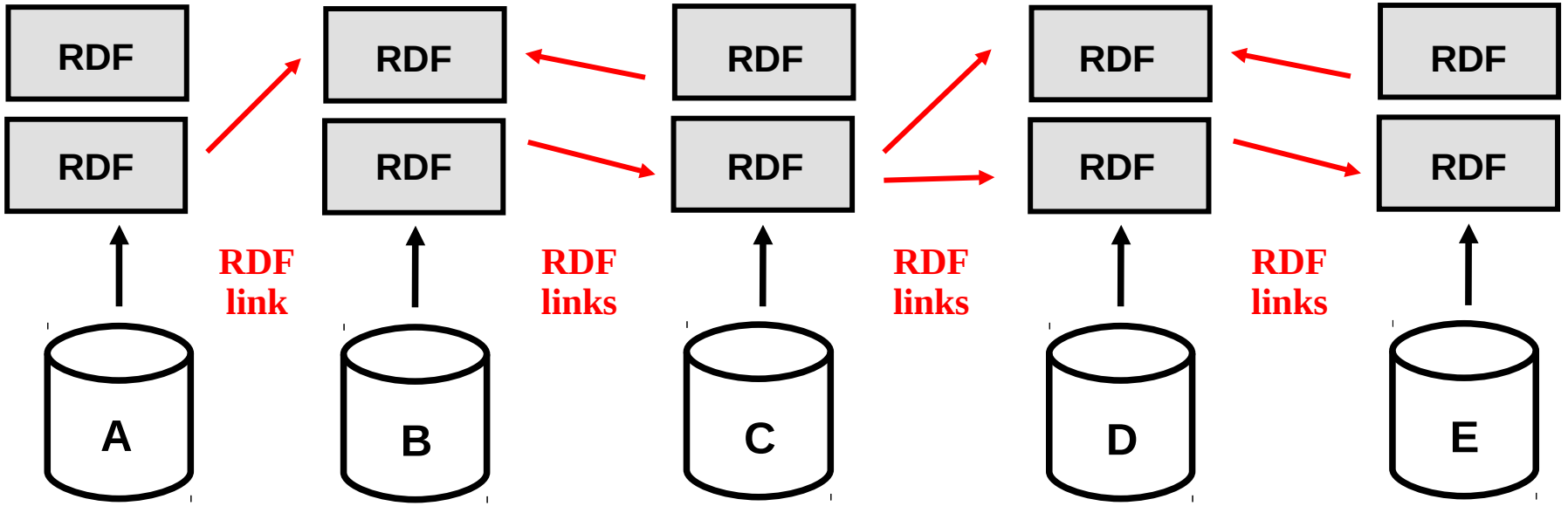
Spletni API-ji razdelijo splet na vrtove z ograjami



Povezani podatki

Uporabi tehnologije semantičnega spleta za:

- Publiciranje strukturiranih podatkov na splet,
- Definicijo povezav med podatki iz enega spletnega mesta s podatki v drugih spletnih mestih.



Principi povezanih podatkov

1. Uporabi URI-je kot imena za objekte
2. Uporabi HTTP URI, da lahko ljudje dostopajo do teh imen
3. Ko nekdo dostopa do URI zagotovi uporabne RDF podatke
4. Vključi RDF stavke drugih URI, da lahko ljudje odkrijejo sorodne objekte.

Tim Berners-Lee 2007

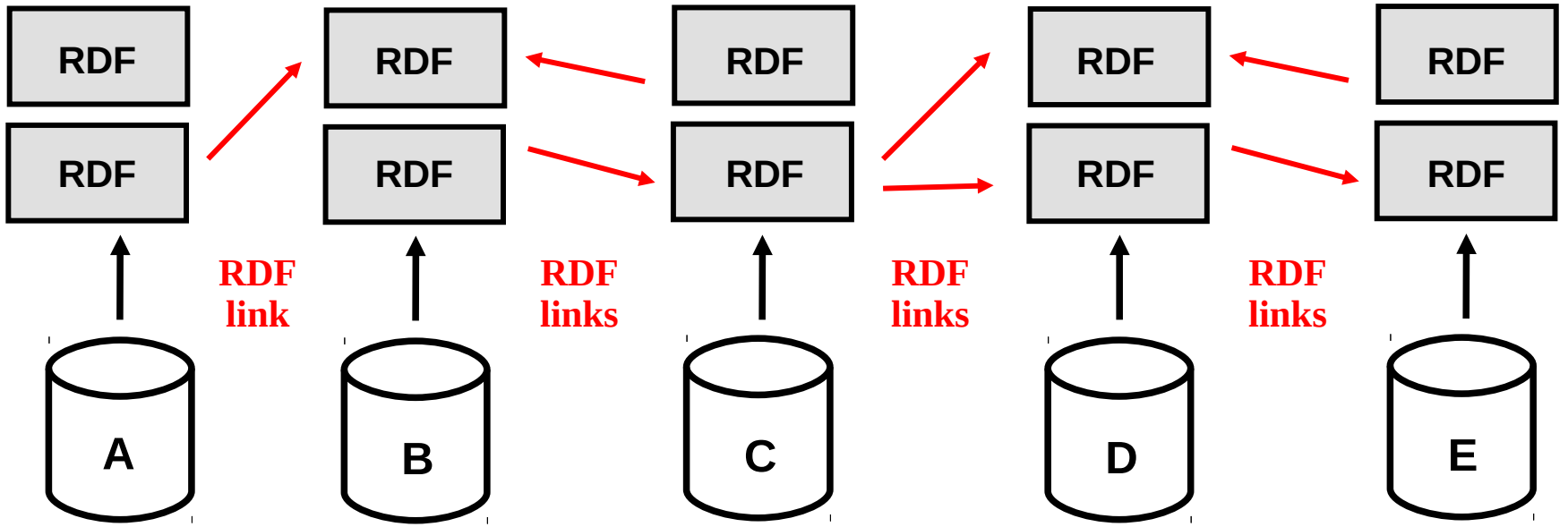
http://www.w3.org/DesignIssues/LinkedData.html

Lastnosti spleta povezanih podatkov

- Kdorkoli lahko publicira podatke na spletu povezanih podatkov
- Entitete so povezane s povezavami
  - Kreacija globalnega podatkovnega grafa, ki povezuje podatkovne vire in omogoča odkrivanje novih virov.
- Podatki so samo-opisni
  - Če aplikacija dobi podatke, ki so predstavljeni z nepoznanim besednjakom, mora aplikacija identificirati URI-je, ki identificirajo slovarje z definicijami RDFS in OWL izrazov.
- Splet podatkov je odprt
  - To pomeni, da lahko aplikacije odkrivajo nove podatkovne vire v času izvajanja.

Implementacija povezanih podatkov na spletu

- **Je to realno?**

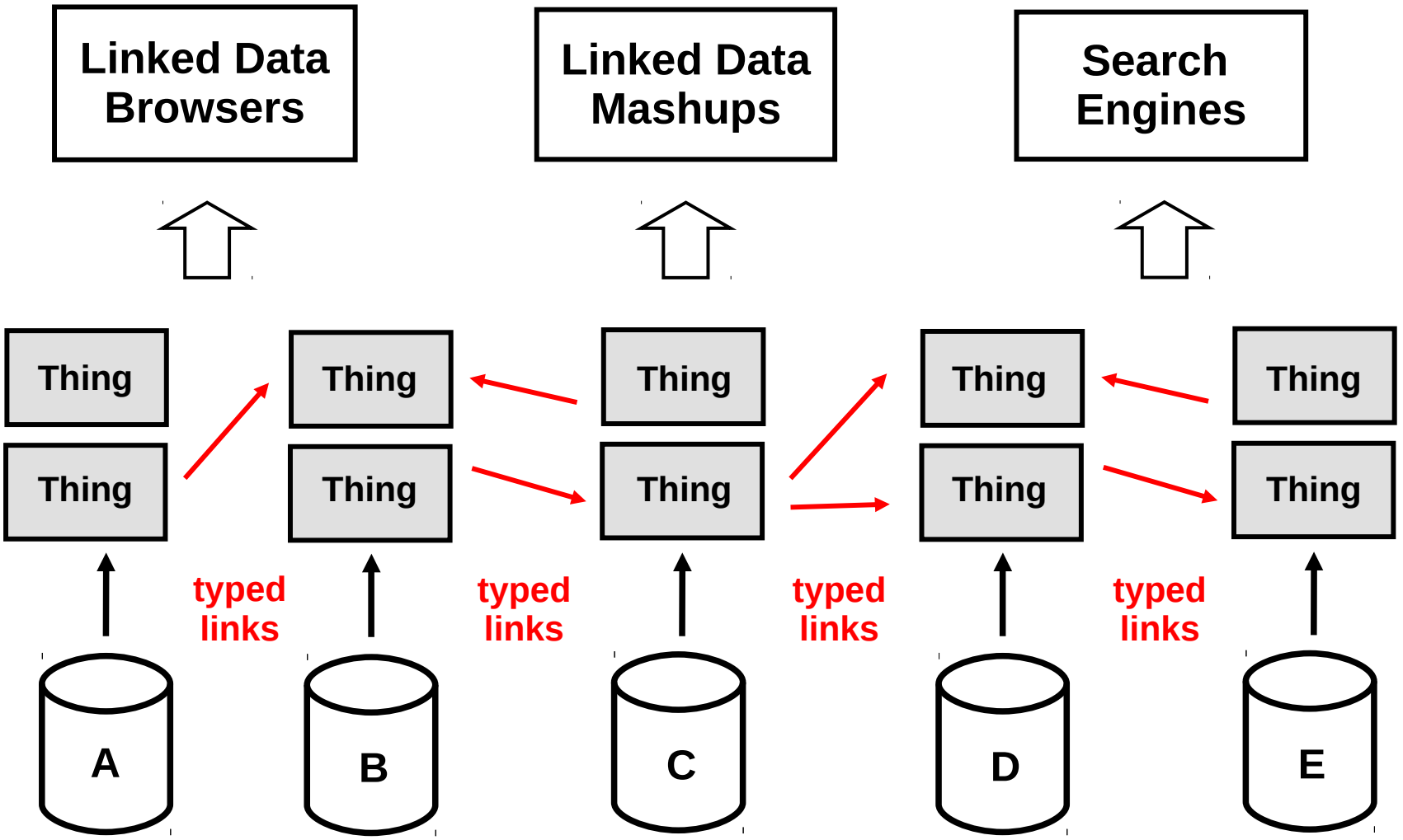


W3C Linking Open Data

- **Skupnost:**
  - Publiciranje obstoječih odprtih podatkovnih zbirk na spletu.
  - Povezovanje stvari med različnimi podatkovnimi viri

Aplikacije

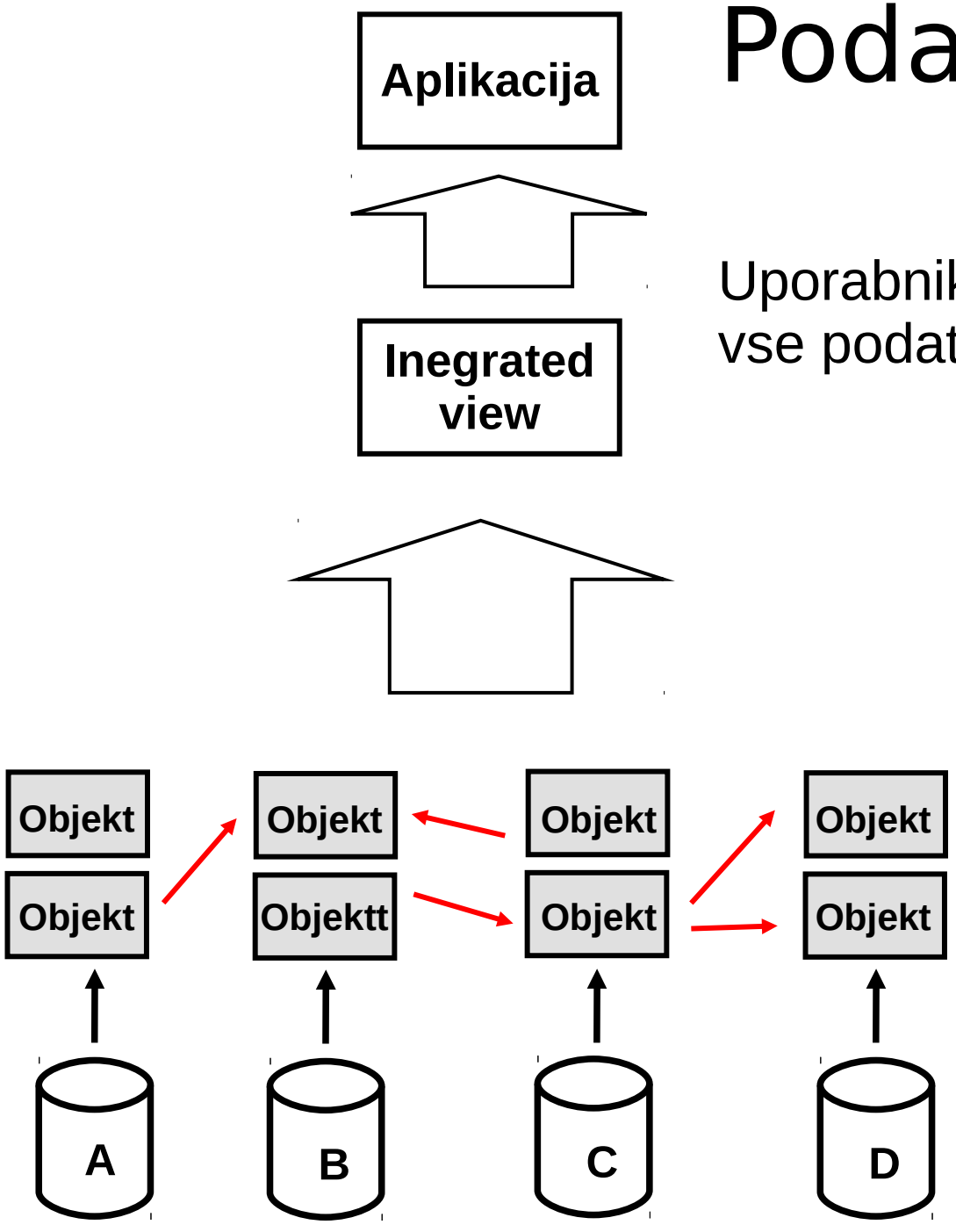
- Kaj lahko naredimo s tem?



Povezani podatki v prepletenih storitvah (meshups)

- Aplikacije s specifično domeno, ki uporabljajo povezane podatke iz spleta

Podatkovna fuzija



Uporabnik želi imeti integriran pogled na vse podatke, ki so dostopni za objekt.

Znani problemi:

Preslikave shem
Reševanje nekonsistentnosti

Design of big3store

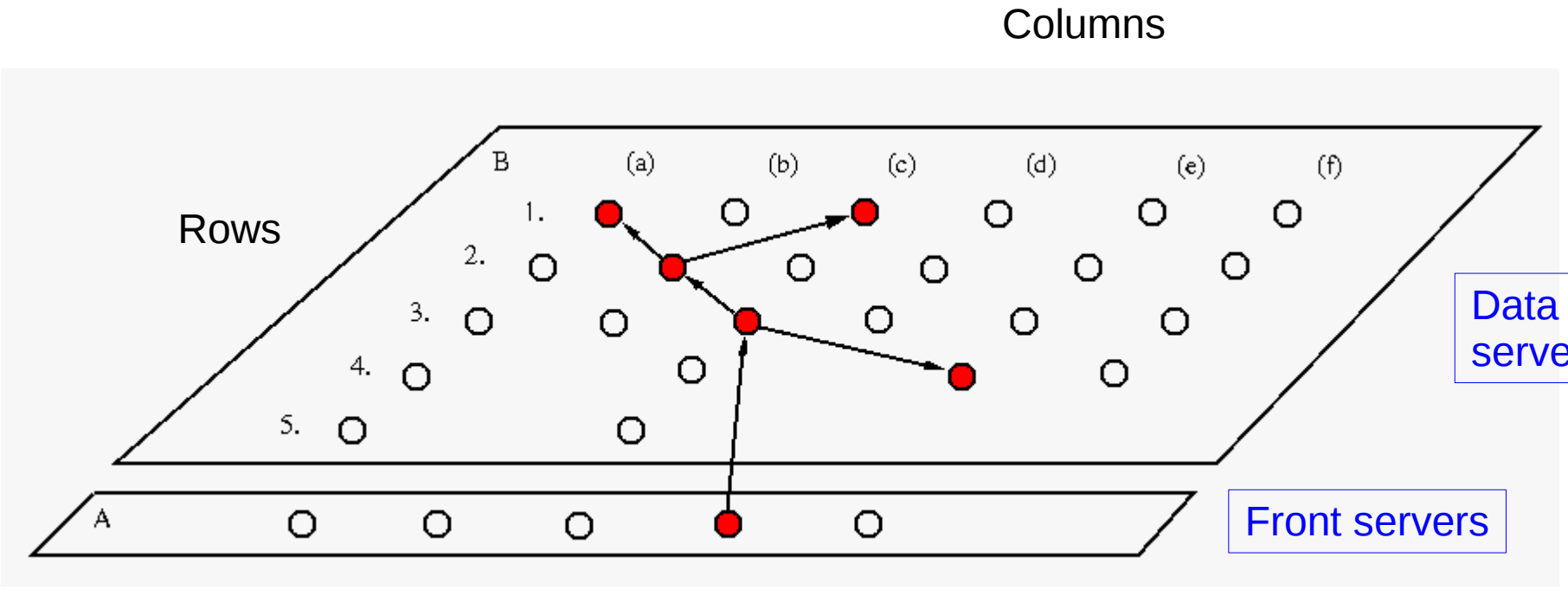
Basic decisions

- Use of inexpensive commodity hardware in shared-nothing cluster
- Concurrent programming language Erlang
- Use relational database system as local triple-store
- Exploit dataflow nature of RDF algebra for parallelisation of query execution

Erlang

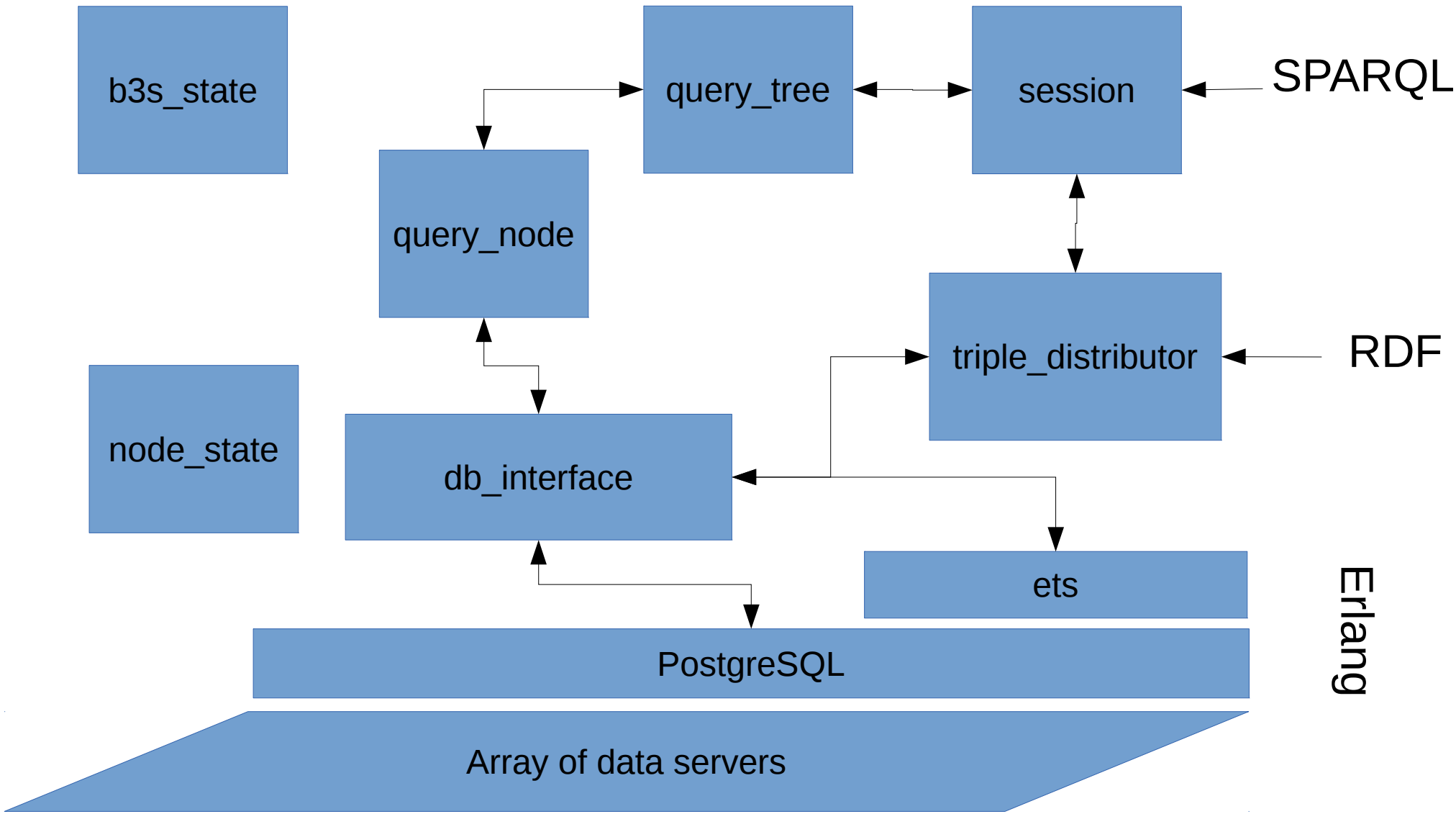
- Build massively scalable soft real-time systems
- Language features
  - Tends to be pure functional language
  - Prolog unification and clauses
  - Many build-in data structures
  - Relational dbms Mnesia
- Light-weight processes
  - Ingenious computing model
  - Processes are true objects !
  - Distributed programming

Architecture

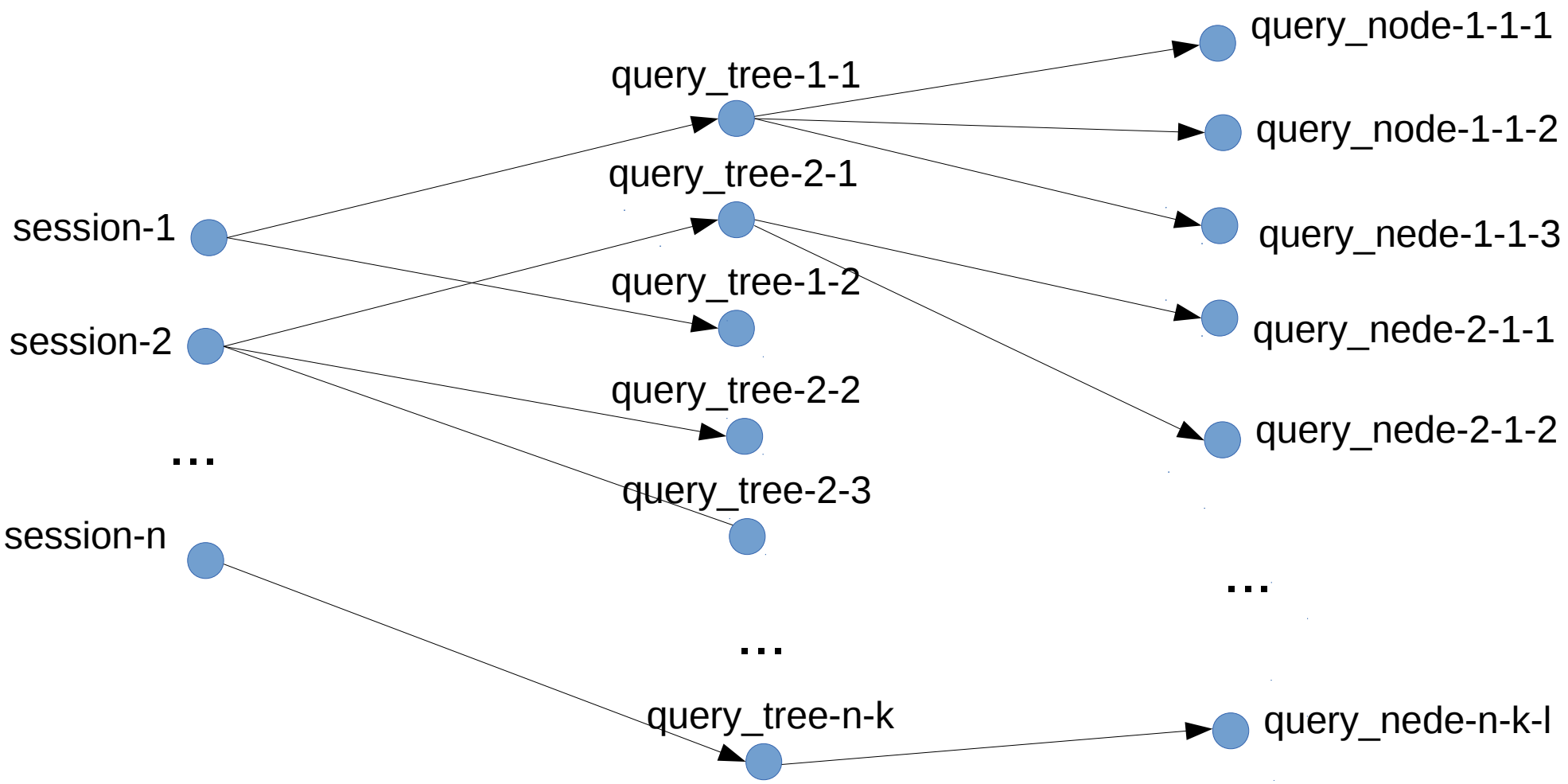


- Triple-base distributed to columns
- Triple-base parts replicated to rows

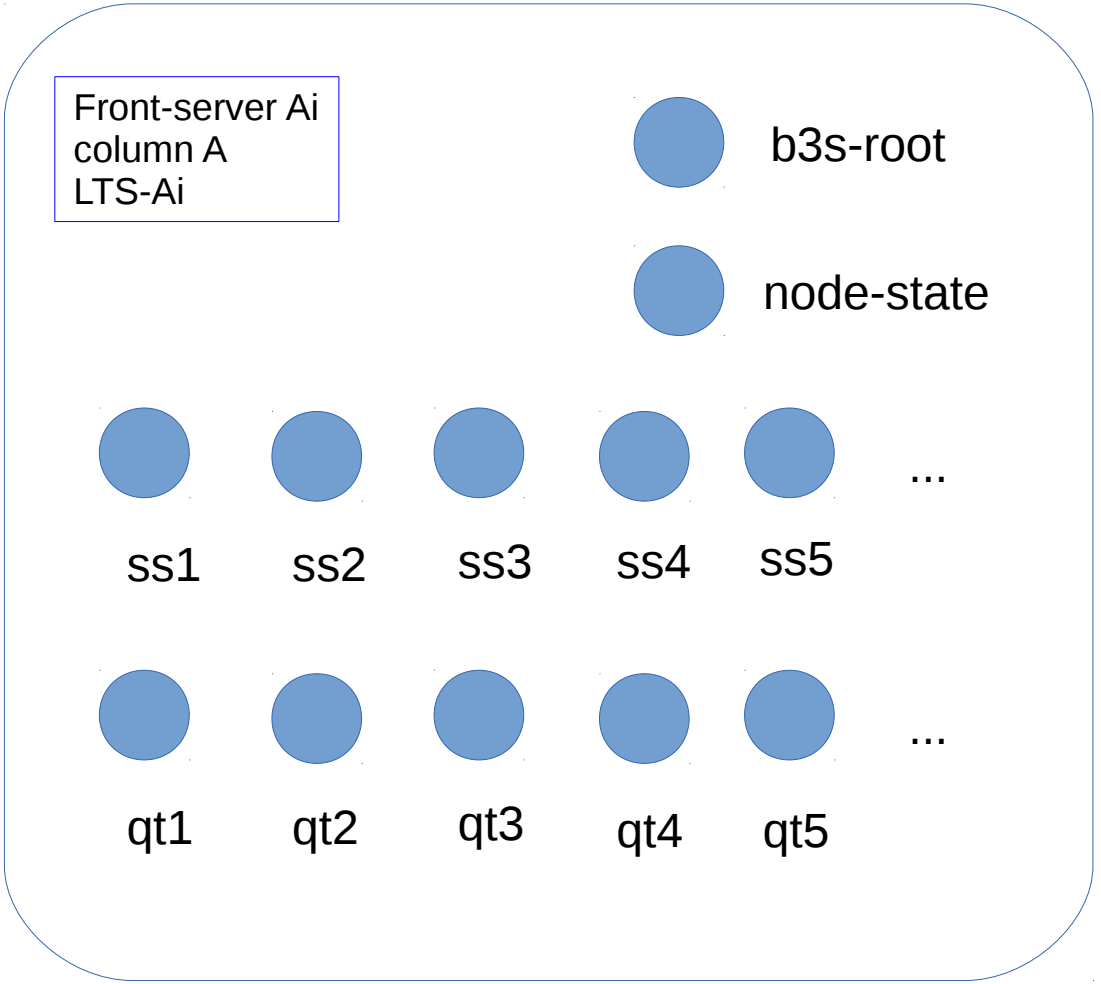
b3s modules – static view



b3s processes – logical view

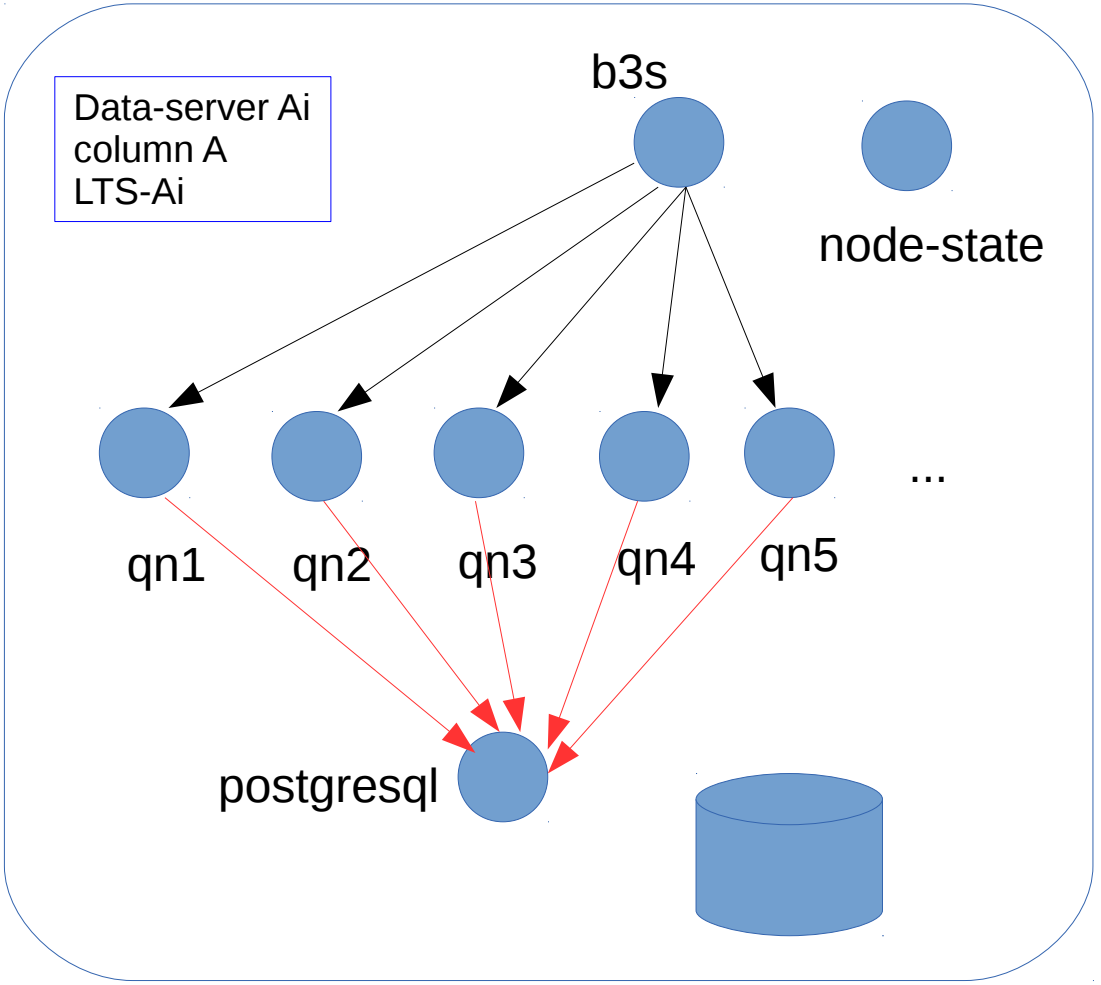


b3s processes – front server



- ss = session
- qt = query-tree

b3s processes – data server



- one node-state process
- triple-db split to columns!
- qn = query-node
- one supervisor b3s per site

access triple-store

Research topics

- Graph algebra
- Graph partitioning
- Local storage manager
- Query scheduling
- Computation of database statistics
- Query optimisation
- Multi-threaded architecture of query executor

Research topics

- Design of algebra of graphs
- RDF algebra based on relational algebra
  - Graph pattern = SQL block
- Denotational semantics
- Implementation in parallel comp env

RDF algebra

- `select`
- `project`
- `join`
- `union, intersect, difference`
- `leftjoin`

- Algebra of sets of graphs
- Sets of graphs are input and output of operations
  - Triple is a very simple graph
  - Graph is a set of triples

RDF algebra

Triple-patterns

Graph-patterns

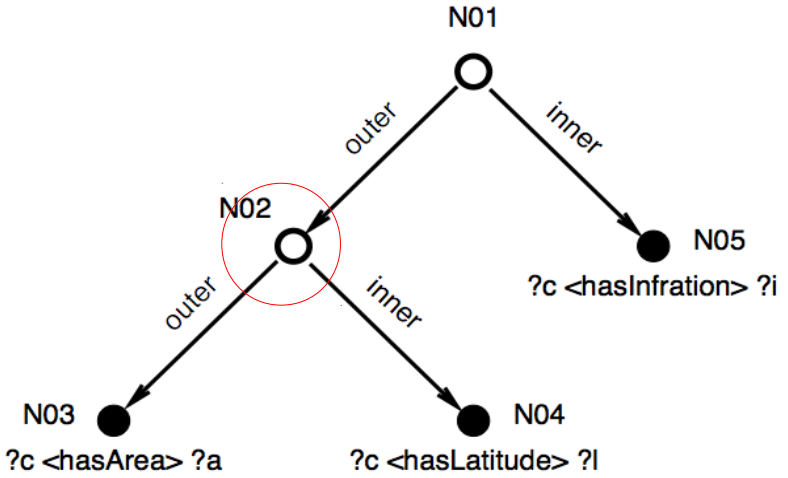
$$
\begin{aligned}
GP &::= TP \mid select(GP, C) \mid join(GP, GP) \mid union(GP, GP) \mid \\
&\quad\; intsc(GP, GP) \mid diff(GP, GP) \mid leftjoin(GP, GP) \\
TP &::= (S \mid V, P \mid V, O \mid V) \\
C &::= V\ OP\ V \mid V\ OP\ O \mid C \wedge C \mid C \vee C \mid \neg C \\
OP &::= = \mid \neq \mid > \mid \geq \mid < \mid \leq \\
S &::= \text{URI} \mid \text{Blank-Node} \\
P &::= \text{URI} \\
O &::= \text{URI} \mid \text{Blank-Node} \mid \text{Literal} \\
V &::= ?a \,..\, ?z
\end{aligned}
$$

Conditions

Variables

RDF algebra

```
SELECT * WHERE {
  ?c <hasArea>      ?a .
  ?c <hasLatitude>  ?l .
  ?c <hasInfration> ?i
}
```



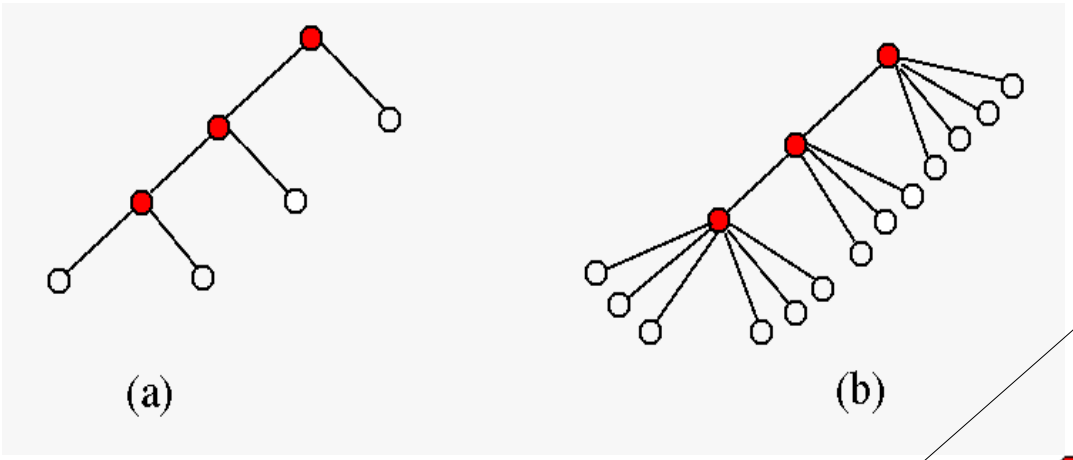
$$[\![join(gp_1, gp_2)]\!]_{db} = \{\, g_1 \cup g_2 \mid g_1 \in [\![gp_1]\!]_{db} \wedge g_2 \in [\![gp_2]\!]_{db} \wedge \forall v \in vs : val(v, gp_1, g_1) = val(v, gp_2, g_2) \,\}$$

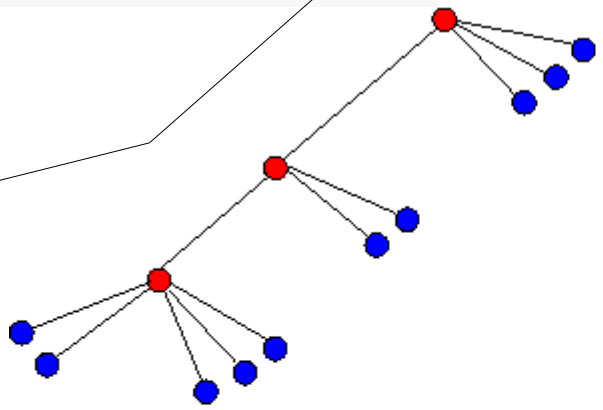
- Index nested-loop join
  - Exploiting DB indexes on subsets of { S, P, O }

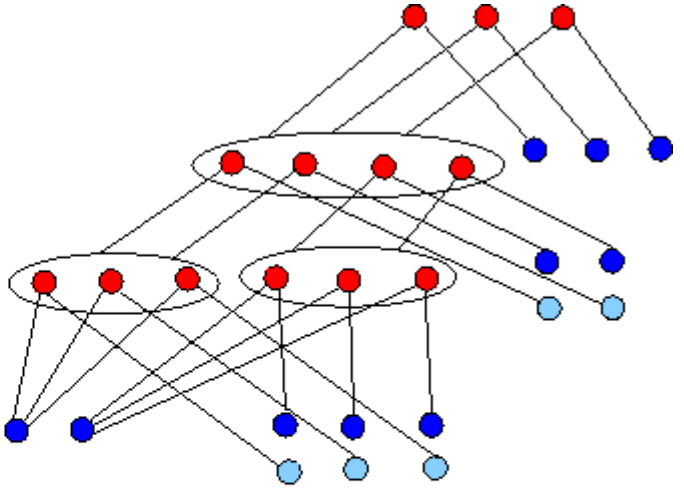
RDF algebra implementation

- Algebra operations implemented as processes on data-servers
- Query trees are left-deep trees (pipelines) !
- Flows (streams) of triples among physical machines
  - Speed of reading output triples ≅ speed of processing one algebra operation
  - Other operations of query work concurrently
- Experiments with bushy trees

Query tree implementation







tp–query node

replicas of tp–query node

join–query node

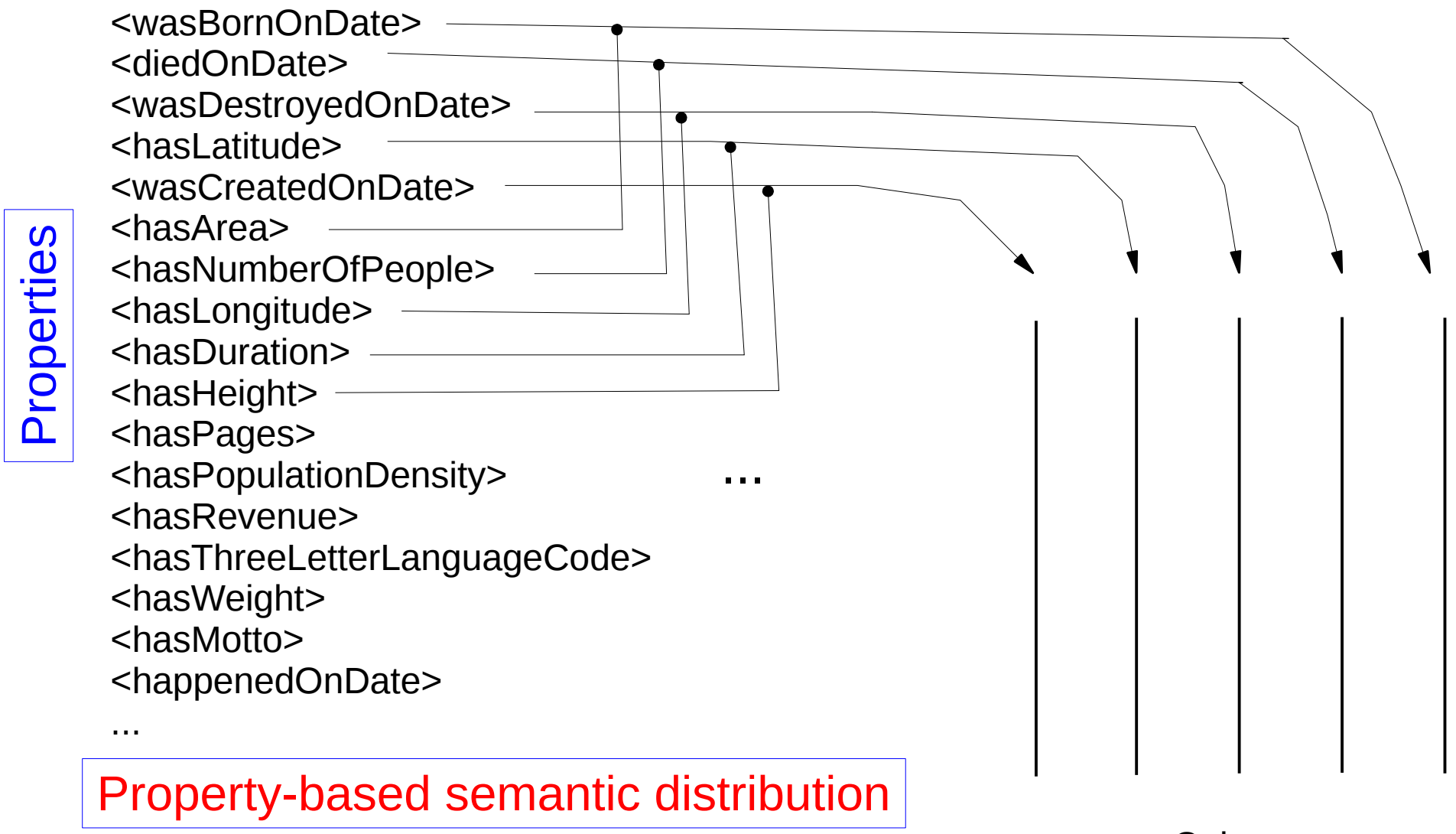
Research topic

- Graph partitioning
- How to partition large graph among the multiple servers to speed-up graph processing?
- Graph-theoretic approaches
  - Identify strongly connected components
- DB approaches
  - Hash-based partitioning
  - Semantic partitioning methods

Graph partitioning

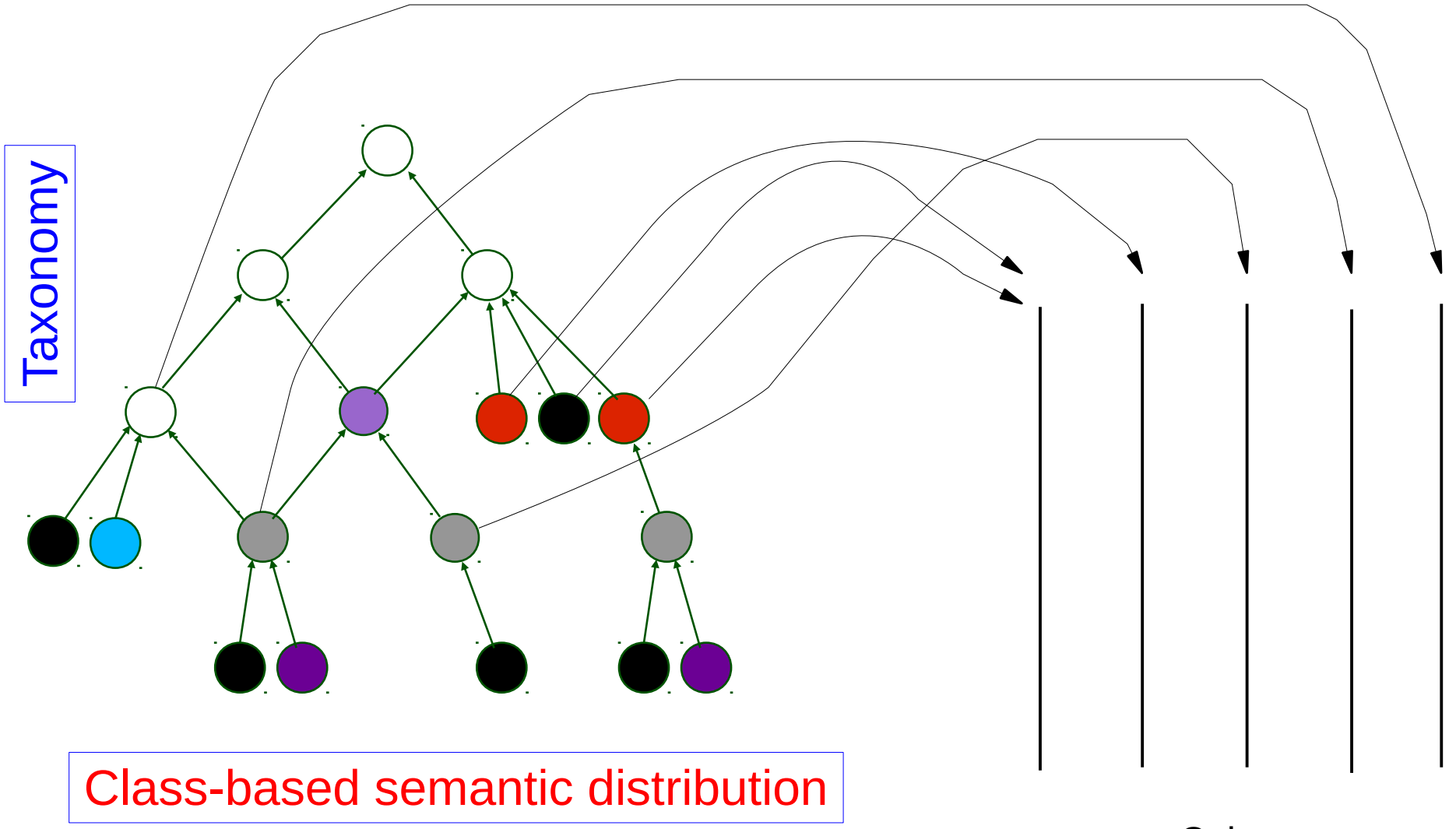
- Query that addresses large part of database should be distributed to as many data servers as possible
- Query that addresses small part of database needs few data servers
- Semantic distribution
  - Distribution based on triple-base schema
  - Property-based distribution
  - Class-based distribution
  - Based on {S, P, O} subset lattice

Semantic distribution



Property-based semantic distribution

Semantic distribution

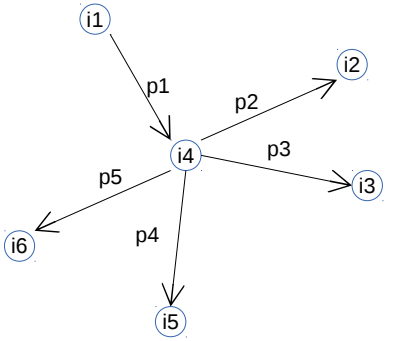


Graph partitioning method

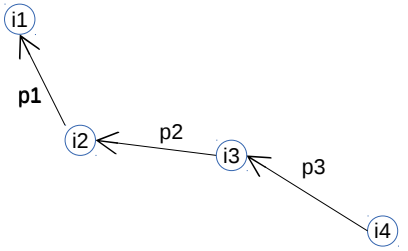
- The main idea of the method
  - Cluster the data on the schema level
    - Use statistics for the estimation
  - Distribute the extensions of the schema partitions

Graph partitioning method

- Graph patterns
  - Star-shaped
    - Triples of star-shaped pattern
      - (i1,p1,i4), (i4,p2,i2), (i4,p3,i3), …
    - Star-shaped patterns have star-shaped schema graph



  - Paths
    - Triples of path-shaped pattern
      - (i2,p1,i1), (i3,p2,i2), (i4,p3,i3)
    - Path-shaped patterns have path-shaped schema graph



Graph partitioning method

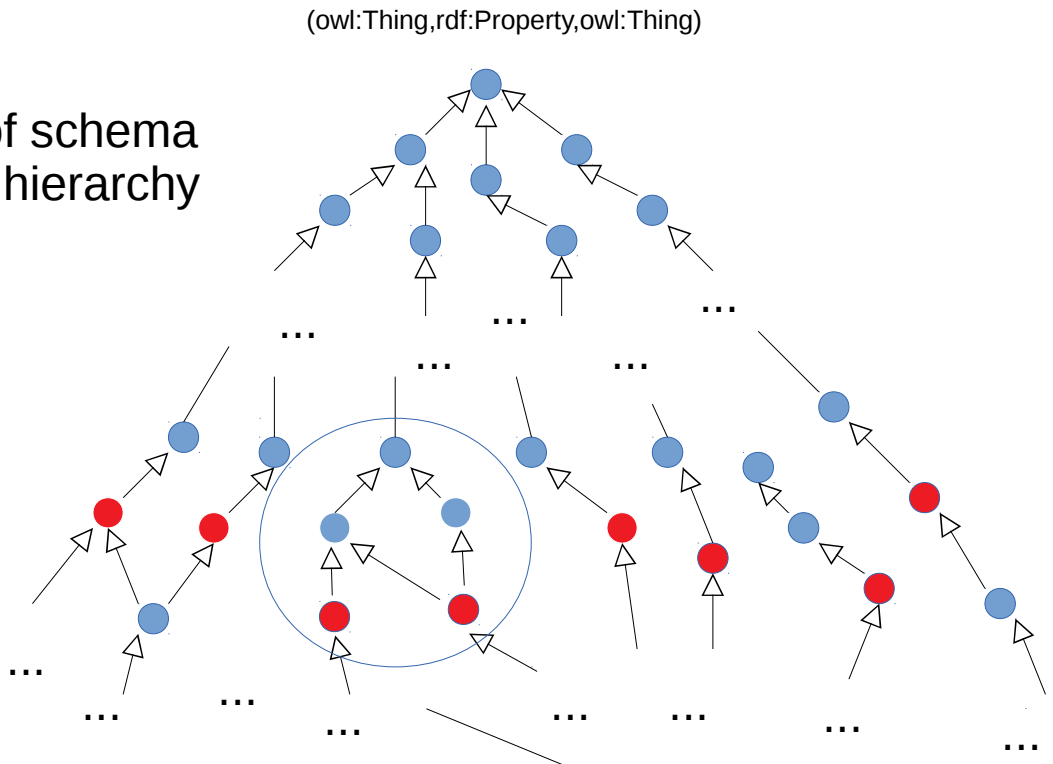
- Locality based partitioning
  - Cluster together similar graph patterns
    - Graph patterns in a cluster have similar shape
      - Edges of graph patterns may have common edge types
      - Some edges may be missing
  - Graph patterns in a cluster are strongly connected
    - Through the common schema graph
      - Similar patterns have common edge types
      - Similar patterns are connected by rdf:type triples

Graph partitioning method

1. Choose a skeleton graph from the hierarchy of edge types
   - Edge types are ordered into partially ordered set
   - Start from the top most general edge type
   - Specialize edge types until they are of appropriate size
- Cluster a skeleton graph to obtain k partitions
   - Cluster strongly connected edges together
   - Connectivity is defined by means of the statistics of edge types

Computing skeleton graph

Top of schema triple hierarchy



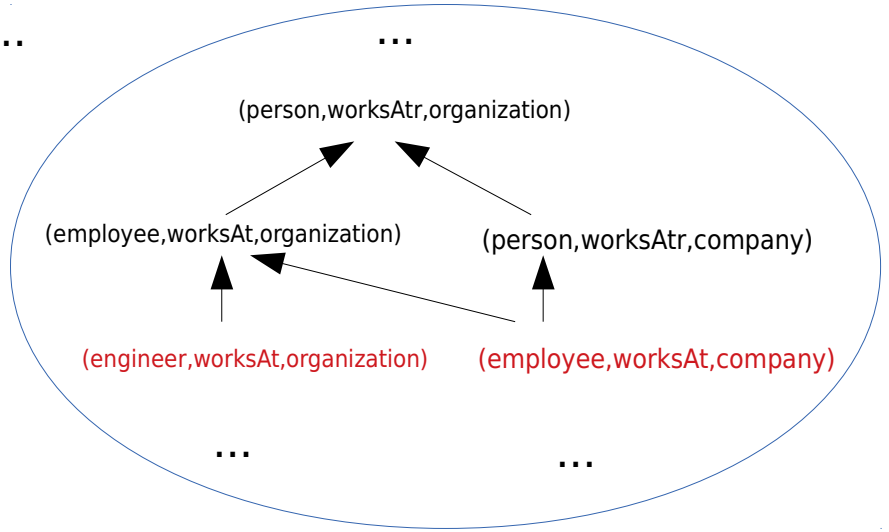
= “is more specific triple”

= schema triple

= schema triples that have the interpretation of appropriate size

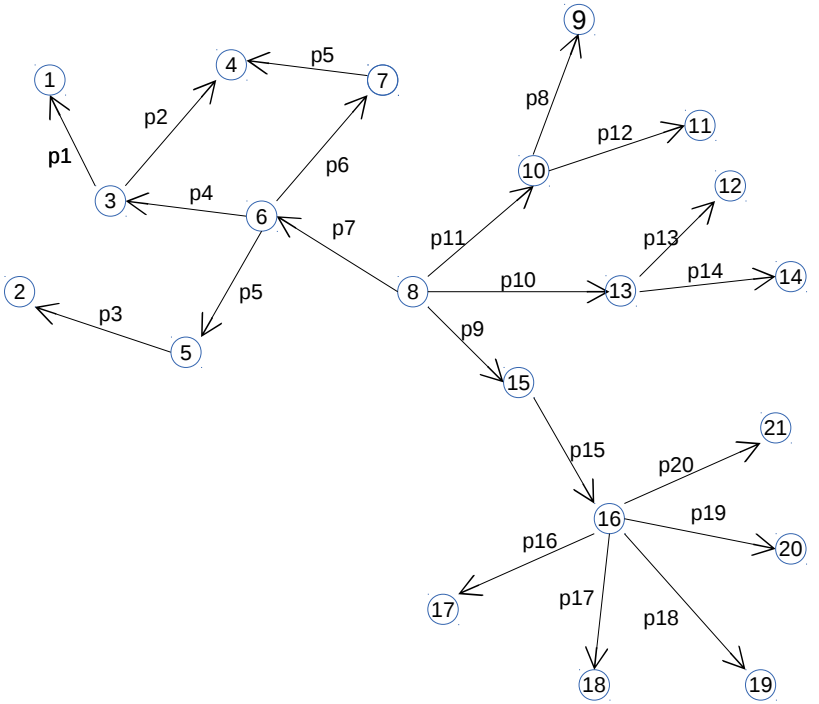
= eges of the skeleton graph

Schema graph = selected schema triples



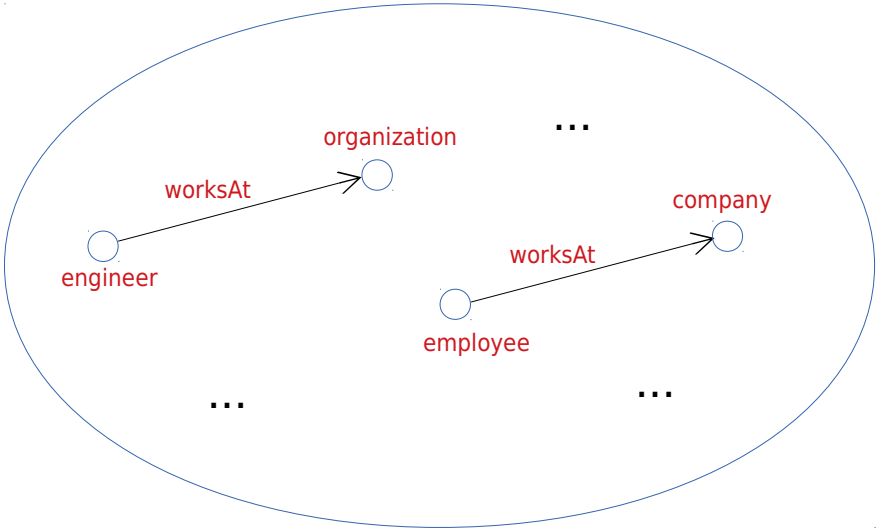
Clustering skeleton graph

Given:
- statistics of TS
- skeleton graph G_s



Schema graph
- selected schema triples
- represented as graph !

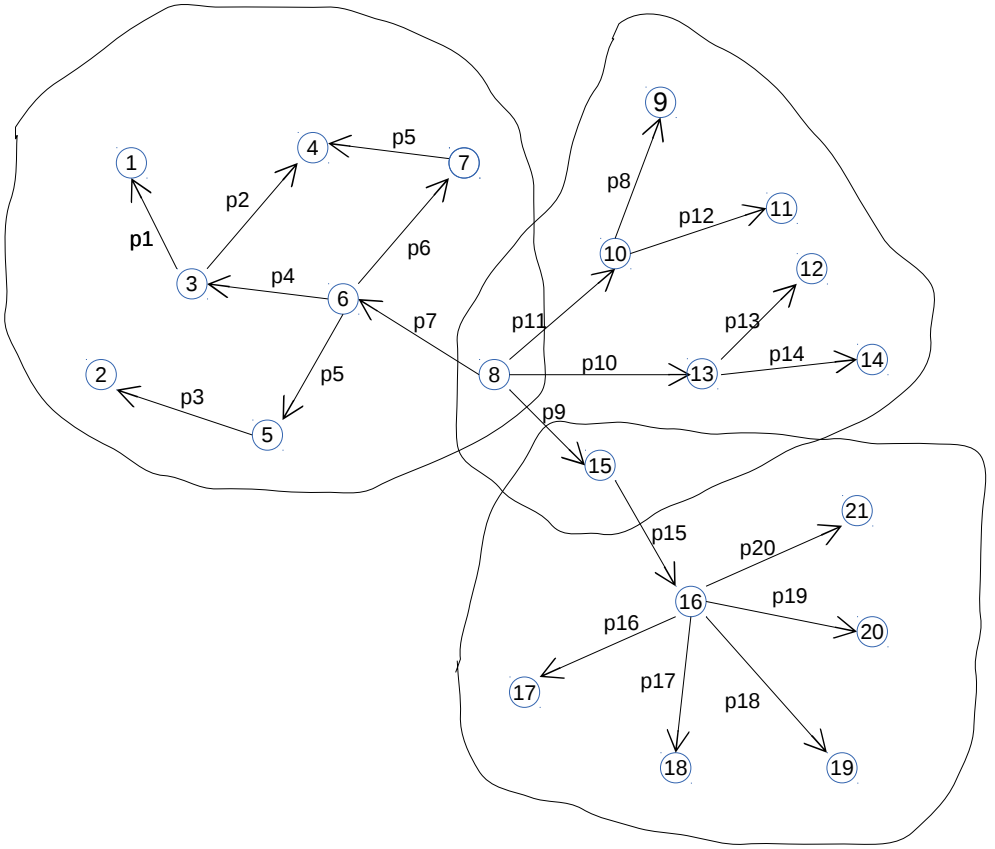
Distance function:
- distance between edges e_1 and e_2
  - based on shortest path *p* starting with e_1 and ending with e_2
  - estimate the number of path *p* instances
  - estimate the cardinality of each join in a path p by using the statistics of TS



Clustering skeleton graph

Clustering algorithm:

- any clustering algorithm
  - strongly connected edge types are clustered together
  - maximize average strength of the paths among all different pairs of nodes from a partition (see problem definition, page 7)



Statistics:

- For each schema triple t_s
  - # instances of edge type t_s
  - # distinct values of edge type t_s
  - estimation of the size of joins

Result:

- partitions of G_s (sets of edges)

Research topic

- Local storage manager
  - Relational approach
    - Triple-table with 7 indexes
  - Special new storage system
    - New indexes and storage structures
  - Graph-theoretic approach
    - Graph represented as nodes and links
    - New paradigm (neo4j), no joins
  - Our approach
    - Postgers triple-table + 7 indexes + Large cache in RAM

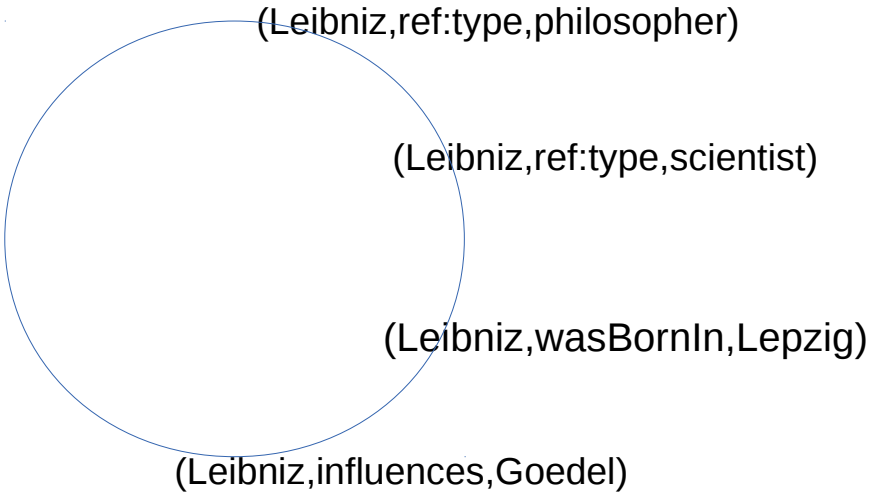
Main-memory usage

- Main-memory databases
  - Trinity DBMS
- Hybrids using large RAM and disk
  - Caching data into RAM
  - Storage manager cache

Distributed cache

- Cost of RAM allows moving significant part of triple-store in RAM
- Problem similar to using cache in multi-processor systems
  - We will use affinity scheduling
  - Queries of one session tend to allocate the same servers to utilise DB cache

```
philosopher rdfs:subClassOf    person .
scientist   rdfs:subClassOf    person .
person      influences         person .
person      wasBornIn          location .
Plato  rdf:type       philosopher .
Leibniz     rdf:type       philosopher .
Leibniz     rdf:type       scientist .
Goedel      rdf:type       scientist .
Athens      rdf:type       location .
Leipzig     rdf:type       location .
Brno   rdf:type       location .
Plato  wasBornIn      Athens .
Plato  influences     Leibniz .
Leibniz     wasBornIn      Leipzig .
Leibniz     influences     Goedel .
Goedel      wasBornIn      Brno .
```



Epsilon cache

```
epsilon# load ../simple.tsv
Loading...
done load
epsilon# print store
3store
rid=0 S=philosopher P=rdfs:subClassOf O=person iS=0 iP=1 iO=2 iSP=0 iSO=0 iPO=1
rid=1 S=scientist P=rdfs:subClassOf O=person iS=1 iP=0 iO=0 iSP=1 iSO=1 iPO=0
rid=2 S=person P=influences O=person iS=3 iP=14 iO=1 iSP=2 iSO=2 iPO=2
rid=3 S=person P=wasBornIn O=location iS=2 iP=15 iO=10 iSP=3 iSO=3 iPO=3
rid=4 S=Plato P=rdf:type O=philosopher iS=12 iP=10 iO=5 iSP=4 iSO=4 iPO=5
rid=5 S=Leibniz P=rdf:type O=philosopher iS=14 iP=4 iO=4 iSP=6 iSO=5 iPO=4
rid=6 S=Leibniz P=rdf:type O=scientist iS=5 iP=5 iO=7 iSP=5 iSO=6 iPO=7
rid=7 S=Goedel P=rdf:type O=scientist iS=15 iP=6 iO=6 iSP=7 iSO=7 iPO=6
rid=8 S=Athens P=rdf:type O=location iS=8 iP=7 iO=3 iSP=8 iSO=8 iPO=10
rid=9 S=Leipzig P=rdf:type O=location iS=9 iP=8 iO=8 iSP=9 iSO=9 iPO=8
rid=10 S=Brno P=rdf:type O=location iS=10 iP=9 iO=9 iSP=10 iSO=10 iPO=9
rid=11 S=Plato P=wasBornIn O=Athens iS=4 iP=3 iO=11 iSP=11 iSO=11 iPO=11
rid=12 S=Plato P=influences O=Leibniz iS=11 iP=2 iO=12 iSP=12 iSO=12 iPO=12
rid=13 S=Leibniz P=wasBornIn O=Leipzig iS=6 iP=11 iO=13 iSP=13 iSO=13 iPO=13
rid=14 S=Leibniz P=influences O=Goedel iS=13 iP=12 iO=14 iSP=14 iSO=14 iPO=14
rid=15 S=Goedel P=wasBornIn O=Brno iS=7 iP=13 iO=15 iSP=15 iSO=15 iPO=15
done print store
epsilon#
```

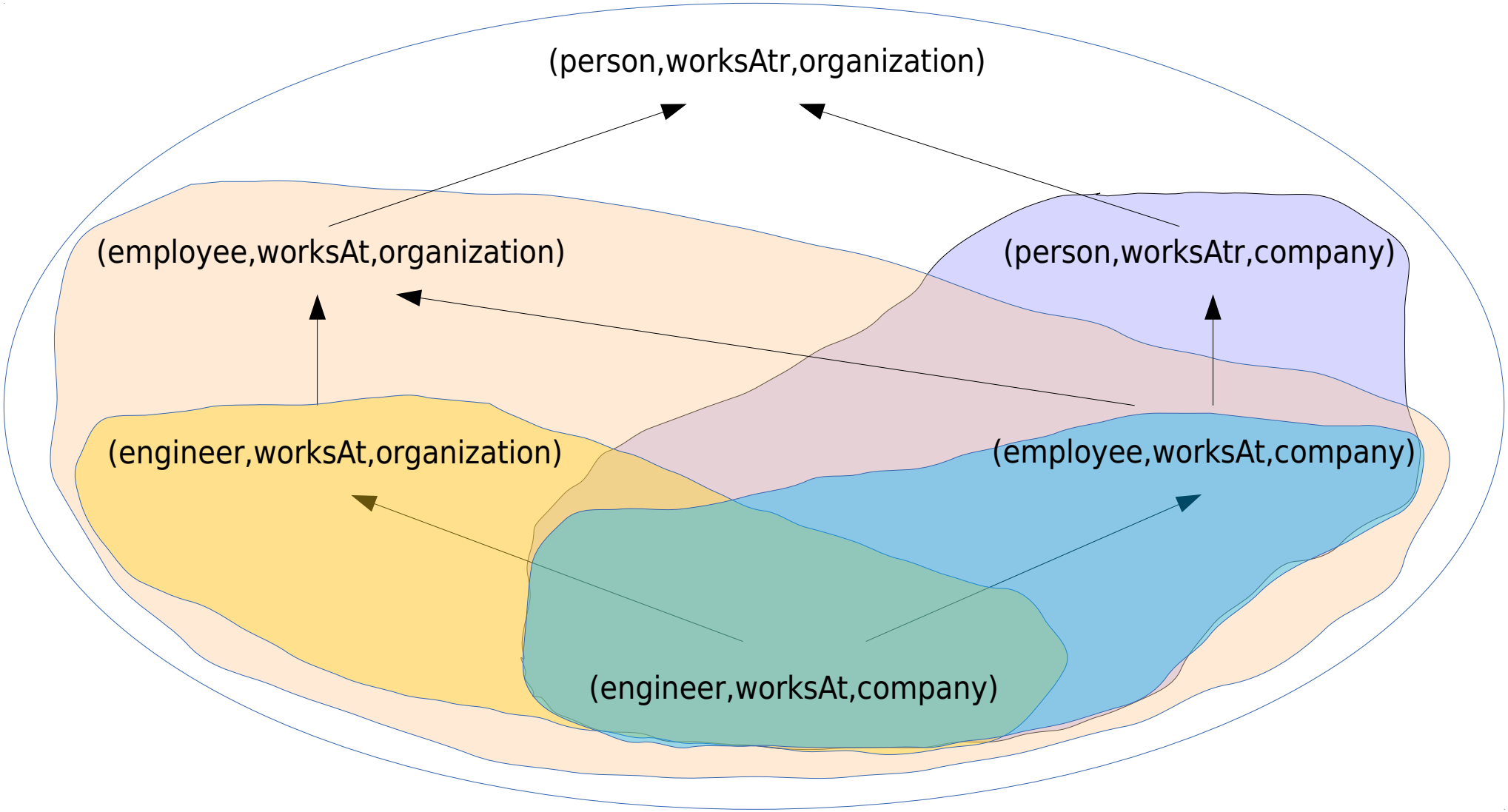
Research topic

- Computation of database statistics
- Estimation of the size of triple-pattern result
- The use of statistics
  - Query optimization
  - Data distribution
- Some solutions
  - Statistics of indexes S,P,O,SP,SO,PO,SPO
  - Gathering histograms for all triple-patterns
  - Gathering statistics for frequent paths

Taxonomy of schema triples

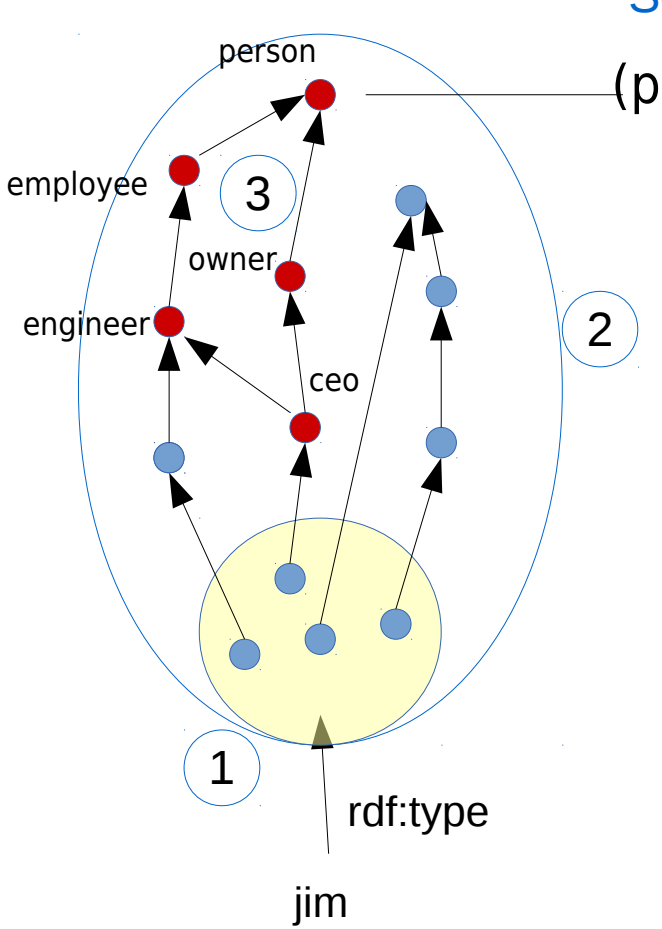
- Schema triple is triple that includes solely class identifiers and predicate
  - Interpretation of a schema triple includes all more specific triples (ground and schema triples)
  - Interpretation defines an area of triple-store
  - Schema triple is defined by the domain (rdfs:domain) and range (rdfs:range) of a predicate
- Areas based on schema triples are partially ordered by the relationship *subsume*

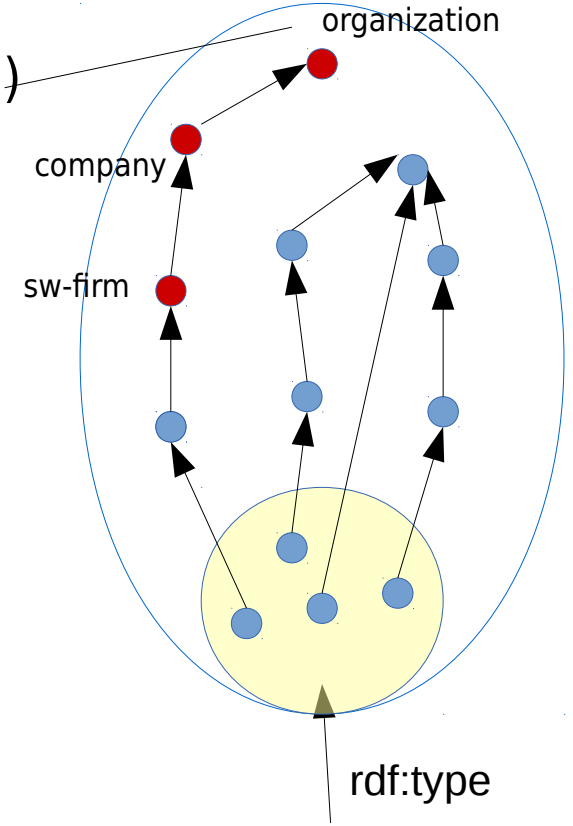
Schema triples and areas



Schema triple

(person, worksAt, organization)





Ground triple

(jim,worksAt,neo4j)

1. Computing set T of types of “jim”
2. Transitive closure of T with respect to rdfs:subClassOf
3. Taking k levels of hierarchy starting at “person”

Update statistics

inc (person,worksAt,organization)
inc (employee,worksAt,organization)
inc (engineer,worksAt,organization)
...
inc (person,worksAt,company)
inc (employee,worksAt,company)
inc (engineer,worksAt,company
...
inc (person,worksAt,sw-firm)
inc (employee,worksAt,sw-firm)
inc (engineer,worksAt,sw-firm)
...

Research topic

- Distributed query optimization
  - The hardest problem in database systems
  - Exploiting relational query optimization
  - Simplicity of triple-store model gives hope...
  - Regular path queries
    - New paradigm for optimization
    - Andreas T. Schmidt, KIT

Regular path queries

- SPARQL 1.1
  - Includes regular path expressions
- Examples

```
?x foaf:mbox <mailto:alice@example> .
?x foaf:knows/foaf:knows/foaf:name ?name .
```

=

```
?x  foaf:mbox <mailto:alice@example> .
?x  foaf:knows ?a1 .
?a1 foaf:knows ?a2 .
?a2 foaf:name ?name .
```

```
?x foaf:mbox <mailto:alice@example> .
?x foaf:knows+/foaf:name ?name .
```

Find the names of all the people that can be reached from Alice by foaf:knows.

```
<http://example/thing> rdf:type/rdfs:subClassOf* ?type .
```

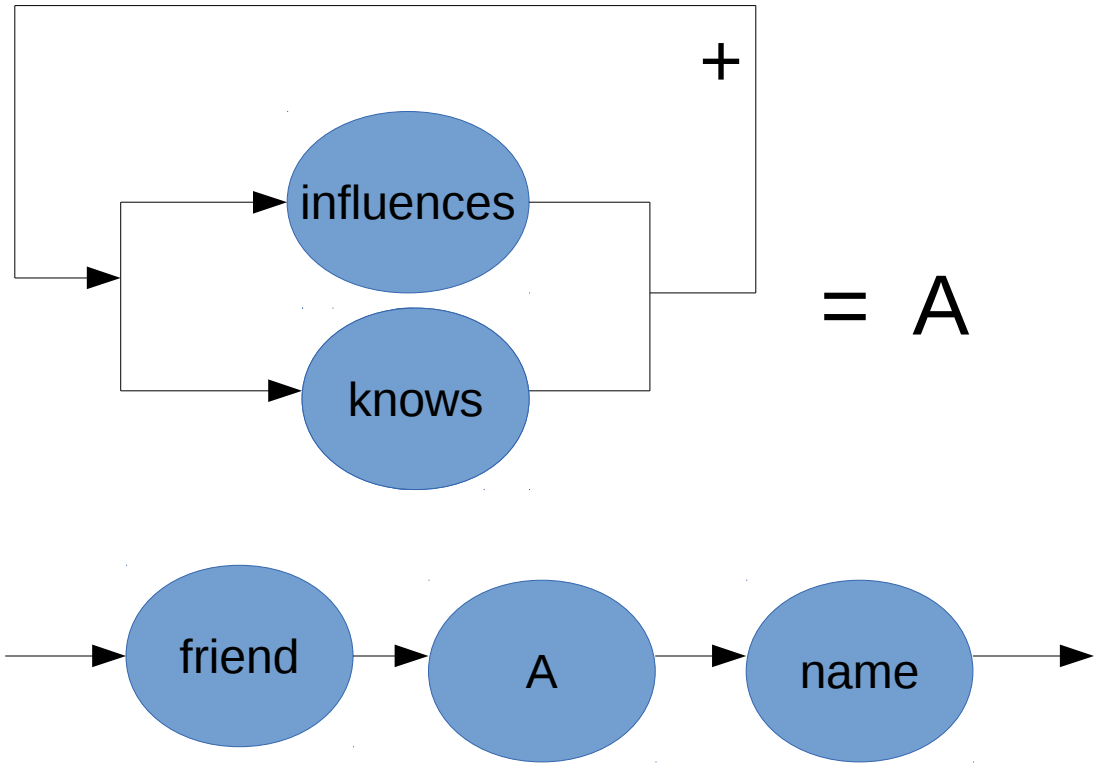
Limited inference:
all types and supertypes of a resource.

```
?x rdf:type/rdfs:subClassOf* ?type .
```

All resources and all their inferred types.

Regular path queries

- Hierarchical optimization based on dynamic programming
  - RPQ: friend /(knows|influences)+/name




```
#  Algorithm: optimize regular path query
optimize-rpq q

(1)   if q is simple path then
(2)      qt <- construct-query-tree q;
(3)      ot <- optimize-path-block qt;
(4)      return ot;
(5)
(6)   ql <- decompose q into outermost components;
(7)
(8)   tl <- empty list;
(9)   for each qi in ql do
(10)     ti <- optimize_rpq qi;
(11)     tl <- ti tl;
(12)
(13)  qt <- construct-query-tree tl;
(14)  ot <- optimize-path-block qt;
```

Research topic

- Efficient scheduling of queries on cluster of servers
  - Task: map nodes of query tree to processes on data servers
  - Input: query tree as data structure
  - Output: tree of processes running on cluster
  - Front server function
- Distribution of queries into cluster columns depends entirely on data distribution
  - Should work so that queries adressing large part of DB should allocate more columns

Scheduling

- Many query trees can be executed in parallel
- Triple-pattern query node must be evaluated on server where data is stored
- Join query node can be evaluated either on inner or outer query node of join
- Load-ballancing among replicas (data servers) of columns
  - Each query node can be started on one of rows (data servers) of a given column

Scheduling

- Load balancing algorithms:
  - Random
  - Dynamic load-balancing
  - Affinity scheduling
- Dynamic load ballancing and affinity sheduling are not easy to implement fast
  - The rows (replica server) of columns must be decided fast
  - Global data structure or data synchronisation

Research topic

- Multi-threaded architecture of query executor
  - We have multiple cores that could be utilized
  - Exploit programming languages paradigm
    - Erlang
  - Parallel algorithm design
    - Boris Motik, Oxford

Thank you !