Information Systems
—— Semantic Web

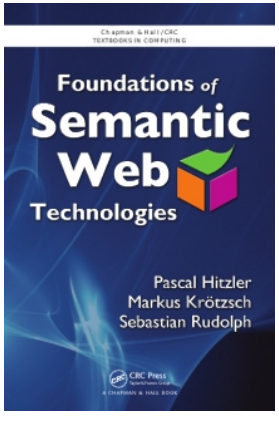
Yue Ma

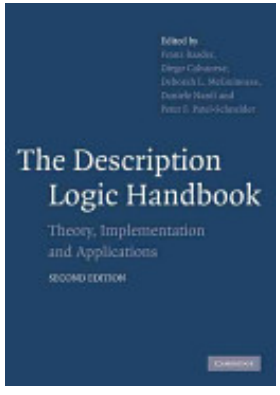
Laboratoire de Recherche en Informatique (LRI)
Université Paris Sud
ma@lri.fr

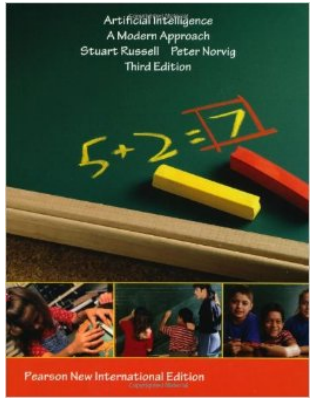
General Information

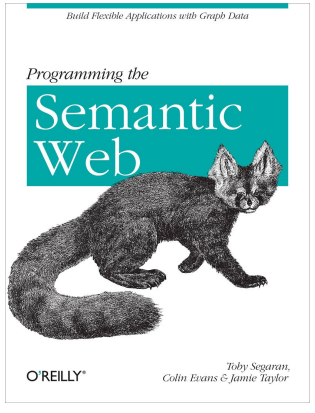
- This lecture is inspired by the following courses :
  - Knowledge Representation for the Semantic Web
    Pascal Hitzler, Wright State University, US
  - Foundations of Semantic Web Technologies
    Sebastian Rudolph, TU Dresden, Germany
  - Knowledge Graph Analysis
    Jens Lehmann, University of Bonn, Germany
  - Introduction à l'intelligence artificielle
    Laurent Simon, Polytech Paris Sud

- Reference books :









General Information

- Course materials : `www.lri.fr/~ma/M2DK`
- Organizational matters :
  - Initial plans :
    - Part 1 : Semantic Web Standards (RDF, RDFS, OWL, SPARQL)
    - Part 2 : Semantics and Reasoning for Semantic Web
    - Projects : Querying Semantic Data, Ontology Reasoning
  - Grading : average(Projects, Exam)

Outline

Why Web and Semantic Web ?

The Current Web

- WWW is penetrating our society (immensely successful).
- Huge amounts of data
- Syntax standards for transfer of structured data
- Machine-processable, human-readable documents

BUT

- Content/knowledge cannot be accessed by machines.
- Meaning (semantics) of transferred data is not accessible.

Limits of the current Web

- Too much information with too little structure made for human consumption
  - Content search is very simplistic
  - future requires better methods
- Web content is heterogeneous
  - in terms of content
  - in terms of structure
  - in terms of character encoding
  - future requires intelligent information integration
- Humans can derive new (implicit) information from given pieces of information but the current Web can only deal with syntax.
  - requires automated reasoning and data exploration techniques

Limits of the current Web

- Too much information with too little structure made for human consumption
  - Content search is very simplistic
  - future requires better methods
- Web content is heterogeneous
  - in terms of content
  - in terms of structure
  - in terms of character encoding
  - future requires intelligent information integration
- Humans can derive new (implicit) information from given pieces of information but the current Web can only deal with syntax.
  - requires automated reasoning and data exploration techniques

Limits of the current Web

- Too much information with too little structure made for human consumption
  - Content search is very simplistic
  - future requires better methods
- Web content is heterogeneous
  - in terms of content
  - in terms of structure
  - in terms of character encoding
  - future requires intelligent information integration
- Humans can derive new (implicit) information from given pieces of information but the current Web can only deal with syntax.
  - requires automated reasoning and data exploration techniques

Examples

- Find that landmark article on data integration written by an Indian researcher in the 1990s.

  [How can you get the answer ?]

- Are lobsters spiders ?

  [This is getting easier these days, but was impossible a few years ago. It still needs finding and integrating over different websites, as well as some background knowledge.]

- Which car is called a "duck" in German ?

  [This needs some intelligent integration of content from different websites plus background knowledge.]

Examples

- Find that landmark article on data integration written by an Indian researcher in the 1990s.

  [How can you get the answer ?]

- Are lobsters spiders ?

  [This is getting easier these days, but was impossible a few years ago. It still needs finding and integrating over different websites, as well as some background knowledge.]

- Which car is called a "duck" in German ?

  [This needs some intelligent integration of content from different websites plus background knowledge.]

Examples

- Find that landmark article on data integration written by an Indian researcher in the 1990s.

  [How can you get the answer ?]

- Are lobsters spiders ?

  [This is getting easier these days, but was impossible a few years ago. It still needs finding and integrating over different websites, as well as some background knowledge.]

- Which car is called a “duck” in German ?

  [This needs some intelligent integration of content from different websites plus background knowledge.]

Examples

- Find that landmark article on data integration written by an Indian researcher in the 1990s.

  [How can you get the answer ?]

- Are lobsters spiders ?

  [This is getting easier these days, but was impossible a few years ago. It still needs finding and integrating over different websites, as well as some background knowledge.]

- Which car is called a “duck” in German ?

  [This needs some intelligent integration of content from different websites plus background knowledge.]

Examples

- Find that landmark article on data integration written by an Indian researcher in the 1990s.

  [How can you get the answer ?]

- Are lobsters spiders ?

  [This is getting easier these days, but was impossible a few years ago. It still needs finding and integrating over different websites, as well as some background knowledge.]

- Which car is called a “duck” in German ?

  [This needs some intelligent integration of content from different websites plus background knowledge.]

Examples

- Find that landmark article on data integration written by an Indian researcher in the 1990s.

  [How can you get the answer ?]

- Are lobsters spiders ?

  [This is getting easier these days, but was impossible a few years ago. It still needs finding and integrating over different websites, as well as some background knowledge.]

- Which car is called a “duck” in German ?

  [This needs some intelligent integration of content from different websites plus background knowledge.]

Another Example

Identify congress members, who have voted "No" on pro environmental legislation in the past four years, with high-pollution industry in their congressional districts.

In principle, all the required knowledge is on the Web – most of it even in machine-readable form.

However, without automated processing and reasoning we cannot obtain a useful answer.

MIT News

Tim Berners-Lee was honored with the Turing Award for his work inventing the World Wide Web, the first web browser, and "the fundamental protocols and algorithms [that allowed] the web to scale."

Photo: Henry Thomas

Tim Berners-Lee wins $1 million Turing Award

CSAIL researcher honored for inventing the web and developing the protocols that spurred its global use.

Adam Conner-Simons | CSAIL
April 4, 2017

Tim Berners-Lee TED talk 2009

Basic ingredients for the Semantic Web

Tim Berners-Lee's three "extremely simple" rules (2009) :

- All kinds of conceptual things, they have names now that start with HTTP.
- If I take one of these HTTP names and I look it up...I will get back some data in a standard format which is kind of useful data that somebody might like to know about that thing, about that event.
- When I get back that information it's not just got somebody's height and weight and when they were born, it's got relationships. And when it has relationships, whenever it expresses a relationship then the other thing that it's related to is given one of those names that starts with HTTP.

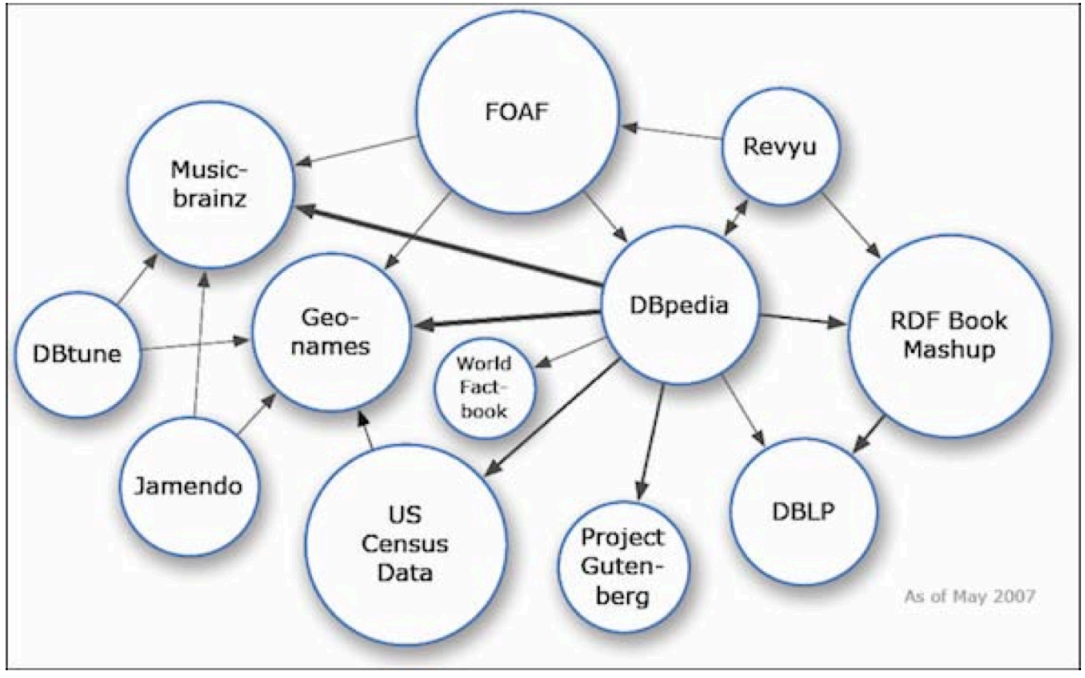
Basic ingredients for the Semantic Web

Four principles of linked data according to Tim Berners-Lee :

- Use URIs (Uniform Resource Identifier) to name (identify) things.
- Use HTTP URIs so that these things can be looked up (interpreted, "dereferenced").
- Provide useful information about what a name identifies when it's looked up, using open standards such as RDF, SPARQL, etc.
- Refer to other things using their HTTP URI-based names when publishing data on the Web.

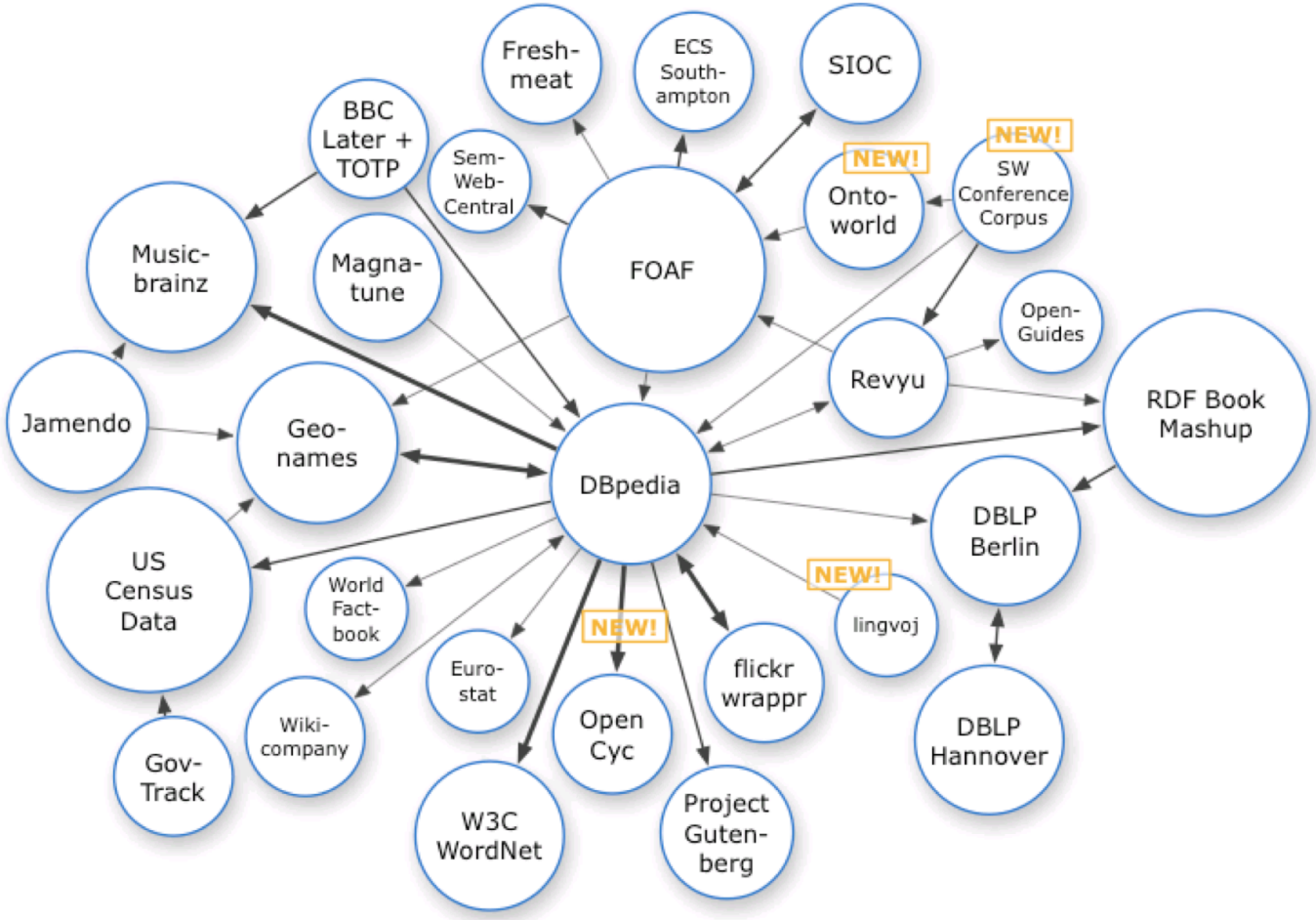
Tim Berners-Lee : "Linked Data", Design Issues. W3C, 2006.

Linked Open Data 2007

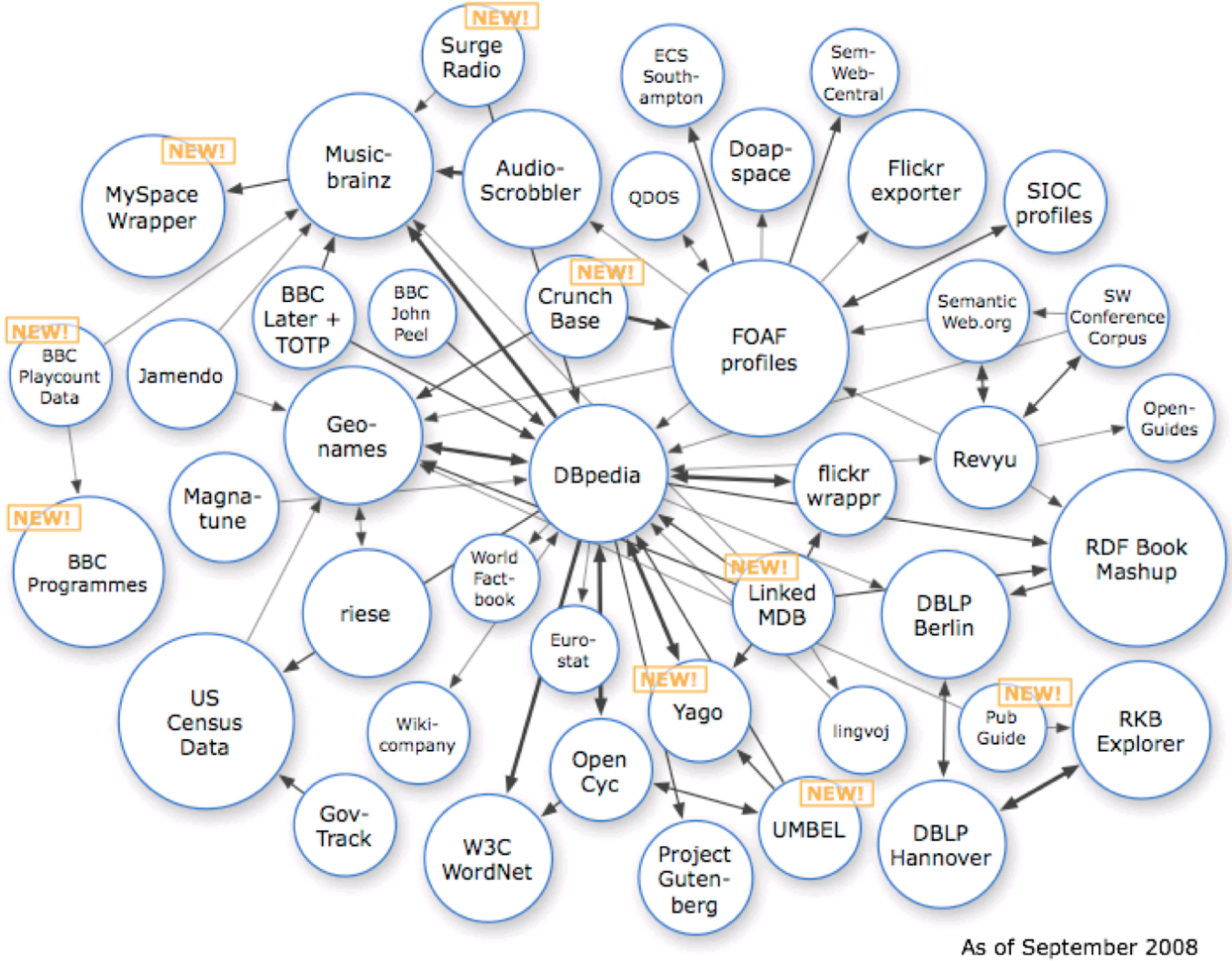


Linking Open Data cloud diagram, this and subsequent pages, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

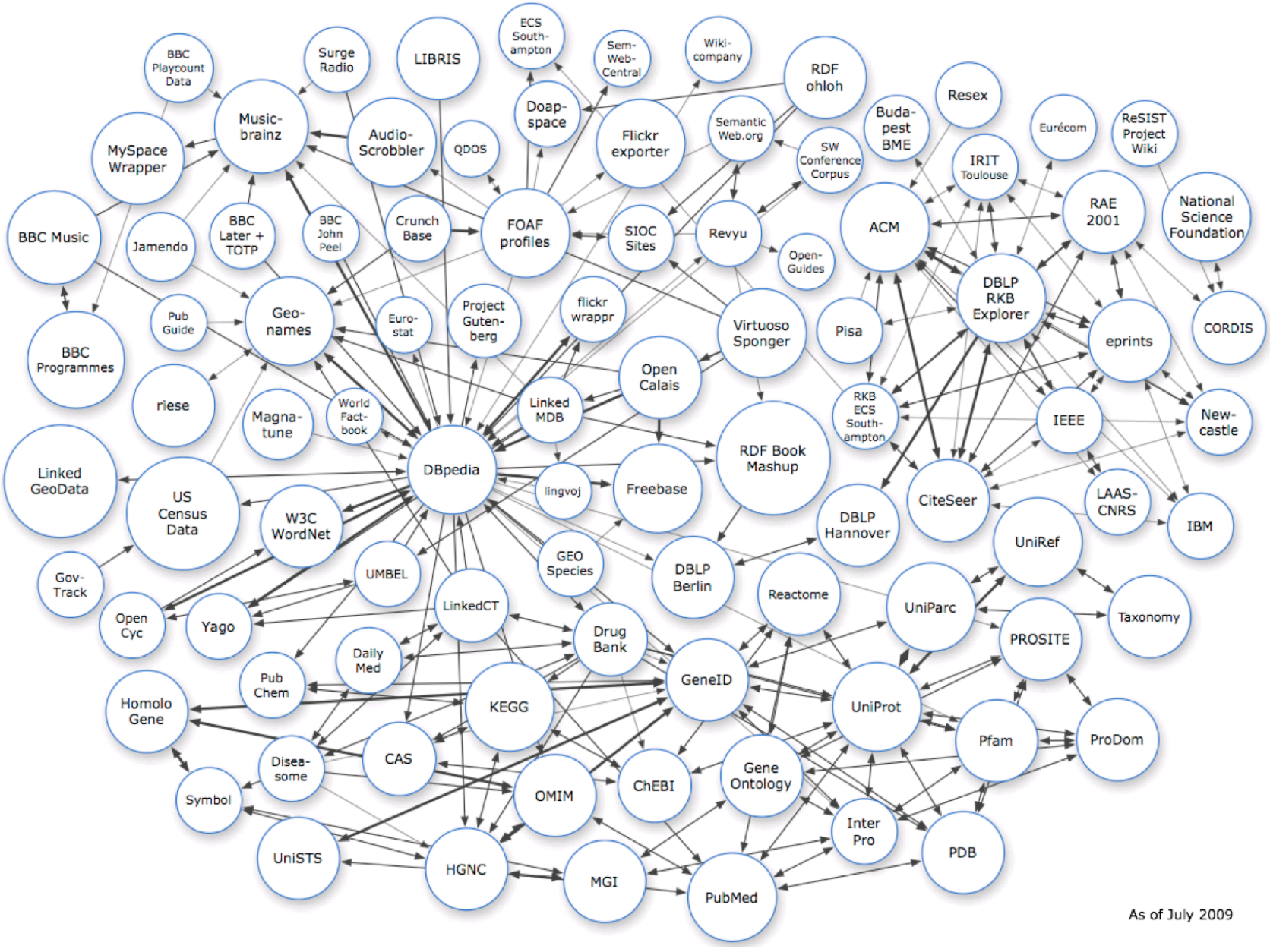
Linked Open Data 2007



Linked Open Data 2008

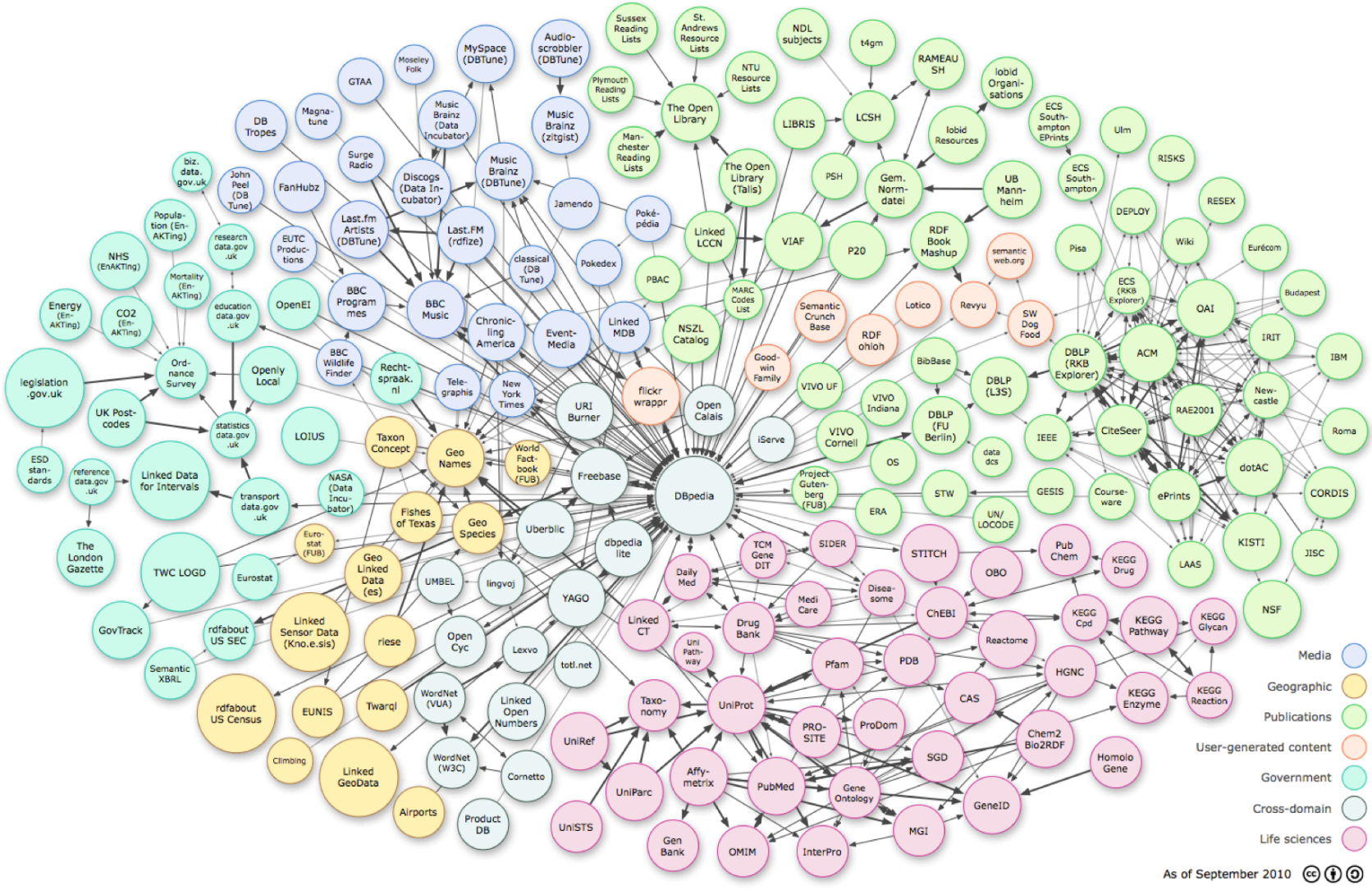


Linked Open Data 2009

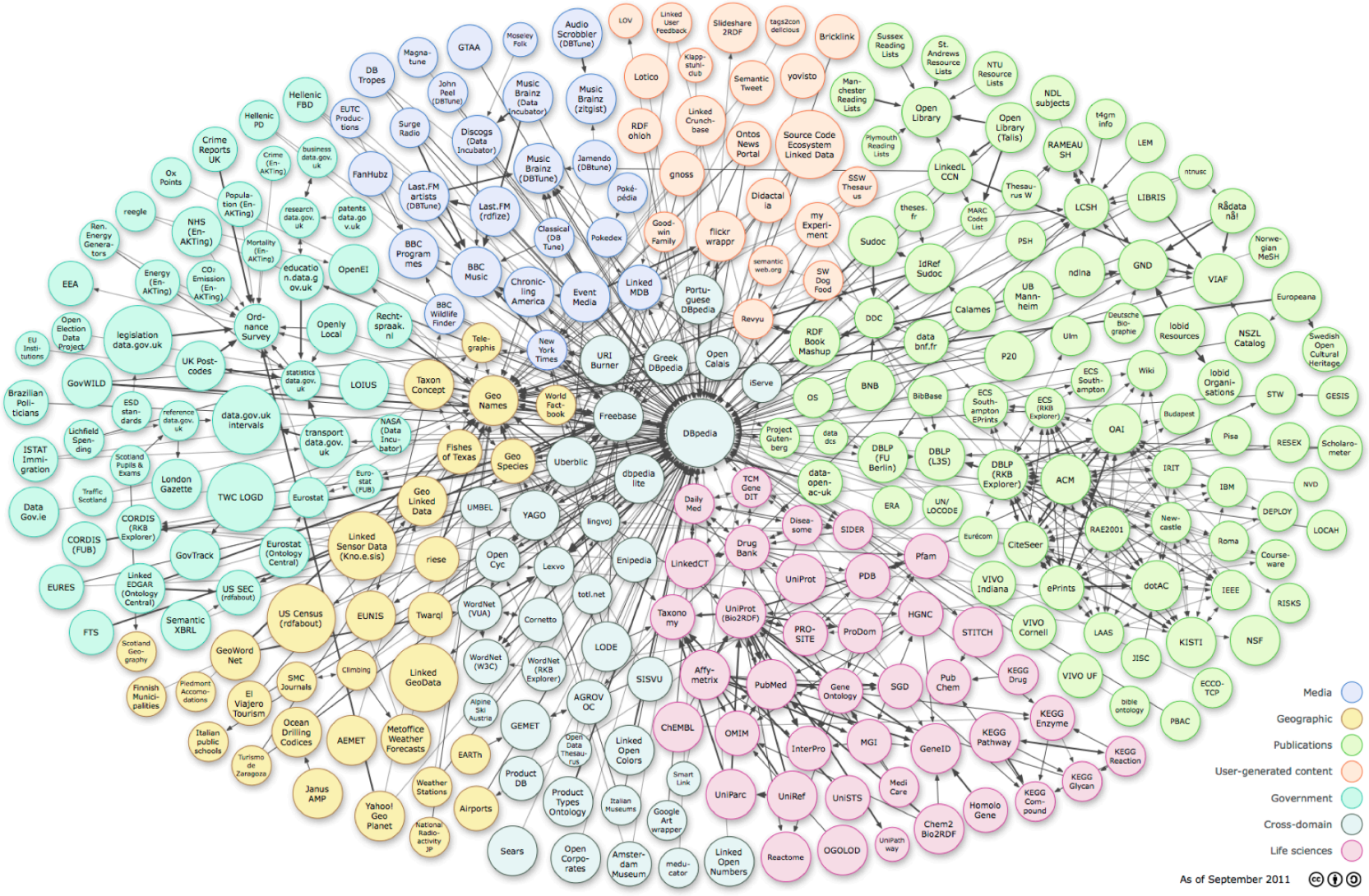


As of July 2009

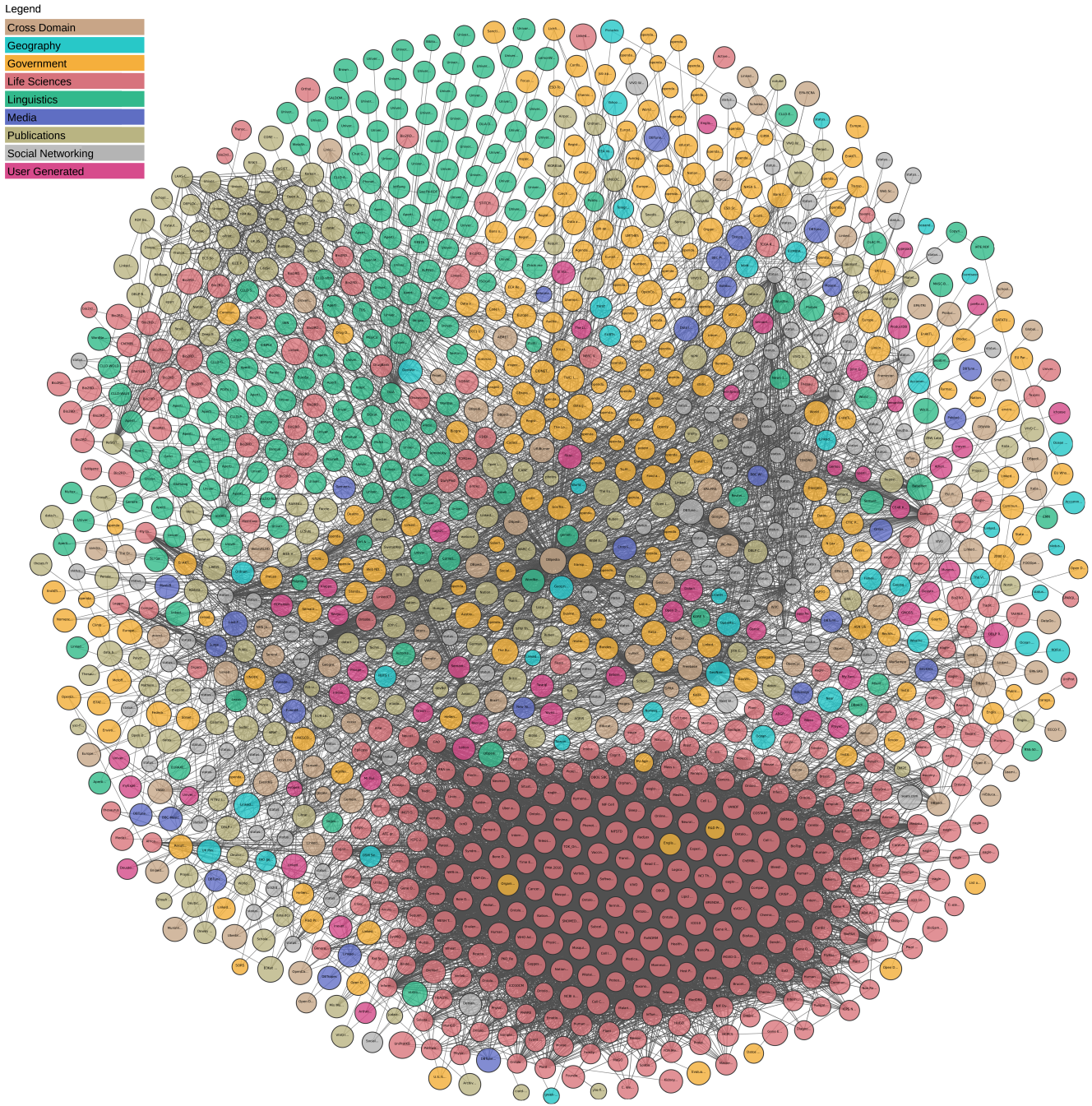
Linked Open Data 2010



Linked Open Data 2011



Linked Open Data 2018



Linked Open Data

By April 30, 2018

- 1,184 Datasets in total
- Domains covered : *Cross-Domain*, *Geography*, *Government*, *Life Sciences*, *Linguistics*, *Media*, *Publications*, *Social Networking*, *User-Generated*

Linked Open Data

Populated Place Features (city, village,...)			
2,518,403	P.PPL	populated place	a city, town, village, or other agglomeration of buildings where people live and work
48,483	P.PPLX	section of populated place	
39,336	P.PPLL	populated locality	an area similar to a locality but with a small group of dwellings or other buildings
13,306	P.PPLQ	abandoned populated place	
2,684	P.PPLA4	seat of a fourth-order administrative division	
2,028	P.PPLA	seat of a first-order administrative division	seat of a first-order administrative division (PPLC takes precedence over PPLA)
1,847	P.PPLW	destroyed populated place	a village, town or city destroyed by a natural disaster, or by war
1,006	P.PPLF	farm village	a populated place where the population is largely engaged in agricultural activities
930	P.PPLA3	seat of a third-order administrative division	
695	P.PPLA2	seat of a second-order administrative division	
253	P.PPLS	populated places	cities, towns, villages, or other agglomerations of buildings where people live and work
249	P.STLMT	israeli settlement	
235	P.PPLC	capital of a political entity	
57	P.		
29	P.PPLR	religious populated place	a populated place whose population is largely engaged in religious occupations
6	P.PPLG	seat of government of a political entity	
2,629,547	Total for P		

rdfs:subClassOf?

XML Problems I

- **How do you encode the piece of knowledge "The book FOST is published by CRC Press"**

- **<book>**
  <title>FOST</title>
  <publisher>CRC Press</publisher>
  </book>

- **<publisher>**
  <name>CRC Press</name>
  <book><title>FOST</title><book>
  </publisher>

- **etc.**

(From P. Hitzler, 2012)

XML Problems II

- **Merging trees is rather cumbersome and the result isn't always clear.**
  - **<publisher>**
    <name>CRC Press</name>
    <book><title>FOST</title><book>
    </publisher>
  - **<book>**
    <title>Semantic Web</title>
    <publisher>Springer</publisher>
    </book>

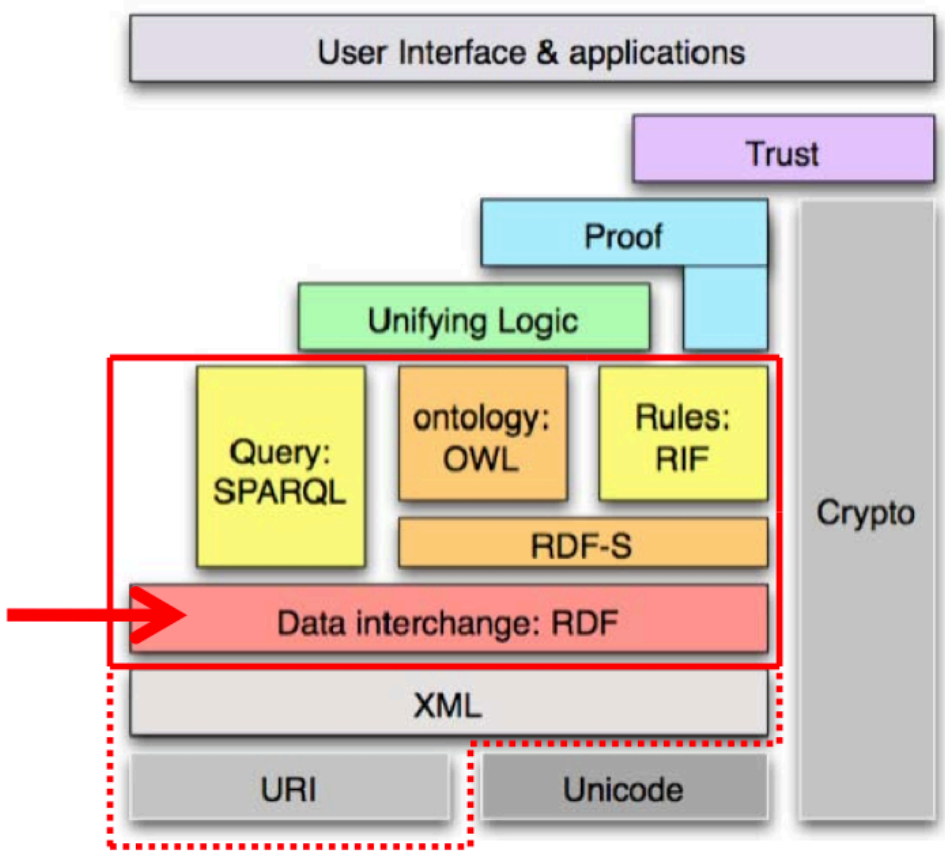
(From P. Hitzler, 2012)

Basic ingredients for the Semantic Web

- Open Standards for describing information on the Web
- Methods for obtaining further information from such descriptions

We'll talk about these matters in this course.

Semantic Web Cake Layer

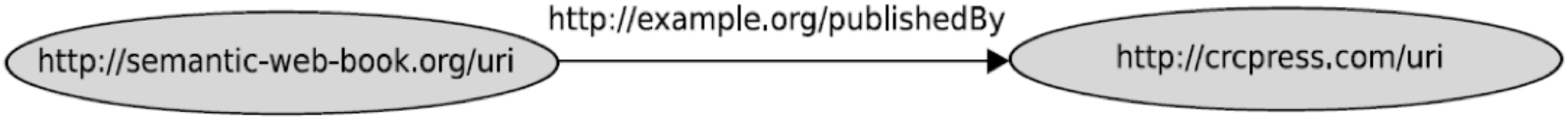


Semantic Web Standards

- RDF
- RDFS
- SPARQL
- OWL

RDF

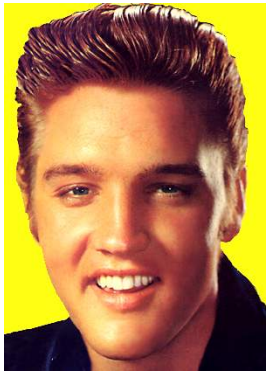
- **Use (directed) graphs as data model**



A collection of such graphs forms a set of semantic data, called Knowledge Base (KB). Each KB can have a name to be reused by others.

Globally identifying entities

KB1

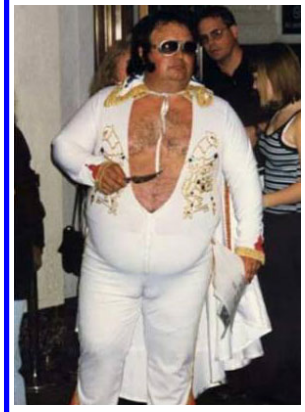


Elvis

KB2

Elvis

KB3



Elvis

KB4

Elvis

Def: Namespace / Qualified Name

A namespace is a named set of (so-called “local”) names.
Wikipedia/Namespace

namespace: KB1
contains local names: Elvis, Priscilla, Lisa

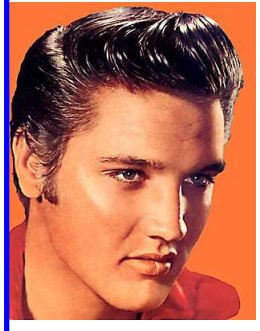
namespace: KB2
contains local names: Elvis, Michael

A qualified name consists of a namespace name and a local name.

KB1:Elvis
KB1:Priscilla
KB2:Elvis

What if KBs have the same name?

ElvisKB



Elvis

ElvisKB

Elvis

ElviPedia

Elvis

ElviPedia

Elvis

Def: URI

A URI (Uniform Resource Identifier) is a string that follows the syntax

<scheme name> : <hierarchical part> [<query>] [& <fragment>]

Examples:

- URLs
  http://elvis.com/biography.html&Birth
- File identifiers
  file:///c:/users/elvis/tripToMoon.txt
- FTP
  ftp://elvis@nsa.gov
- MailTo
  mailto:him@elvis.com?subject=Where%20%are%20you

All URLs are URIs,
but not all URIs
are URLs
("dereferenceable")

We assign a URI to each KB

ElviPedia: http://elvis-alive.org/

ElviPedia': http://elvipedia.com/

ElvisKB: http://elvis.org/kb/

YAGO: http://yago-knowledge.org/

Each of them forms a namespace.

URI of ElviPedia:

http://elvis.org/kb/

Name in that namespace:

Priscilla

Qualified name:

http://elvis.org/kb/Priscilla

(again a URI)

Namespaces

http://elvis.is/king/of/sing

World-wide unique mapping to domain owner

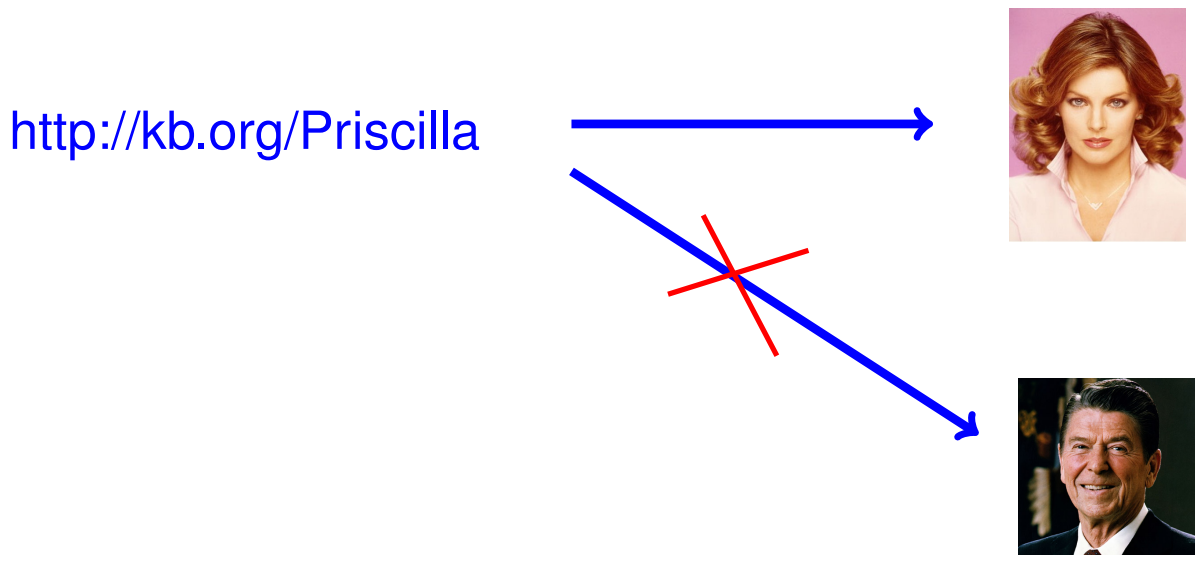
in the responsibility of the domain owner

=> There should be no overlap

- a company can create URIs to identify its products
- an organization can assign sub-domains and each sub-domain can define URIs
- individual people can create URIs from their homepage
- people can create URIs from any URL for which they have exclusive rights to create URIs

URIs are never ambiguous

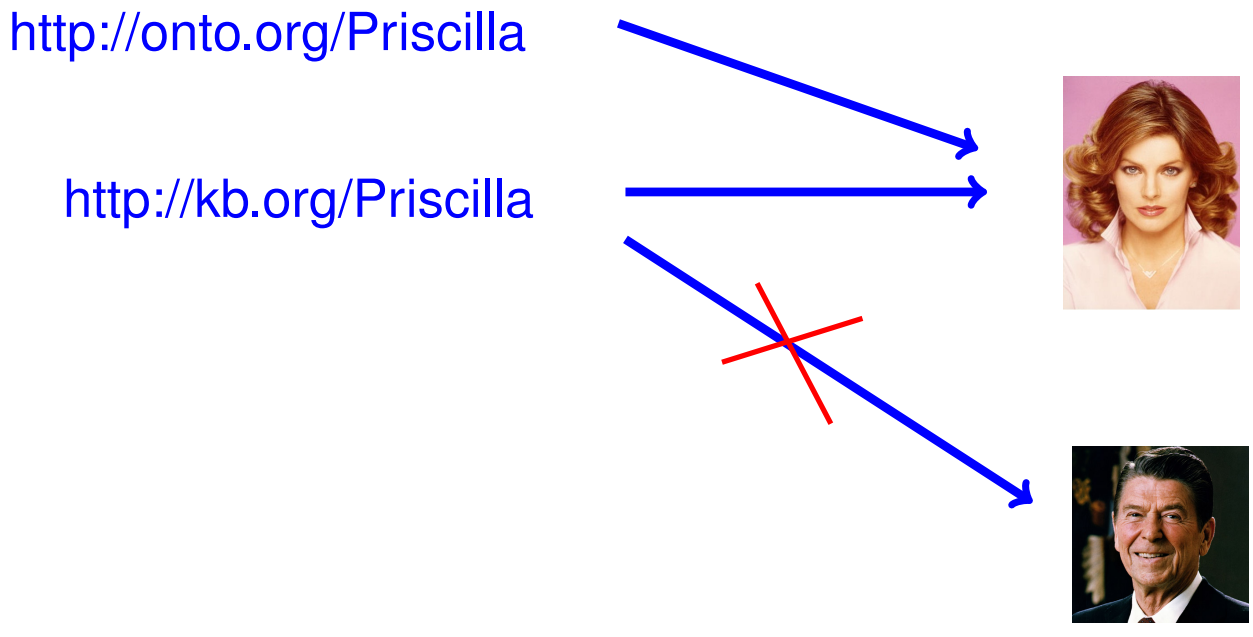
A URI always refers to one entity, never to more entities.



URIs can be synonymous

A URI always refers to one entity, never to more entities.

One entity can be referred to by several URIs.



Def: Namespace prefix, CURIE, base

A namespace prefix is an abbreviation for the first part of a URI.
A prefix with a local name yields a CURIE (also:Qname).

```
@prefix dbp: <http://dbpedia.org/> .

dbp:Elvis
=
<http://dbpedia.org/Elvis>
```

CURIE (Compact URI)
or Qname (qualified name)

A base URI is a URI relative to which URIs in the same document are interpreted.

```
@base <http://yago-knowledge.org/> .
<Elvis> = <http://yago-knowledge.org/Elvis>
```

Def: Turtle

Turtle (Terse RDF Triple Language) is a particular syntax for writing RDF facts.

Turtle can declare namespace prefixes and a base as follows:

@prefix P: <URI> .

@base <URI> .

A simple Turtle fact has the form

URI—Curie URI—Curie URI—Curie—literal .

Example:

@prefix y: <http://yago-knowledge.org/>

y:Elvis y:loves y:Priscilla .

y:Priscilla y:loves <http://kb.org/cake>.

y:Elvis y:isCalled ”The King” .

Each line is a triple of 3 URIs. Each URI identitifies an entity.

The URI in the middle identifies a relation entity.

Each URI can be given explicitly or as a Curie. The object can also be a literal.

Literals with data types

Turtle allows attaching a datatype to a literal in the form

”literal”^^datatype

The datatype is given by a URI or Curie.

It is common to use the XML datatypes

xsd:boolean	true, false
xsd:decimal	Arbitrary-precision decimal numbers
xsd:integer	Arbitrary-size integer numbers IEEE floating-point
xsd:double	64-bit floating point numbers incl. Inf, 0, NaN
xsd:float	32-bit floating point numbers incl. Inf, 0, NaN
xsd:date	Dates (yyyy-mm-dd) with or without timezone
xsd:time	Times (hh:mm:ss.sss. . .) with or without timezone
xsd:dateTime	Date and time with or without timezone
...	

Summary: URIs & Turtle

- URIs are identifiers, often look like URLs

  (IRIs, an extension of URIs, allow internationalized characters)

  http://sing.it/elvis

- Curies abbreviate URIs

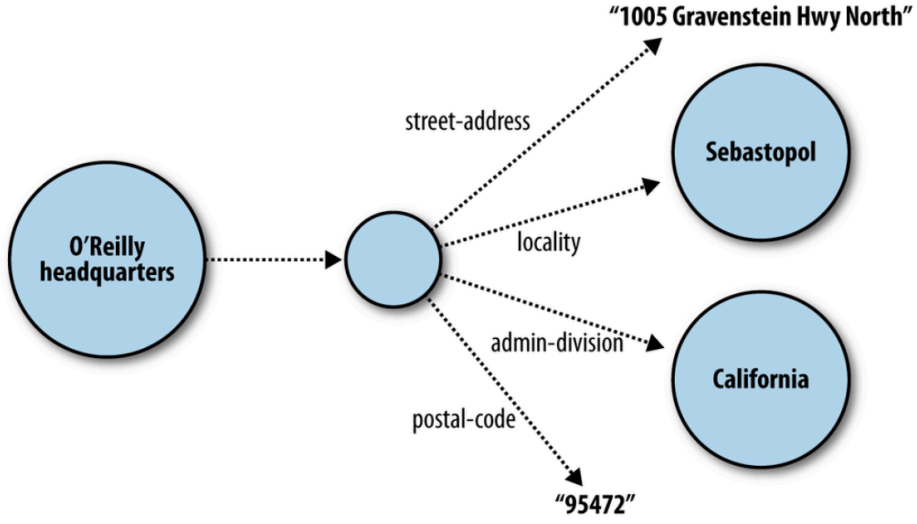
  y:Elvis

- Turtle is a syntax for RDF facts

  <http://kb.org/Elvis> y:sings y:AllShookUp .

  (TriG, an extension of Turtle, allows dealing with named graphs)

Blank Nodes : Example



@prefix ex : <http ://www.lri.fr/sw/example> .
ex :OreillyHeadquarters ex :mailingaddress _ :entity .
_ :entity ex :street-address "1005 Gravenstein Hwy North" .
_ :entity ex :locality ex :Sebastopol .
_ :entity ex :admin-division ex :California .
_ :entity ex :postal-code "95472" .

Blank Nodes

Blank nodes arise from embedded descriptions
RDF graphs may have blank nodes (bnodes, anonymous nodes)

- Node ID used only locally in an RDF graph
- Can be represented in RDF syntax, e.g.(Turtle), as _ :name or []
- No need for an IRI for external reference

Considered an annoying feature but needed, too

- Must be disambiguated when combining graphs
- Names may change when writing/reading graphs

Multi-Valued Relationships

Cooking with RDF :
"For the preparation of Chutney, you need 1 lb green mango, a teaspoon Cayenne pepper, ..."

- first modeling attempt :

@prefix ex : <http ://example.org/> .
ex :Chutney ex :hasIngredient "1lb green mango",
"1 tsp. Cayenne pepper",

....

- Not satisfactory : ingredients plus amounts encoded as one string. Search for recipes containing green mango not direct (or difficult).

Multi-Valued Relationships

Cooking with RDF :
"For the preparation of Chutney, you need 1 lb green mango, a teaspoon Cayenne pepper, ..."

- second modeling attempt :

```
@prefix ex : <http ://example.org/> .
ex :Chutney   ex :hasIngredient    ex :geenMango ;
                  ex :amount    "1 lb" ;
                  ex :hasIngredient    ex :CayennePepper ;
                  ex :amount    "1 tsp." ; ...
```

- Even worse : why ?
- How to solve this modeling problem then ?

Multi-Valued Relationships

Cooking with RDF :
"For the preparation of Chutney, you need 1 lb green mango, a teaspoon Cayenne pepper, ..."

- second modeling attempt :

```
@prefix ex : <http ://example.org/> .
ex :Chutney   ex :hasIngredient    ex :geenMango ;
                  ex :amount    "1 lb" ;
                  ex :hasIngredient    ex :CayennePepper ;
                  ex :amount    "1 tsp." ; ...
```

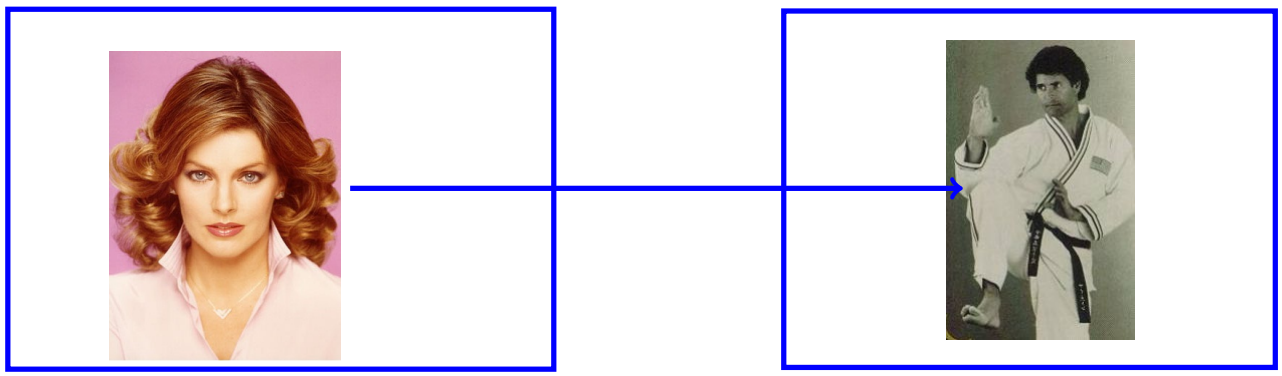
- Even worse : why ?
- How to solve this modeling problem then ? By bnodes !

Cross-referencing

A KB can make statements about entities defined in other KBs.

@prefix y: <http://yago-knowledge.org/>
@prefix d: <http://dbpedia.org/>

y:Priscilla y:loves d:MikeStone .

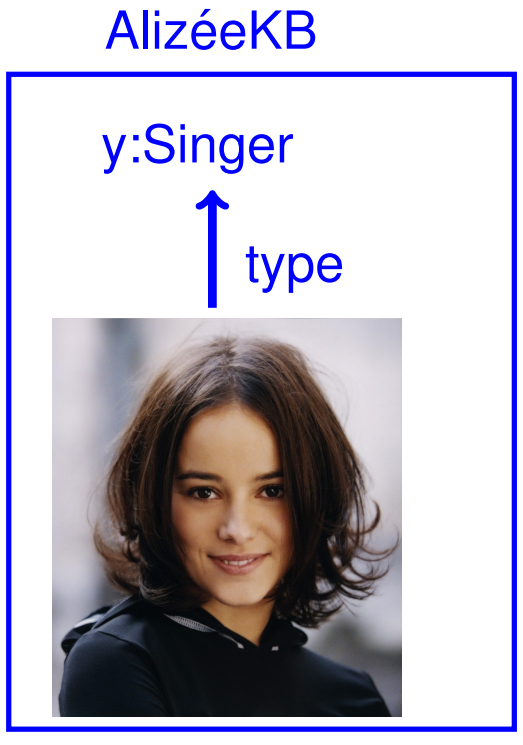


yago
select knowledge

DBpedia

Standard vocabulary

A KB can define vocabulary that is used by other KBs.

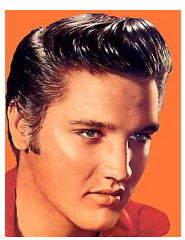


Def: RDF Vocabulary

RDF is also a vocabulary (=KB) that defines basic notions of KB representation.

@prefix rdf: <http://www.w3.org/.../rdf/>
rdf:type, rdf:Property, rdf:Statement ...

We can use notions from this KB:



rdf:type → y:Singer

Popularty of RDF

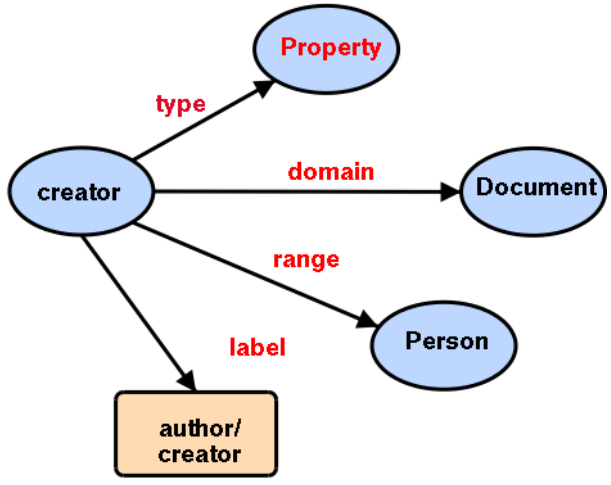
- today, a plethora of RDF tools exists
- there are libraries for virtually all programming languges
- freely available systems to work with large RDF data sets (so-called RDF Stores or Triple Stores)
- also commercial players (like Oracle) support RDF
- RDF is basis for other data formats : RSS 1.0, XMP (Adobe), SVG (vector graphics)

Def: RDFS Vocabulary

RDFS is a vocabulary (=KB) that defines basic notions for class representation.

@prefix rdfs: <http://www.w3.org/.../rdfs/>
rdfs:label, rdfs:subClassOf,
rdfs:domain, rdfs:range,
rdfs:Class, rdfs:Resource ← "entity"

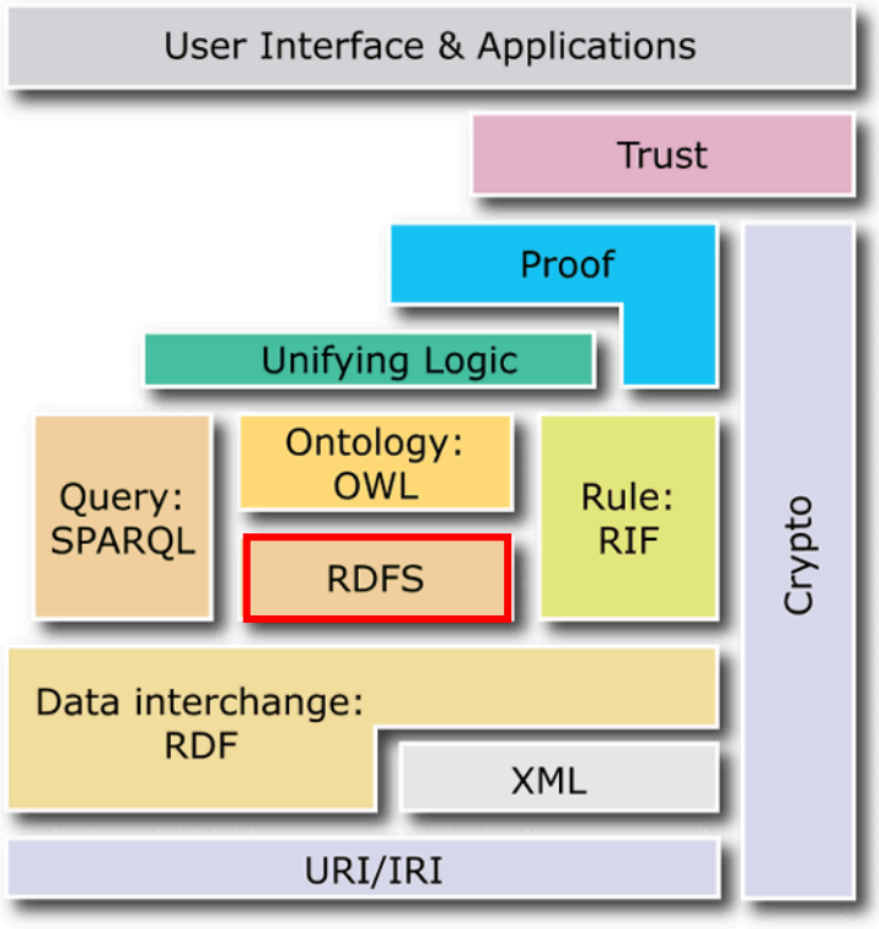
Implicit Information Deduced from RDFS Vocabulary



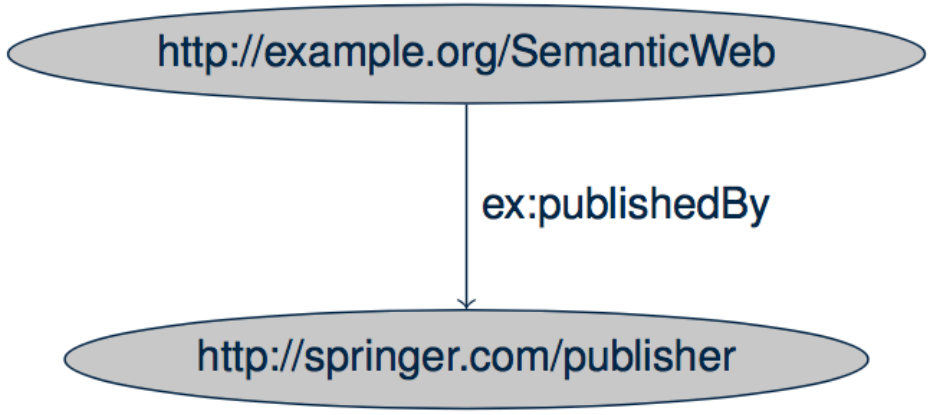
@prefix ex : <http ://www.lri.fr/sw/example> .
ex :LesMiserables ex :creator ex :Hugo .
ex :creator rdf :type rdf :Property .
ex :creator rdfs :domain ex :Document .
ex :creator rdfs :range ex :Person .
ex :creator rdfs :label “author/creator” .

So, we can deduce ex :Hugo is a person and ex :LesMiserables is a document.

RDFS


User Interface & Applications
Trust
Proof
Unifying Logic
Query: SPARQL
Ontology: OWL
RDFS
Rule: RIF
Crypto
Data interchange: RDF
XML
URI/IRI

Motivation



- RDF provides universal possibility to encode factual data on the Web

  = proposition about single resources (individuals) and their relationships
- desirable : propositions about generic sets of individuals (classes), e.g. publishers, organizations, persons etc.

Motivation

- also desirable : specification of logical interdependencies between individuals, classes and relationships to capture as much of the semantics of the described domain as possible, e.g. :

  Publishers are Organizations.

  Only persons write books.
- in database speak : schema knowledge

Schema Knowledge with RDFS

RDF Schema (RDFS) :

- part of the W3C Recommendation of RDF
- allows for specifying schematic (also : terminological) Knowledge
- use of dedicated RDF vocabulary (thus : every RDFS document is an RDF document)
- name space (usually abbreviated with rdfs) :

  http ://www.w3.org/2000/01/rdf-schema#

Classes and Instances

ex :SemanticWeb rdf :type ex :Textbook .

- characterizes Semantic Web as instance of the (newly defined) class “Textbook”
- class membership is not exclusive, e.g. we may have :

  ex :SemanticWeb rdf :type ex :ClassicalTextbook .
- in general : a priori individual and class names cannot be distinguished syntactically
- also in reality, this distinction is sometimes difficult : e.g. for

  http ://www.un.org/#URI

The Class of all Classes

- however, sometimes one wants to state that a URI denotes a class can be done by typing that URI as rdfs :Class

  ex :Textbook rdf :type rdfs :Class .
- rdfs :Class is the class of all classes and therefore also contains itself, thus the following triple is always valid :

  rdfs :Class rdf :type rdfs :Class .

Subclasses Motivation

- given the triple

  ex :SemanticWeb rdf :type ex :Textbook .
- we do not get a result when searching for instances of the class ex :Book
- option : add the triple

  ex :SemanticWeb rdf :type ex :Book .
- this just solves the problem only for the specific resource ex :SemanticWeb
- automatically adding it for all instances would blow up the RDF document

Subclasses

- better : one statement telling that every textbook is also a book, i.e., every instance of ex :Textbook is automatically also an instance of ex :Book
- realized via the rdfs :subClassOf property :

  ex :Textbook rdfs :subClassOf ex :Book .

  The class of all textbooks is a subclass of the class of all books.

Subclasses

- the rdfs :subClassOf property is reflexive, i.e., every class is its own subclass, thus :

  ex :Textbook rdfs :subClassOf ex :Textbook .

- on the contrary, we can enforce that two URIs refer to the same class by declaring them as mutual subclasses, like :

  ex :Haven rdfs :subClassOf ex :Port .

  ex :Port rdfs :subClassOf ex :Haven .

Class Hierarchies

- common : not just singular subclass relationships but whole class hierarchies (aka : taxonomies) e.g. :

  ex :Textbook rdfs :subClassOf ex :Book .

  ex :Book rdfs :subClassOf ex :PrintMedia .

  ex :Journal rdfs :subClassOf ex :PrintMedia .

- "built in" in RDFS semantics : transitivity of the rdfs :subClassOf property, i.e., it follows

  ex :Textbook rdfs :subClassOf ex :PrintMedia .

- class hierarchies particularly often used for modeling, e.g. in biology (e.g. Classification of living beings)

Classes

- intuitive conection to set theory :

  rdf :type corresponds to $\in$

  rdfs :subClassOf corresponds to $\subseteq$
- this also justifies the reflexivity and transitivity of rdfs :subClassOf

Classes in RDF/XML Syntax

- abbreviated notation for specifying class instances:

```
<ex:HomoSapiens rdf:about="&ex;SebastianRudolph"/>
```

instead of

```
<rdf:Description rdf:about="&ex;SebastianRudolph">
  <rdf:type rdf:resource="&ex;HomoSapiens">
</rdf:Description>
```

- Likewise:

```
<rdfs:Class rdf:about="&ex;HomoSapiens"/>
```

Predefined Class URIs

- rdfs :Resource

  class of all resources (i.e., all elements of the domain)
- rdf :Property

  class of all relationships

  (= those resources, that are referenced via predicate URIs)
- rdf :List, rdf :Seq, rdf :Bag, rdf :Alt, rdfs :Container

  diverse kinds of lists
- rdfs :ContainerMembershipProperty

  class of all relationships that represent a containedness relationship

Predefined Class URIs

- rdf :XMLLiteral

  class of all values of the predefined datatype XMLLiteral
- rdfs :Literal

  class of all literal values (every datatype is a subclass of this class)
- rdfs :Datatype

  class of all datatypes (therefore it is a class of classes, similar to rdfs :Class)
- rdf :Statement

  class of all reified propositions

Properties

- also called : relations, relationships
- beware : unlike in OOP, properties in RDF(S) are not assigned to classes
- property URIs normally in predicate position of a triple
- properties characterize, in which way two resources are related to each other
- mathematically often represented as set of pairs :

  $\text{marriedWith} = \{(\text{Adam, Eve}), (\text{Brad, Angelina}), \ldots\}$
- URI can be marked as property name by typing it accordingly :

  ex :publishedBy rdf :type rdf :Property .

Subproperties

- like sub-/superclasses also sub-/superproperties possible and useful
- specification in RDFS via rdfs :subPropertyOf e.g. :

  ex :happilyMarriedWith rdf :subPropertyOf rdf :marriedWith .

  Then, given the triple

  ex :anne ex :happilyMarriedWith ex :thomas .

  we can infer

  ex :anne ex :marriedWith ex :thomas .

rdf :marriedWith should be reflexive? (Not in RDFS, but only in OWL)

Property Restrictions

- common : usage of property only makes sense for certain kinds of resources, e.g. ex :publishedBy only connects publications with publishers
- thus, for all URIs a, b, the triple

  a ex :publishedBy b .

  intuitively entails :

  a rdf :type ex :Publication .

  b rdf :type ex :Publisher .
- We can express this directly in RDFS :

  ex :publishedBy rdfs :domain ex :Publication .

  ex :publishedBy rdfs :range ex :Publisher .
- Can also be used to prescribe datatypes for literals :

  ex :hasAge rdfs :range xsd :nonNegativeInteger .

Property Restrictions

- property restrictions are the only way of specifying semantic interdependencies between properties and classes
- beware : property restrictions are interpreted globally and conjunctively :

  ex :authorOf rdfs :range ex :Cookbook .

  ex :authorOf rdfs :range ex :Storybook .

  means : every entity having an author is both a cookbook and a storybook
- thus : always pick the most general possible class for domain/range specifications

Comments in RDFS

- like with programming languages, one sometimes wants to add comments (without changing the semantics)
- purpose : increase understandability for human users
- thus : defined set of properties that serve this purpose

Comments in RDFS

rdfs :label

- property that assigns a name (Literal) to an arbitrary resource
- often, URIs themselves are difficult to read, or bulky at best
- names provided via rdfs :label are often used by tools that graphically represent the data

example (also feat. language information) :

```
<rdfs :Class rdf :about="&ex;Hominidae">
  <rdfs :label xml :lang="en">great apes</rdfs :label>
</rdfs :Class>
```

Comments in RDFS

rdfs :comment

- property assigning an extensive comment (literal) to an arbitrary resource
- may e.g. contain the natural language description of a newly introduced class this facilitates later usage

rdfs :seeAlso, rdfs :definedBy

- properties giving resources (URIs !) where one can find further information or a definition of the subject resource

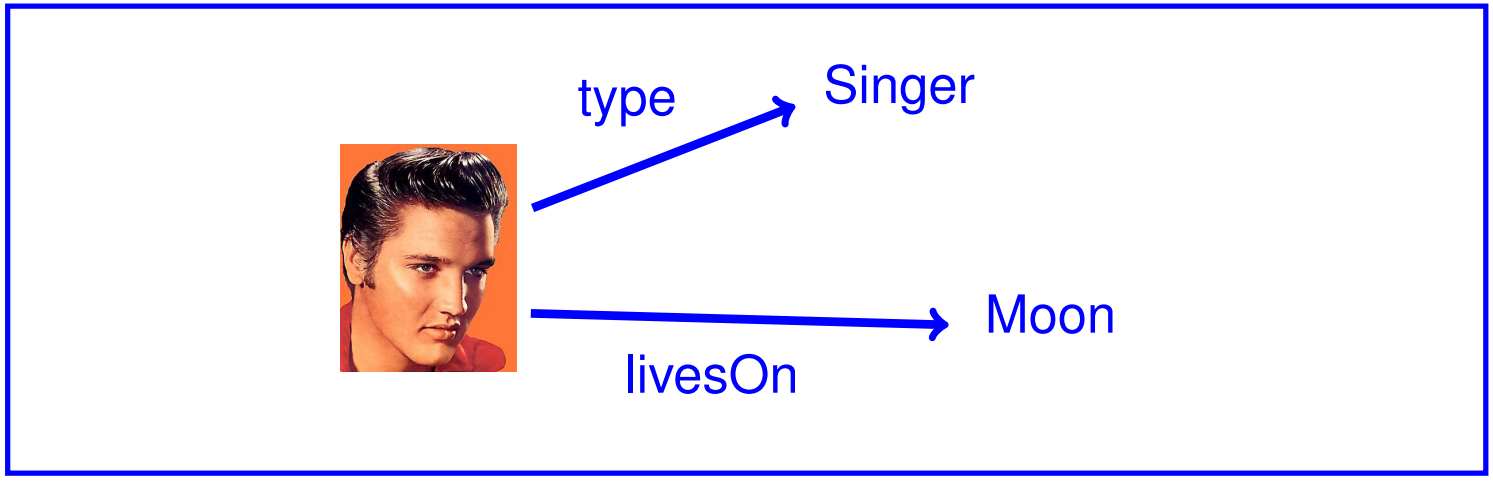
Example

```
xmlns:wikipedia="http://en.wikipedia.org/wiki" : <rdfs:Class
rdf:about="\&ex;Primates">
  <rdfs:label xml:lang="de">Primaten</rdfs:label>
  <rdfs:comment>
    An order of mammals. Primates are characterized by a highly
    developed brain. Most primates live in tropical or subtropical
    regions.
  </rdfs:comment>
  <rdfs:seeAlso rdfs:resource="/&wikipedia;Primate"/>
  <rdfs:subClassOf rdfs:resource="\&ex;Mammalia"/>
</rdfs:Class>
```

Task: Turtle & RDF

Write the following facts in Turtle,
using RDF vocabulary where possible.

URI of KB: <http://whatyoushouldknow.org/>



Start with

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns>

More vocabularies

- Dublin Core (for describing documents)

  http://purl.org/dc/elements/1.1/
- Schema.org (for Web content)

  http://schema.org
- Creative Commons (types of licences)

  http://creativecommons.org/ns&
- Facebook Open Graph (for Web content)

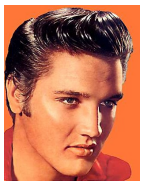
  http://ogp.me/

Dublin Core

Dublin Core is a vocabulary (=KB) of terms (=entities)
for describing documents.

dc:creator, dc:title, dc:format,
dc:MediaType, dc:language...

see this KB



dc:creator

“Elvis: An auto-biography”

dc:title

dc:issued

“1980”

dc:description

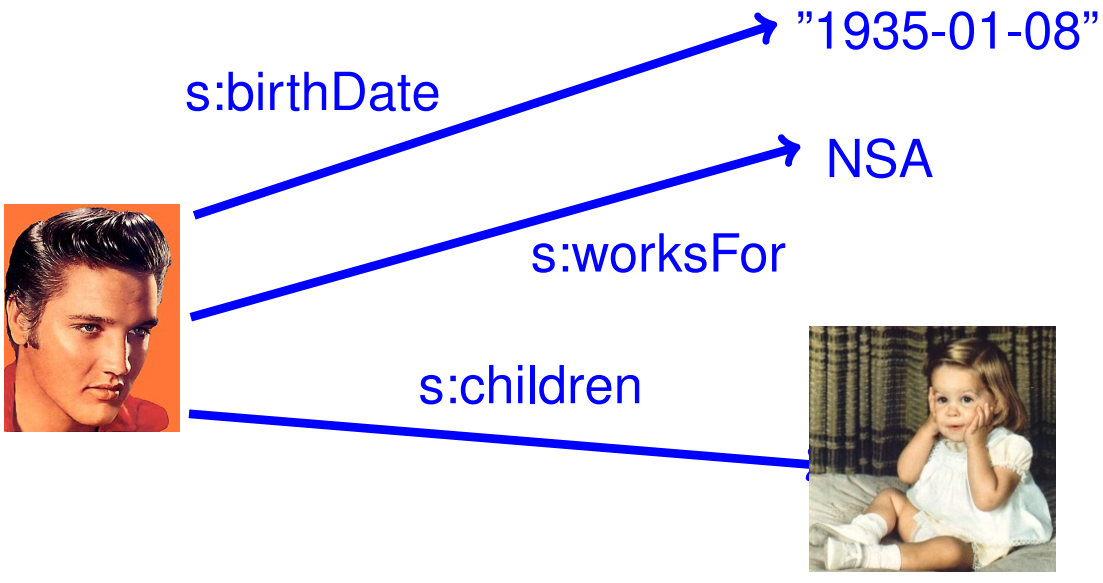
“All about
my life&wife”

Schema.org

Schema.org is a KB by Google, Yahoo & Microsoft for describing Web content.

s:Person, s:Movie, s:address,
s:follows, s:worksFor, ...

see this KB

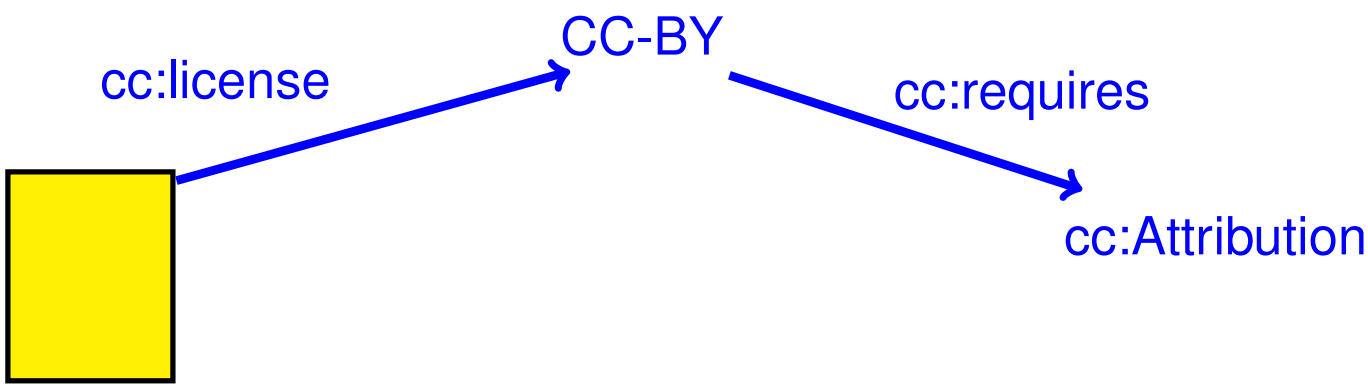


Creative Commons

Creative Commons provides their vocabulary in RDF.

cc:license, cc:attributionName,
cc:permits, cc:Reproduction, ...

see this KB



Graph Definitions

- An RDF triple consists of three components :
  - the subject, which can be an IRI or a bnode,
  - the predicate, which has to be an IRI, and
  - the object, which can be an IRI, a bnode or a Literal.
- The predicate is also denoted as property.
- An RDF graph (or simply graph) is a set of RDF triples. The graph nodes are the subjects and objects of these triples.
- A (proper) subgraph of an RDF graph is a (proper) subset of its triples.
- A ground graph is an RDF graph without blank nodes.

SPARQL : The RDF Query Language

SPARQL is the standard language to query graph data represented as RDF triples.

- SPARQL Protocol and RDF Query Language
- One of the three core standards of the Semantic Web, along with RDF and OWL.
- Became a W3C standard January 2008.
- SPARQL 1.1 is a W3C Recommendation since March 2013.

Types of SPARQL queries

- SELECT

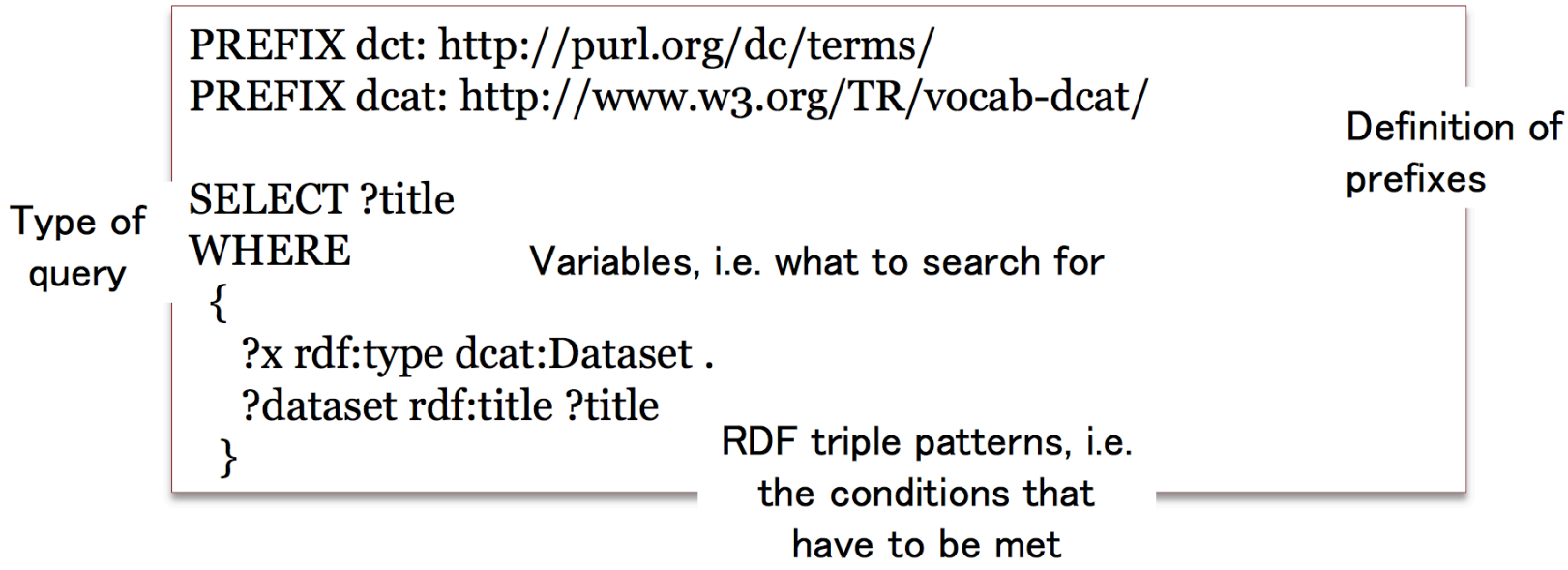
  Return a table of all X, Y, etc. satisfying the following conditions ...
- CONSTRUCT

  Find all X, Y, etc. satisfying the following conditions ... and substitute them into the following template in order to generate (possibly new) RDF statements, creating a new graph.
- DESCRIBE

  Find all statements in the dataset that provide information about the following resource(s) ... (identified by name or description)
- ASK

  Are there any X, Y, etc. satisfying the following conditions ...

Structure of a SPARQL Query



SPARQL : Examples

- Find out the distint properties used in DBpedia :

  select distinct ?p where { ?x ?p ?y.}

SPARQL : Examples

- Find out the distint properties used in DBpedia :

  select distinct ?p where { ?x ?p ?y.}

SPARQL : Examples

- Find out the distint properties used in DBpedia :

  select distinct ?p where { ?x ?p ?y.}

Try this query over http ://dbpedia.org/sparql with format CSV, among the results of which you will find :
"http ://dbpedia.org/property/allProducer". Remark. By choosing the format Turtle, you'll find dbp :allProducer (So Prefix dbp : <http ://dbpedia.org/property/> is http ://dbpedia.org/sparql by defaut).

SPARQL : Examples

- Find out the distint properties used in DBpedia :

  select distinct ?p where { ?x ?p ?y.}

Then you can use the property dbp :allProducer to get more ideas of the data :

select distinct ?x ?p where { ?x dbp :allProducer ?p} LIMIT 100

Getting to know a Linked Data set

What triples are there in a triple store ?

```
Select ?x ?y ?z where { ?x ?y ?z .}
```

SPARQL example

Q : who are born in Europe and received a Nobel Prize ?

SPARQL example

Q : who are born in Europe and received a Nobel Prize ?

SPARQL query :

```
SELECT ?x WHERE {
        ?y :hasName ?x ;
        ?y :hasWonPrize :NobelPrize ;
        ?y :bornIn :Europe.
        }
```

SPARQL example

Q : who are born in Europe and received a Nobel Prize ?

SPARQL query :

```
SELECT ?x WHERE {
        ?y :hasName ?x ;
        ?y :hasWonPrize :NobelPrize ;
        ?y :bornIn :Europe.
        }
```

RDF triple store :

<http ://.../marie-curie, hasName, “Marie Curie”>
<http ://.../marie-curie, type, Scientist>
<http ://.../marie-curie, hasWonPrize, NobelPrize>
<http ://.../marie-curie, bornIn, Europe>
<http ://.../schmidt, hasName, “Brian P. Schmidt”>
<http ://.../Schmidt, type, Physicist>
<http ://.../schmidt, hasWonPrize, NobelPrize>
<http ://.../einstein, bornIn, Australie>

SPARQL example

Q : who are born in Europe and received a Nobel Prize ?

SPARQL query :

```
SELECT ?x WHERE {
        ?y :hasName ?x ;
        ?y :hasWonPrize :NobelPrize ;
        ?y :bornIn :Europe.
        }
```

RDF triple store :

<http ://.../marie-curie, hasName, “Marie Curie”>
<http ://.../marie-curie, type, Scientist>
<http ://.../marie-curie, hasWonPrize, NobelPrize>
<http ://.../marie-curie, bornIn, Europe>
<http ://.../schmidt, hasName, “Brian P. Schmidt”>
<http ://.../Schmidt, type, Physicist>
<http ://.../schmidt, hasWonPrize, NobelPrize>
<http ://.../einstein, bornIn, Australie>

SPARQL example

Q : who are born in Europe and received a Nobel Prize ?

SPARQL query :

```
SELECT ?x WHERE {
        ?y :hasName ?x ;
        ?y :hasWonPrize :NobelPrize ;
        ?y :bornIn :Europe.
        }
```

A : “Marie Curie ”

RDF triple store :

<http ://.../marie-curie, hasName, “Marie Curie”>
<http ://.../marie-curie, type, Scientist>
<http ://.../marie-curie, hasWonPrize, NobelPrize>
<http ://.../marie-curie, bornIn, Europe>
<http ://.../schmidt, hasName, “Brian P. Schmidt”>
<http ://.../Schmidt, type, Physicist>
<http ://.../schmidt, hasWonPrize, NobelPrize>
<http ://.../einstein, bornIn, Australie>

SPARQL example

Q : who are born in Europe and received a Nobel Prize ?

SPARQL query :

```
SELECT ?x WHERE {
                ?y :hasName ?x ;
                ?y :hasWonPrize :NobelPrize .
  OPTIONAL{     ?y :bornIn :Europe. }
                }
```

A : “Marie Curie ” “Brian P. Schmidt”

RDF triple store :

<http ://.../marie-curie, hasName, “Marie Curie”>
<http ://.../marie-curie, type, Scientist>
<http ://.../marie-curie, hasWonPrize, NobelPrize>
<http ://.../marie-curie, bornIn, Europe>
<http ://.../schmidt, hasName, “Brian P. Schmidt”>
<http ://.../Schmidt, type, Physicist>
<http ://.../schmidt, hasWonPrize, NobelPrize>
<http ://.../einstein, bornIn, Australie>