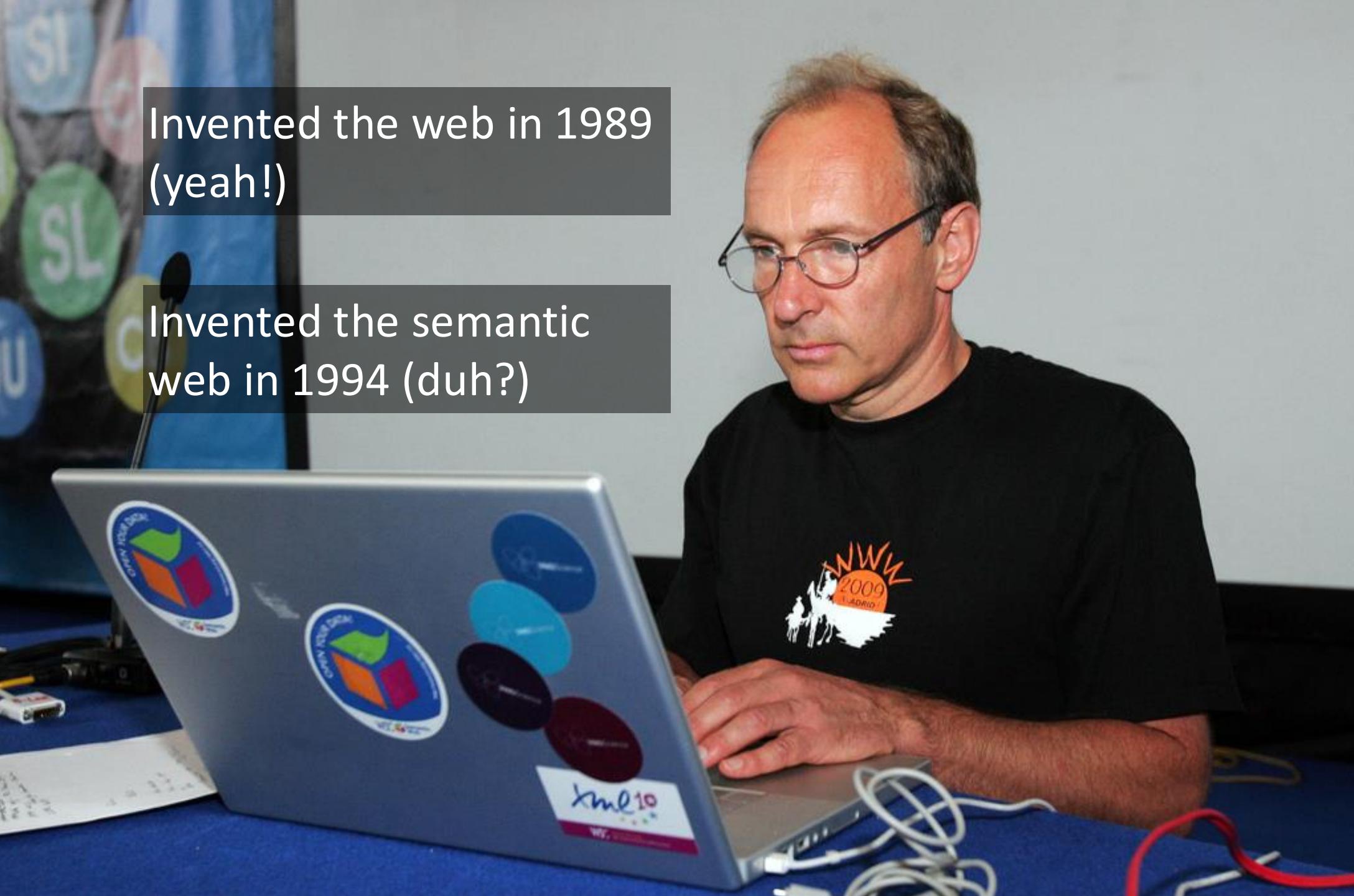


Introduction to Knowledge Graphs

Enrico Daga
@ Knowledge Media Institute,
The Open University, UK
<https://www.enridaga.net/>

His research is exploring novel methods for Knowledge Graph construction and infrastructure, with particular focus on management, architectures, policies and process knowledge and applications to smart cities, healthcare, humanities, and cultural heritage domains.





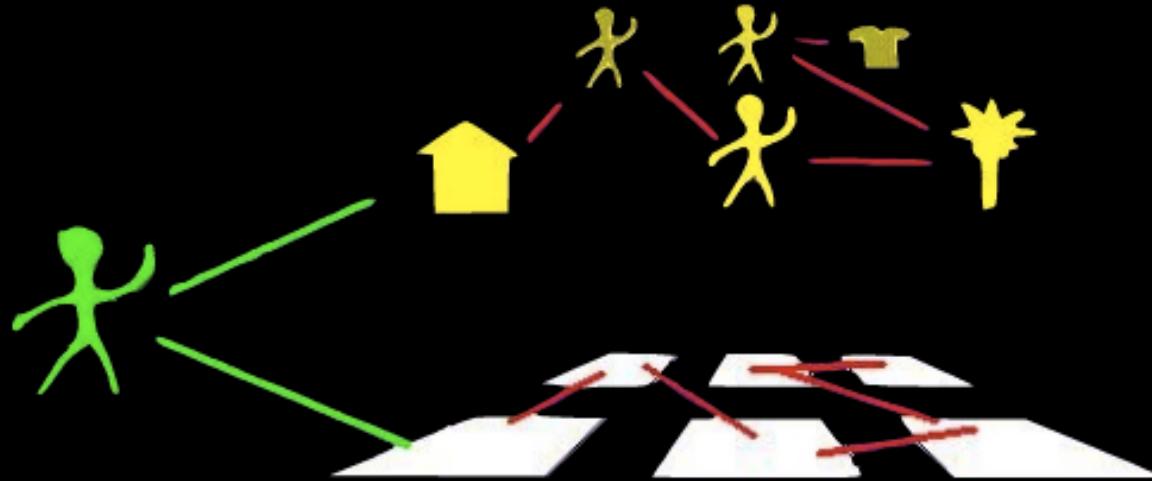
Invented the web in 1989
(yeah!)

Invented the semantic
web in 1994 (duh?)

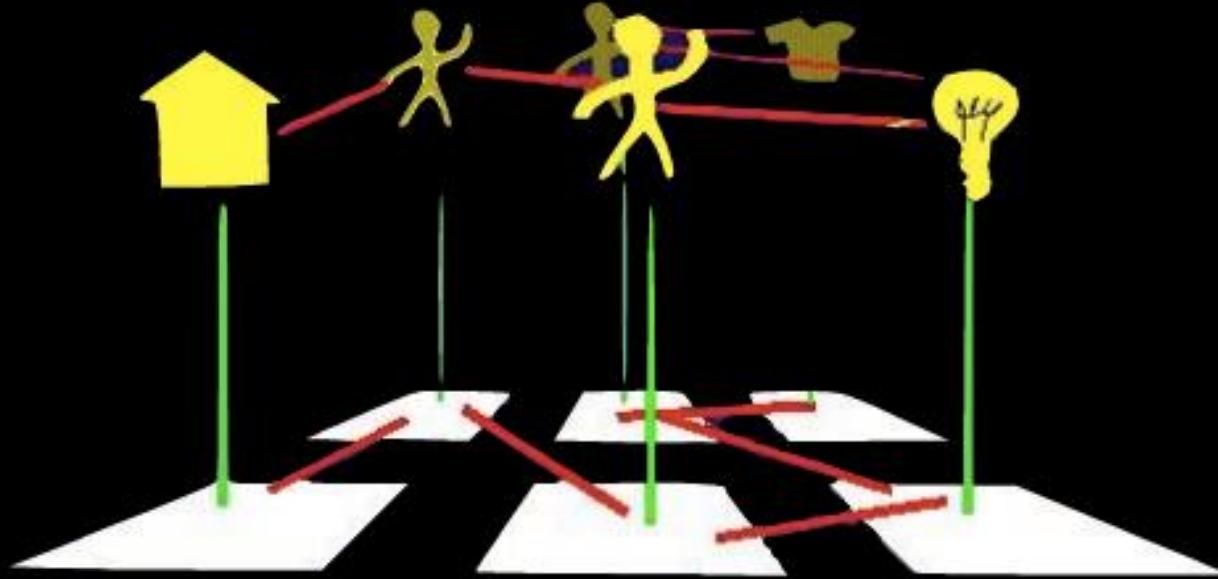


“To a computer, then, the web is a **flat**,
boring world devoid of **meaning**”

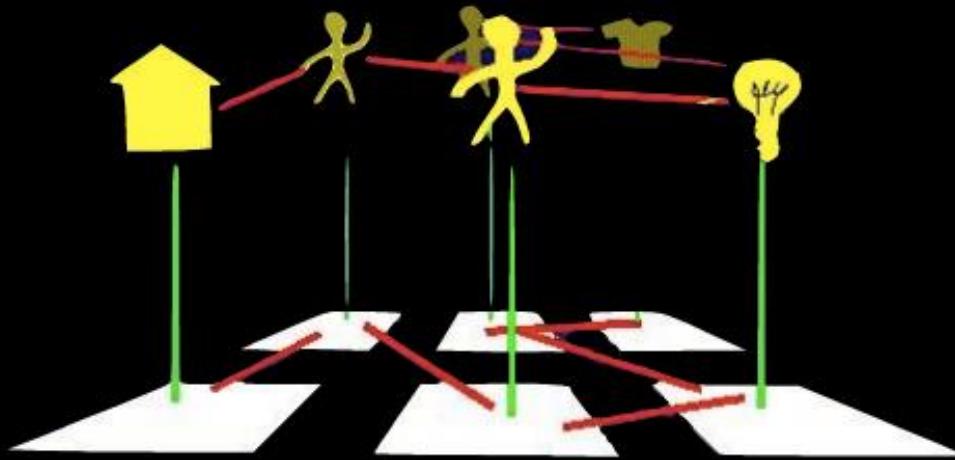
Tim Berners Lee, <http://www.w3.org/Talks/WWW94Tim/>



“This is a pity, as in fact **documents** on the web describe **real objects** and imaginary **concepts**, and give particular **relationships** between them”



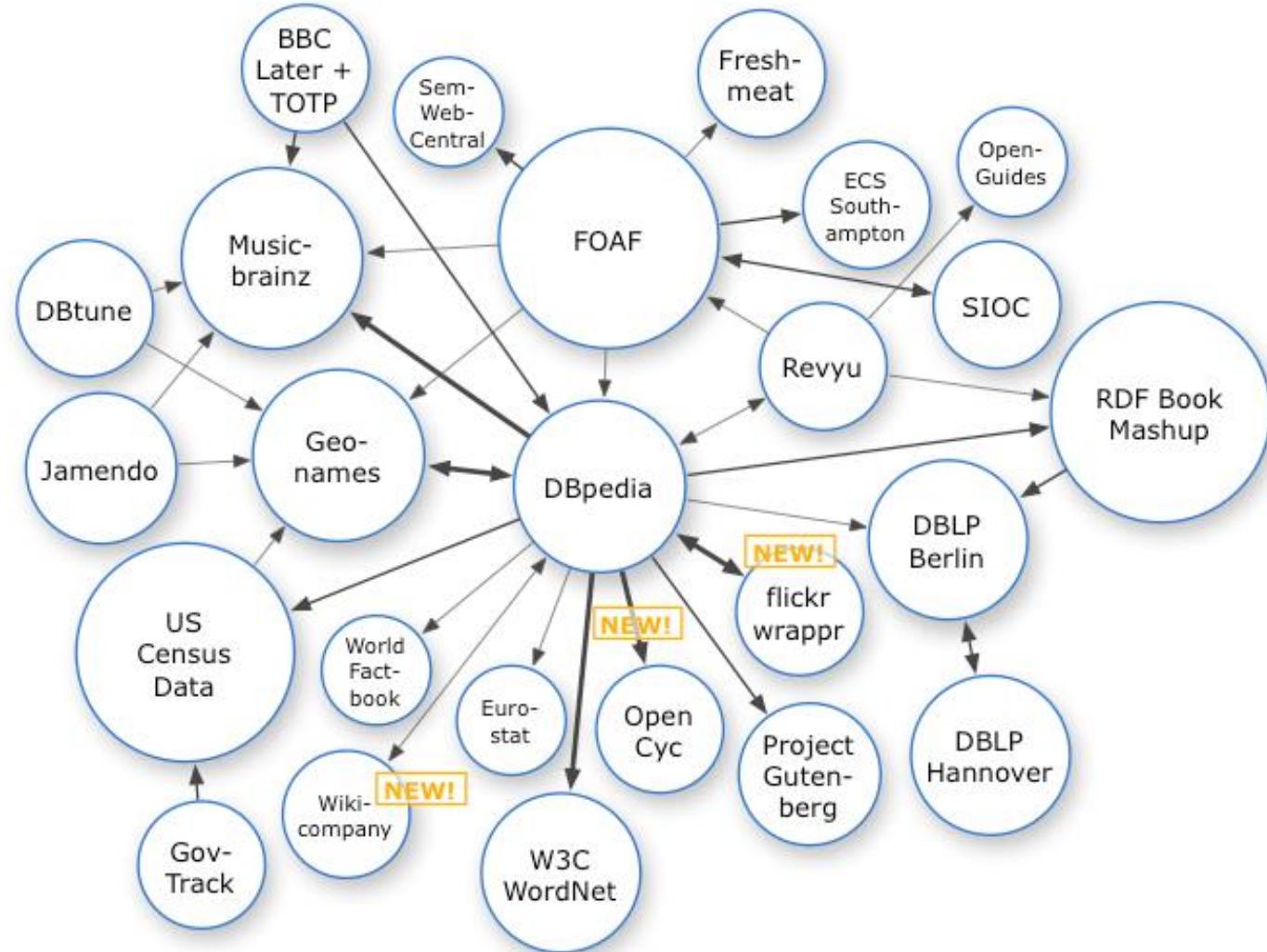
“Adding semantics to the web involves two things: allowing **documents** which have information in **machine-readable** forms, and allowing **links** to be created with **relationship values**.”



“The Semantic Web is not a separate Web but an **extension** of the current one, in which information is given well-defined **meaning**, better enabling **computers and people** to work in cooperation.”

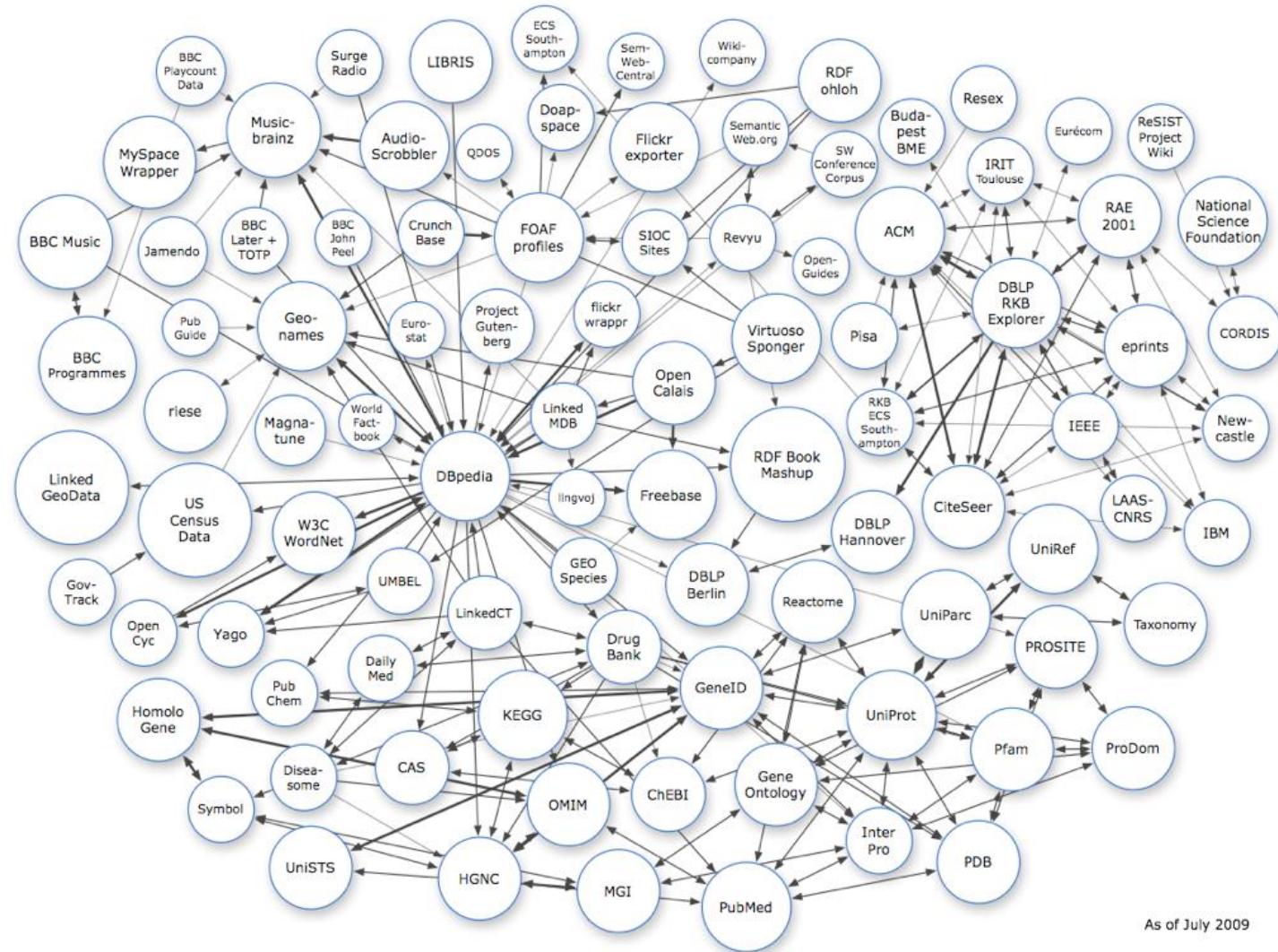
Linked Open Data in 2007

- “Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch.
[http://lod-cloud.net/”](http://lod-cloud.net/)



2009

- <https://lod-cloud.net/>





Linked Data: The story so far (2009)

Linked Data - The Story So Far

Christian Bizer, Freie Universität Berlin, Germany
Tom Heath, Talis Information Ltd, United Kingdom
Tim Berners-Lee, Massachusetts Institute of Technology, USA

This is a preprint of a paper to appear in: Heath, T., Hepp, M., and Bizer, C. (eds.). Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS). <http://linkeddata.org/docs/ijswis-special-issue>

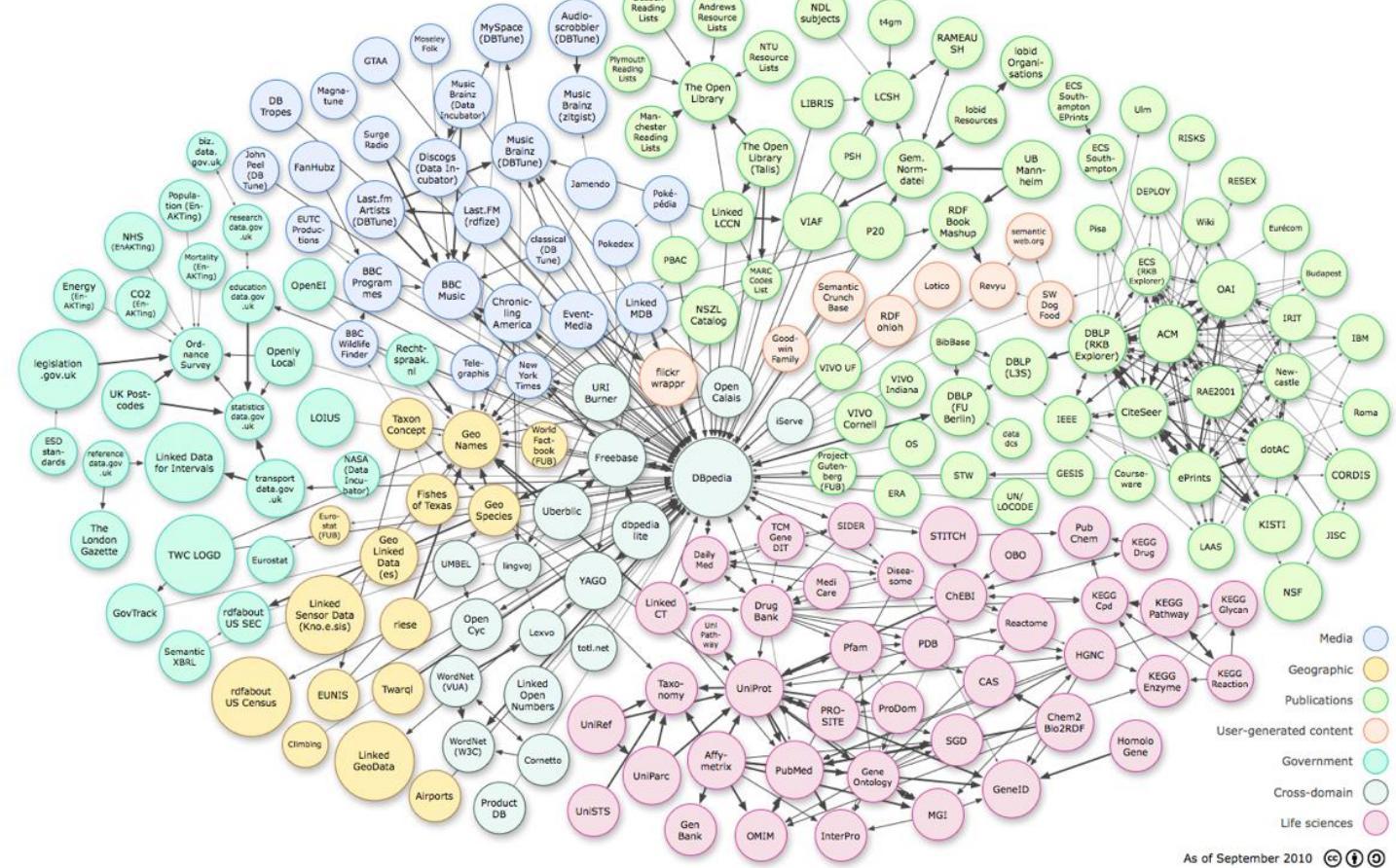
Abstract

The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the last three years, leading to the creation of a global data space containing billions of assertions - the Web of Data. In this article we present the concept and technical principles of Linked Data, and situate these within the broader context of related technological developments. We describe progress to date in publishing Linked Data on the Web, review applications that have been developed to exploit the Web of Data, and map out a research agenda for the Linked Data community as it moves forward.

Keywords: Linked Data, Web of Data, Semantic Web, Data Sharing, Data Exploration

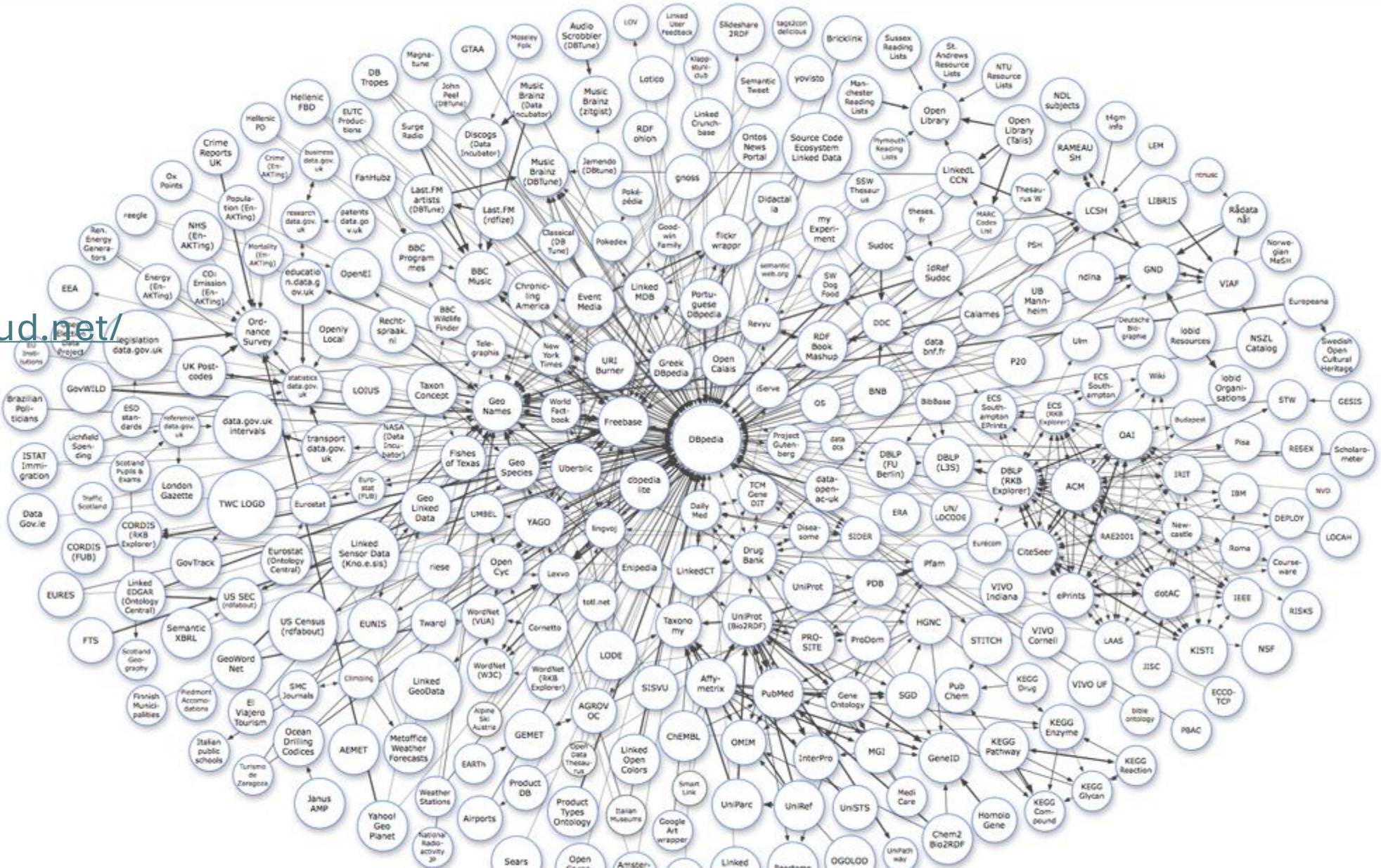
2010

- <https://lod-cloud.net/>



2011

- <https://lod-cloud.net/>





Google Knowledge Graph

中文 23 languages ▾

Article Talk

Read Edit View history Tools ▾

From Wikipedia, the free encyclopedia

"Knowledge Graph" redirects here. For the general concept in information science, see [Knowledge graph](#). For other uses, see [Knowledge Graph \(disambiguation\)](#).

The **Google Knowledge Graph** is a [knowledge base](#) from which Google serves relevant information in an infobox beside its [search results](#). This allows the user to see the answer in a glance, as an [instant answer](#). The data is generated automatically from a variety of sources, covering places, people, businesses, and more.^{[1][2]}

The information covered by Google's Knowledge Graph grew quickly after launch, tripling its data size within seven months (covering 570 million entities and 18 billion facts^[3]). By mid-2016, Google reported that it held 70 billion facts^[4] and answered "roughly one-third" of the 100 billion monthly searches they handled. By May 2020, this had grown to 500 billion facts on 5 billion entities.^[5]

There is no official documentation of how the Google Knowledge Graph is implemented.^[6] According to Google, its information is retrieved from many



Thomas Jefferson

3rd U.S. President

Thomas Jefferson was an American Founding Father, the principal author of the Declaration of Independence, and the third President of the United States. [Wikipedia](#)

Born: April 13, 1743, Shadwell, VA
Died: July 4, 1826, Charlottesville, VA
Presidential term: March 4, 1801 – March 4, 1809
Spouse: Martha Jefferson (m. 1772–1782)
Party: Democratic-Republican Party
Awards: AIA Gold Medal

Get updates about Thomas Jefferson

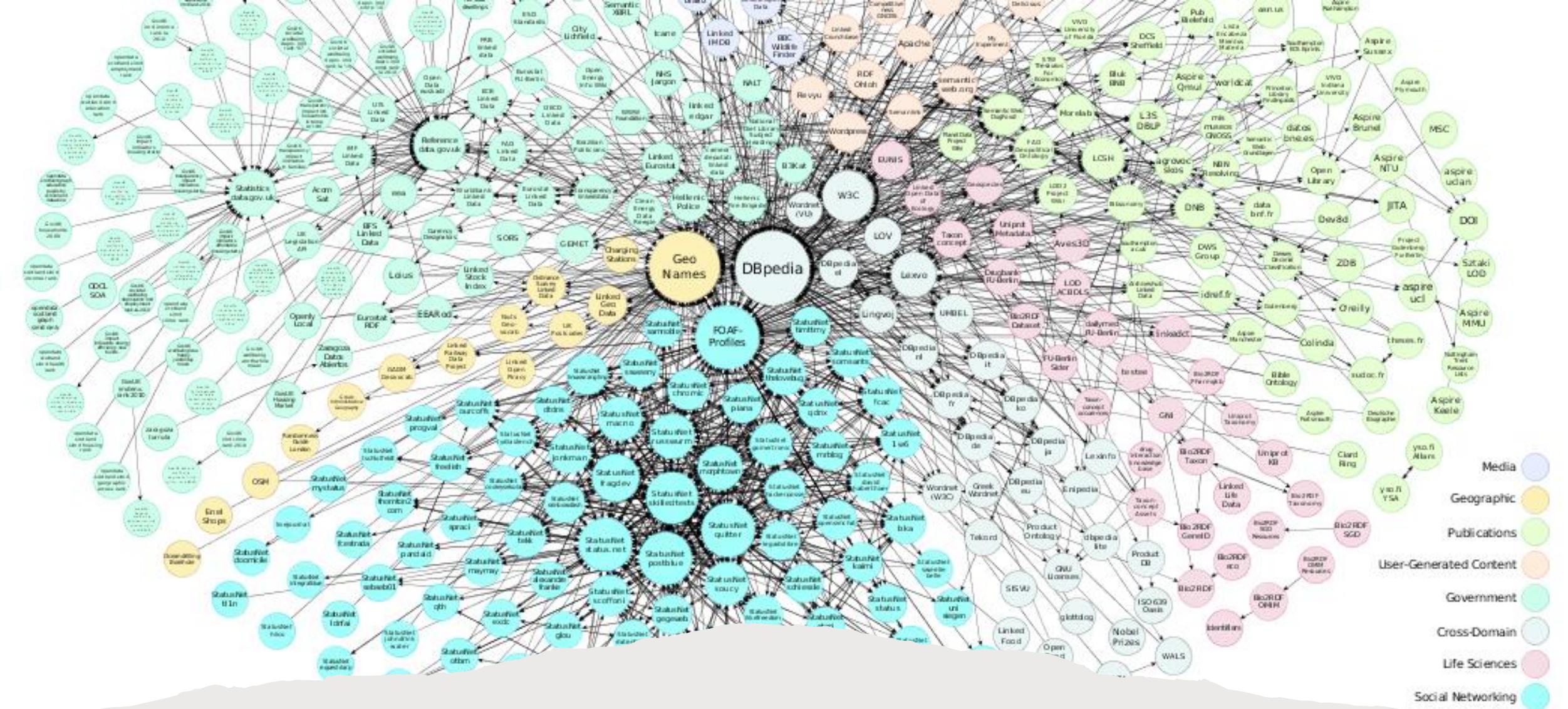
Keep me updated

People also search for

View 15+ more



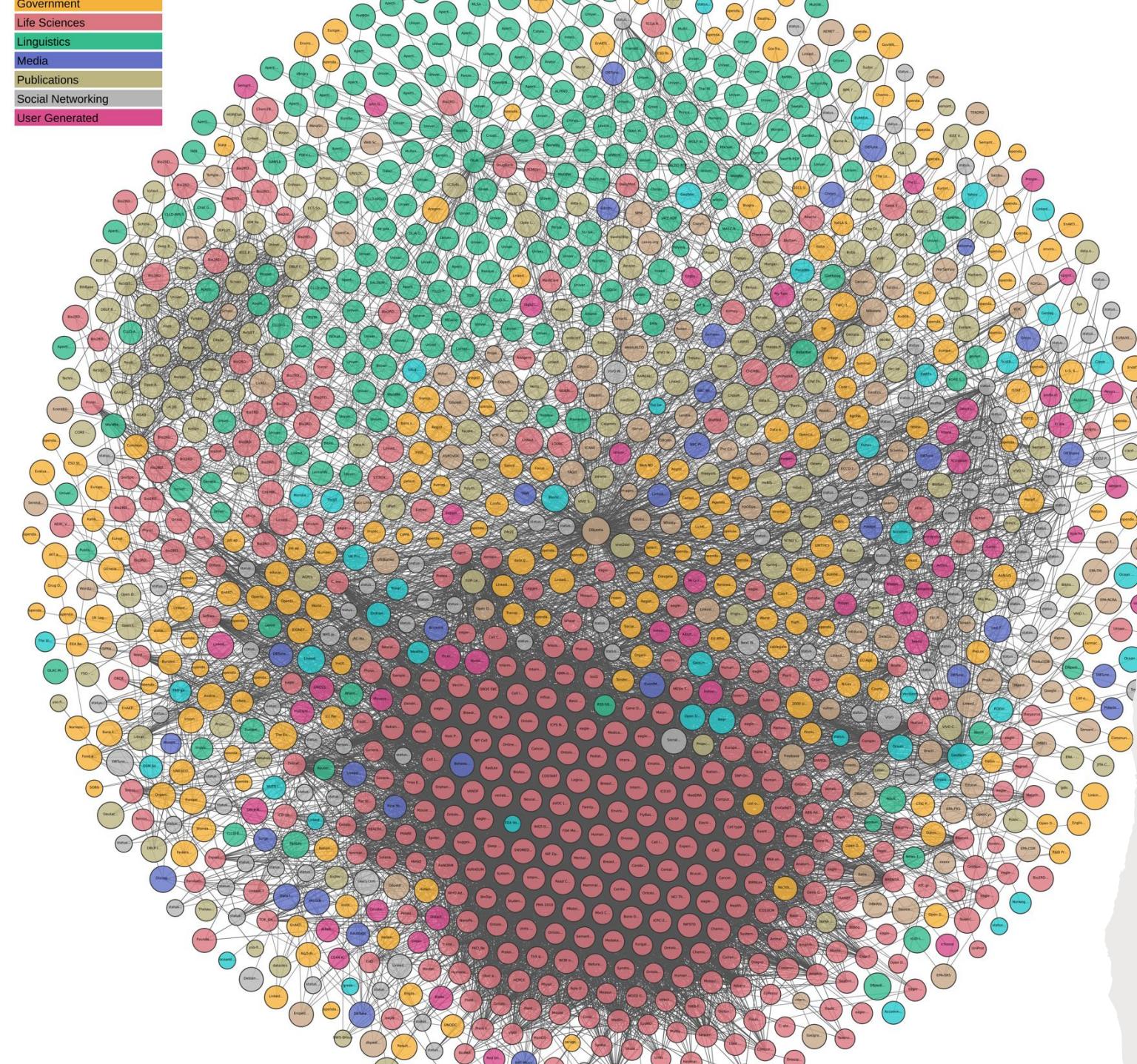
Feedback



(crawlable) 2014

<http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

Government
Life Sciences
Linguistics
Media
Publications
Social Networking
User Generated



2019

<https://lod-cloud.net/>

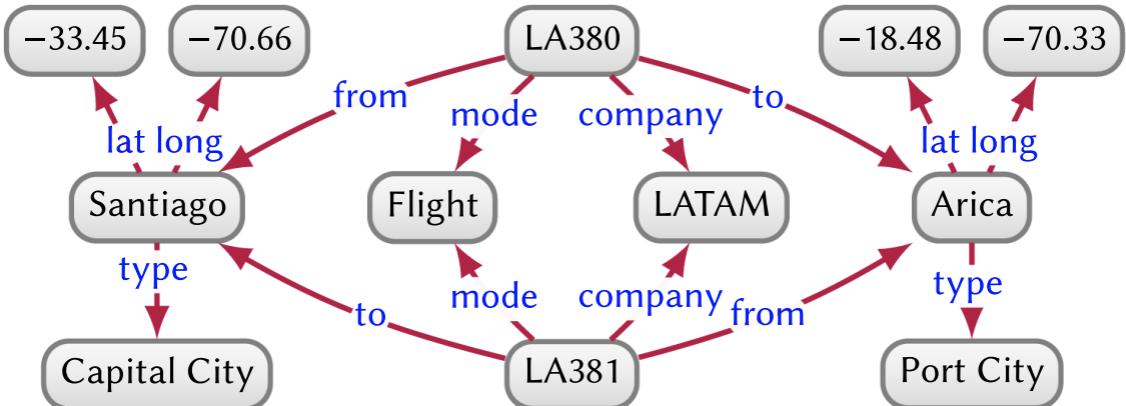


A parenthesis on terminology

- Semantic Web: TBL's vision of a global, distributed database
- Linked Data: a data publishing/consuming method
- Ontologies: (methods for) knowledge representation
- Knowledge Graphs: ?

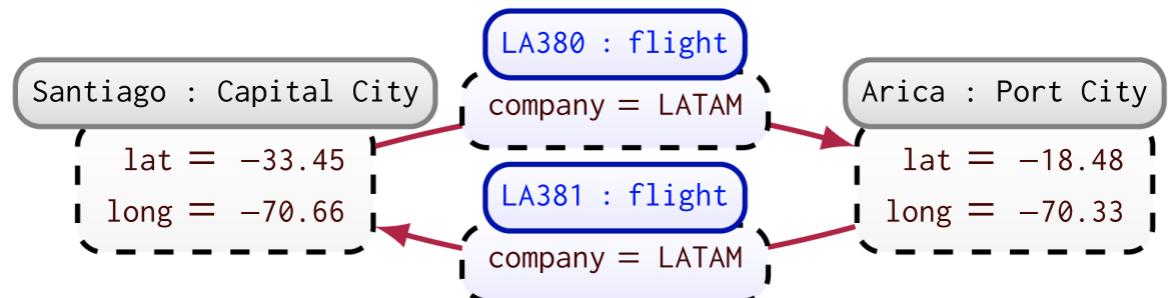


Directed Edge-labelled Graphs



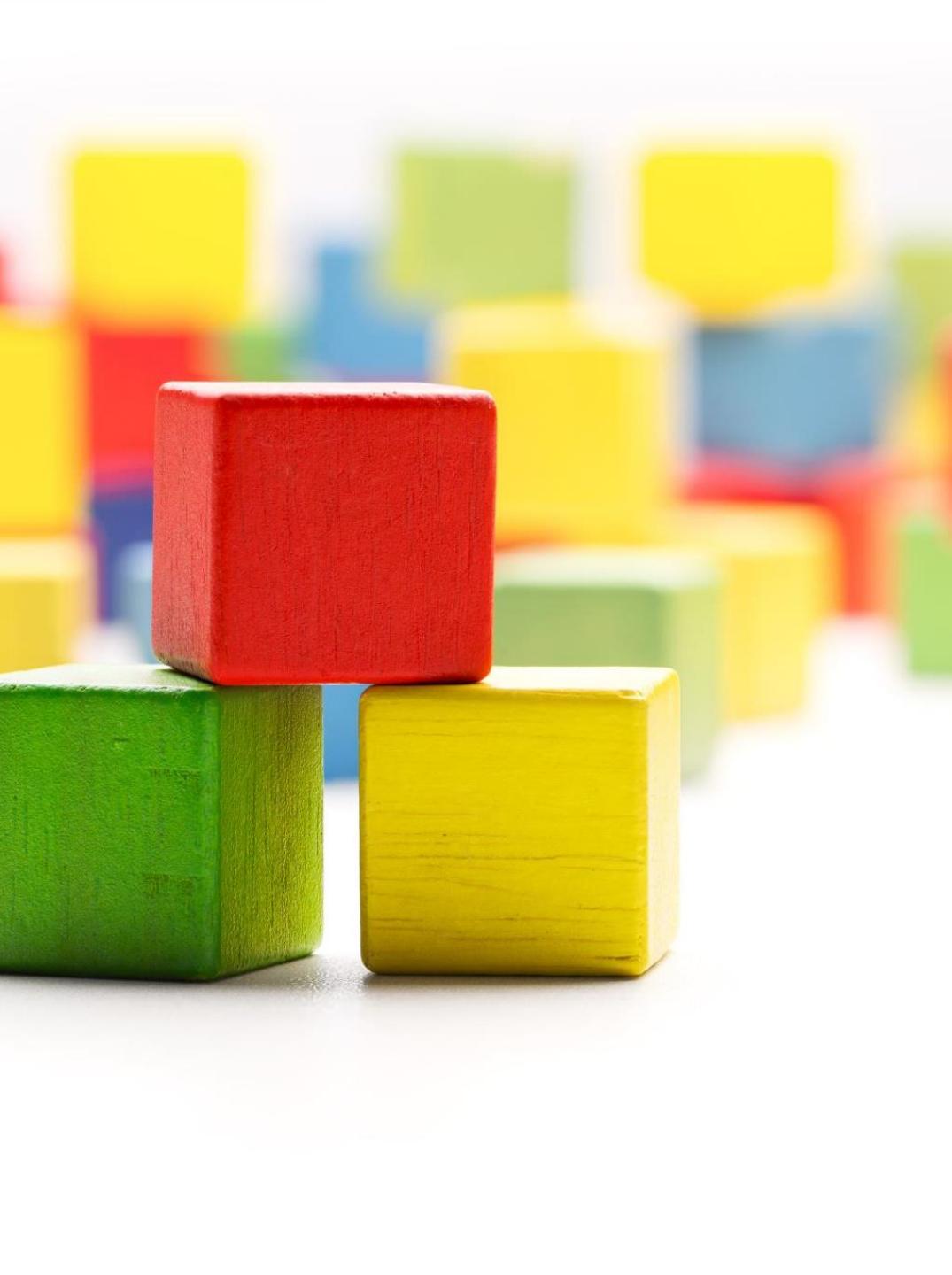
(a) Del graph

Property Graphs



(b) Property graph

Fig. 3. Flight data in a del graph and a property graph.

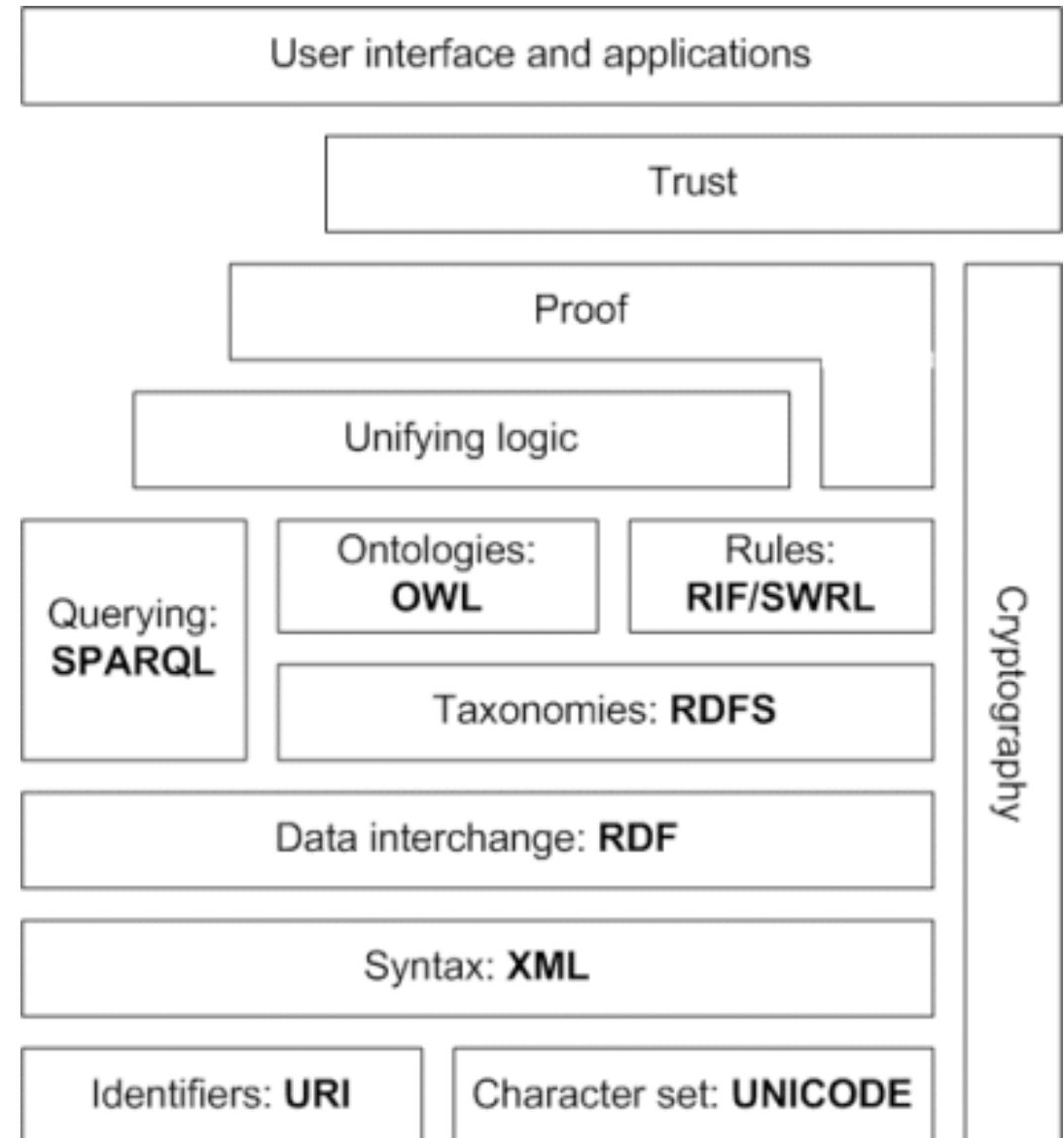


Resource Description Framework (RDF)

The very basics

The W3C “Layer Cake”

- A hierarchy of languages
- Each layer exploits and uses capabilities of the layers below



<https://www.w3.org/>

The semantic Web



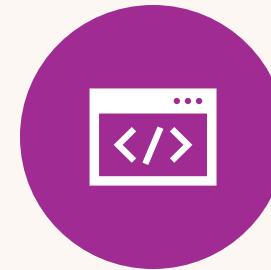
A PRINCIPLE:
HYPertext



A PROTOCOL: **HTTP**



AN IDENTIFICATION
SCHEME: **URNs/URIs**



A LANGUAGE: **RDF**

The traditional Web



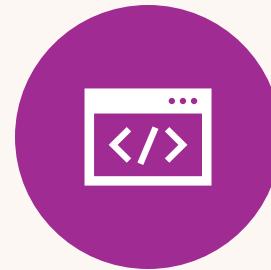
A PRINCIPLE:
HYPertext



A PROTOCOL: **HTTP**



AN IDENTIFICATION
SCHEME: **URNs/URIs**

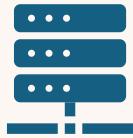


A LANGUAGE: **HTML**



Uniform Resource Identifiers (URIs) -- Internationalized Resource Identifiers (IRIs)

To identify things



HyperText Transfer Protocol (HTTP)

To access data about them



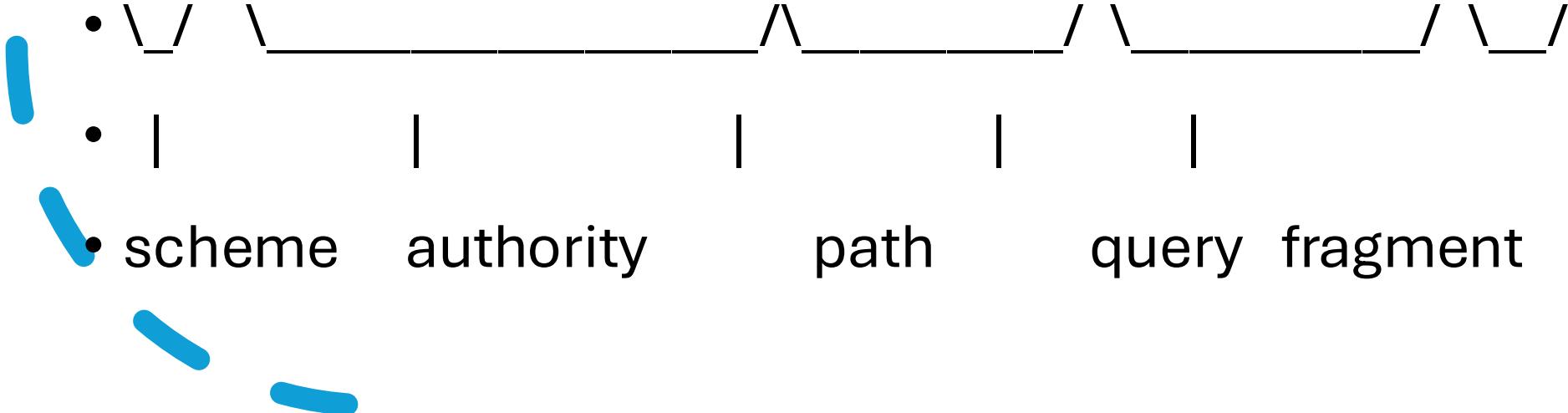
Resource Description Framework (RDF)

a *meta-model* for data representation.
it does not specify a particular
schema
offers a *structure* for representing
schemas and data



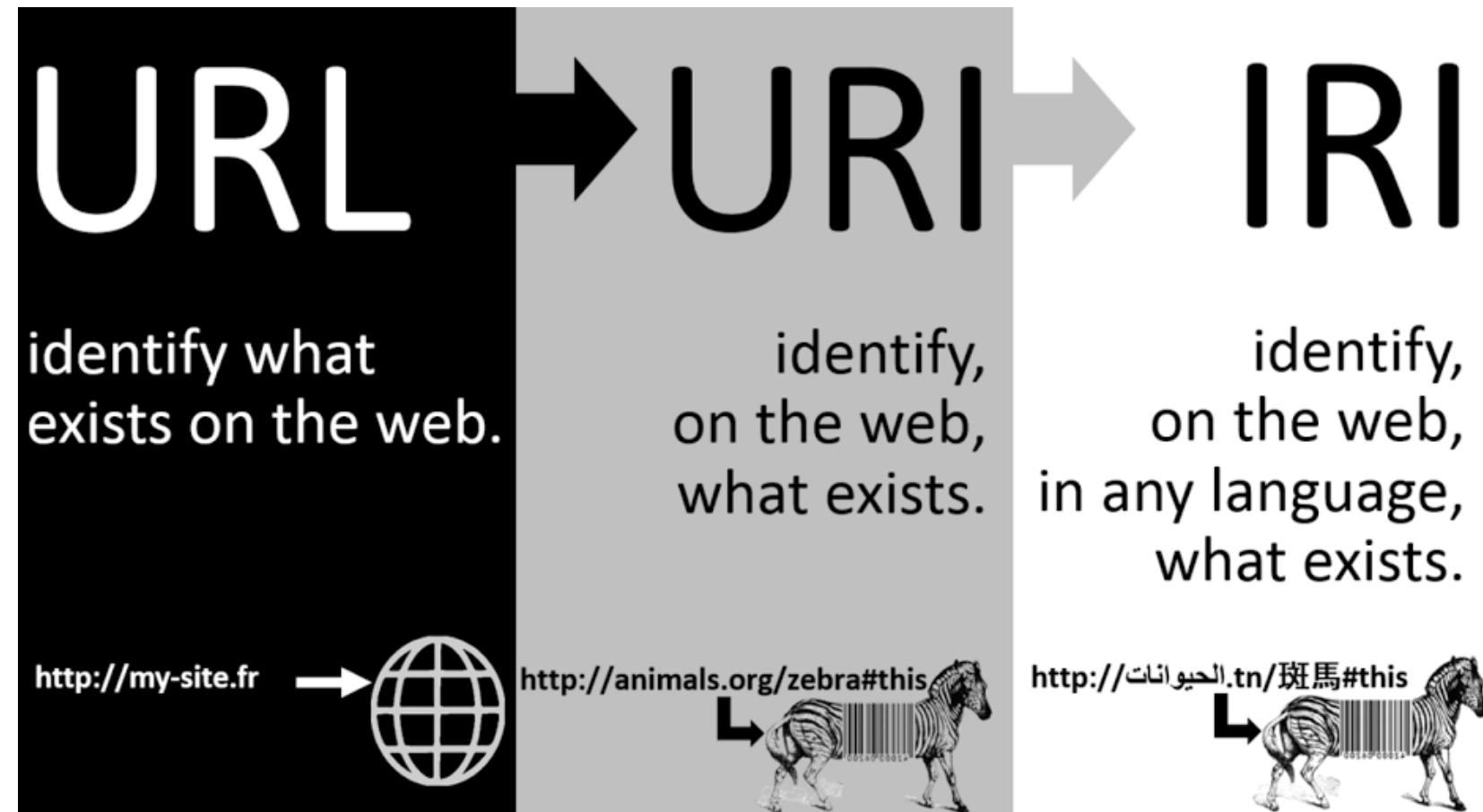
SPARQL Protocol and Query Language (SPARQL)

To query LD databases directly on the
Web

- A Uniform Resource Identifier (URI) is a compact **sequence of characters** that **identifies** an abstract or physical **resource**. [RFC3986]
- Syntax
- URI = scheme ":" hier-part ["?" query] ["#" fragment]
- Example
- `foo://example.com:8042/over/there?name=ferret#nose`
- 

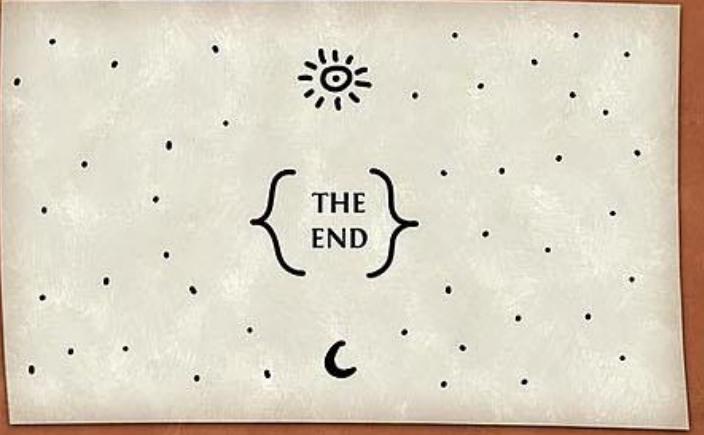
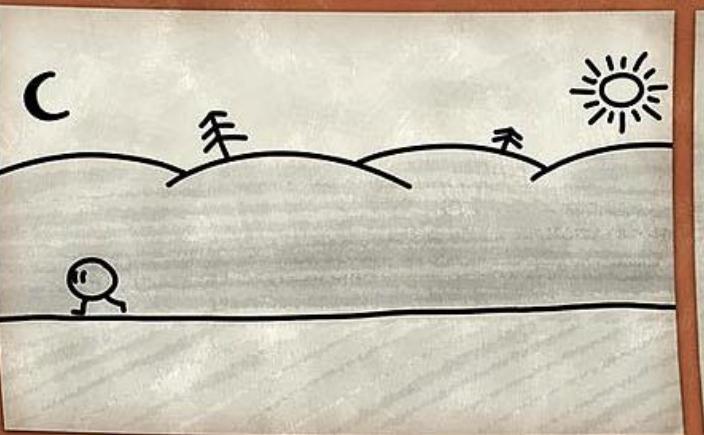
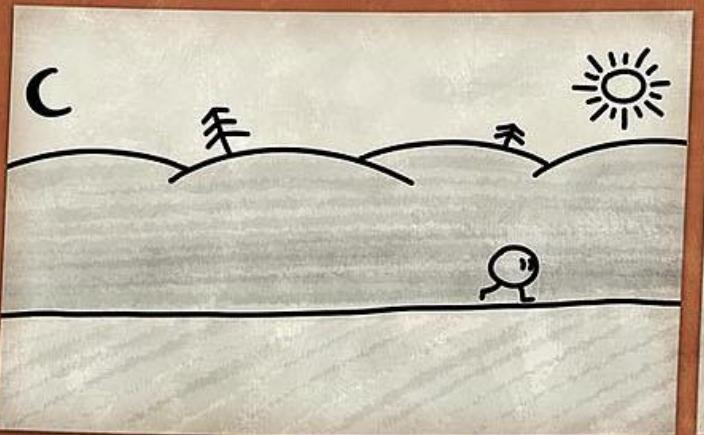
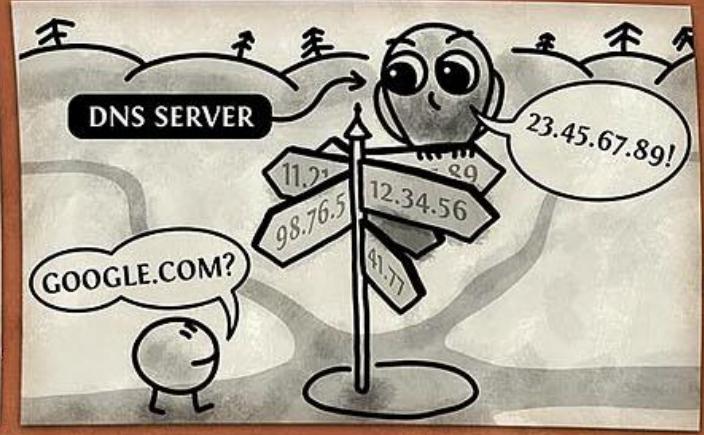
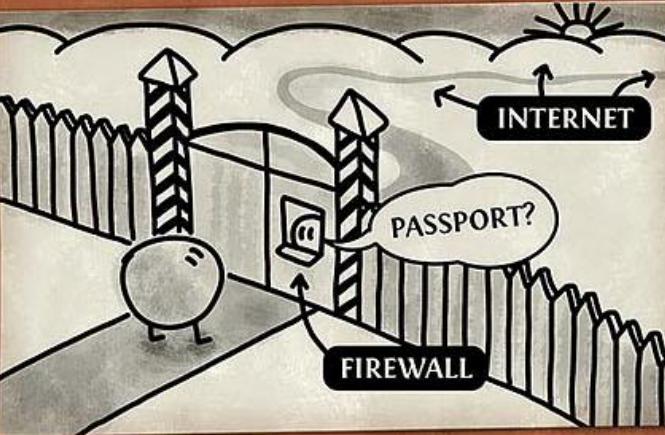
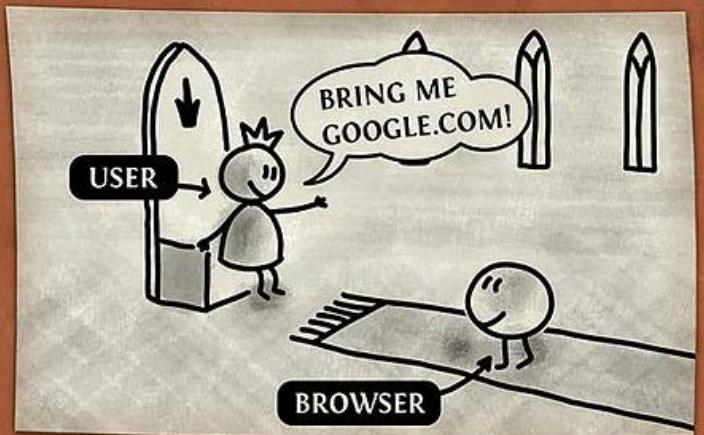
```
\_/\_ \_____\^_____/ \_____/ \_/\_
| | | | |
scheme authority path query fragment
```

- URIs (Unique Resource Identifiers) are used to identify **things** (also called **entities**) in the real world
- For instance: people, places, events, companies, products, movies, etc.



HTTP

- 
- On top of
 - The Internet Protocol (IPv4/v6)
 - Domain Name System (DNS): e.g. **dbpedia.org**
 - A Client / Server protocol: *Request -> Response*
 - Message structure: **Headers + Body** (content)



HTTP

Message structure

Headers

- Vary between Request and Response
(two newlines)

Body

- Any data

HTTP

http://dbpedia.org/resource/Wolfgang_Amadeus_Mozart

REQUEST

```
GET /resource/wolfgang_Amadeus_Mozart
HTTP/1.1
Host: dbpedia.org
User-Agent: curl/7.19.7
Accept: */*
```



RESPONSE

```
HTTP/1.1 303 See Other
Date: Wed, 03 Jul 2019 13:41:14 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Connection: keep-alive
Server: Virtuoso/07.20.3230
Location: http://dbpedia.org/page/wolfgang_Amadeus_Mozart
Access-Control-Allow-Origin: *
...
```

- cURL is a command line tool and library for transferring data with URLs



HTTP

```
curl -v http://dbpedia.org/resource/Wolfgang_Amadeus_Mozart
```

REQUEST
GET /resource/wolfgang_Amadeus_Mozart HTTP/1.1
Host: dbpedia.org
User-Agent: curl/7.19.7
Accept: */*

 HTTP/1.1 303 See Other
Date: wed, 03 Jul 2019 13:41:14 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Connection: keep-alive
Server: virtuoso/07.20.3230
Location: https://dbpedia.org/page/wolfgang_Amadeus_Mozart
Access-Control-Allow-Origin: *

RESPONSE

HTTP

```
curl -v https://dbpedia.org/resource/Wolfgang_Amadeus_Mozart
```

REQUEST
GET /resource/wolfgang_Amadeus_Mozart HTTP/1.1
Host: dbpedia.org
User-Agent: curl/7.19.7
Accept: */*

HTTP/1.1 303 See Other

Date: wed, 03 Jul 2019 13:41:14 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Connection: keep-alive
Server: virtuoso/07.20.3230
Location: https://dbpedia.org/page/wolfgang_Amadeus_Mozart
Access-Control-Allow-Origin: *

RESPONSE

```
curl -v http://dbpedia.org/page/Wolfgang_Amadeus_Mozart
```

```
GET /page/Wolfgang_Amadeus_Mozart HTTP/1.1
```

```
Host: dbpedia.org
```

```
User-Agent: curl/7.19.7
```

```
Accept: */*
```



```
HTTP/1.1 200 OK
```

```
Date: wed, 03 Jul 2019 13:41:14 GMT
```

```
Content-Type: text/html; charset=UTF-8
```

```
Content-Length: 0
```

```
Connection: keep-alive
```

```
Server: virtuoso/07.20.3230
```

```
[...]
```

```
<html> [...]
```

```
curl -v https://dbpedia.org/resource/Wolfgang\_Amadeus\_Mozart  
      -H "Accept: text/turtle"
```

GET /resource/Wolfgang_Amadeus_Mozart HTTP/1.1

Host: dbpedia.org

User-Agent: curl/7.19.7

Accept: text/turtle



HTTP/1.1 303 See Other

Date: Wed, 03 Jul 2019 13:41:14 GMT

Content-Type: text/html; charset=UTF-8

Content-Length: 0

Connection: keep-alive

Server: virtuoso/07.20.3230



Location: https://dbpedia.org/data/Wolfgang_Amadeus_Mozart.ttl

```
curl -v http://dbpedia.org/data/Wolfgang_Amadeus_Mozart.ttl
```

```
GET /data/wolfgang_Amadeus_Mozart.ttl HTTP/1.1
```

```
Host: dbpedia.org
```

```
User-Agent: curl/7.19.7
```

```
Accept: text/turtle
```

```
HTTP/1.1 200 OK
```

```
Content-Type: text/turtle; charset=UTF-8
```

```
Content-Length: 50708
```

```
[...]
```

```
→ @prefix dbo: <http://dbpedia.org/ontology/> .
```

```
@prefix dbr: <http://dbpedia.org/resource/> .
```

```
dbr:Amadeus_Mozart dbo:wikiPageRedirects dbr:wolfgang_Amadeus_Mozart .
```

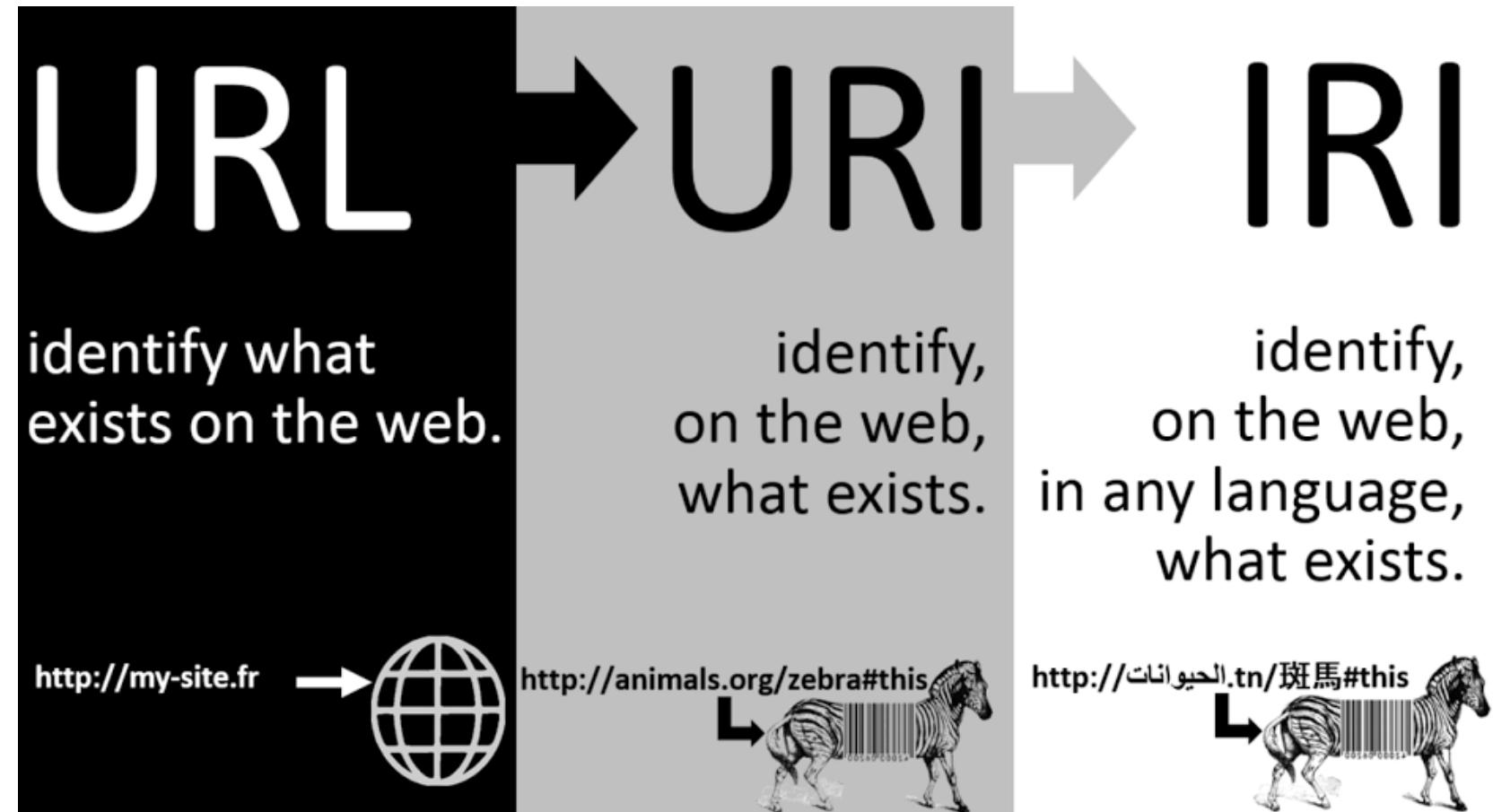
```
dbr:The_Story_of_Mozart dbo:wikiPageRedirects dbr:wolfgang_Amadeus_Mozart
```

```
dbr:Mozartian dbo:wikiPageRedirects dbr:wolfgang_Amadeus_Mozart .
```

```
<http://dbpedia.org/resource/Don_Giovanni_(1979_film)> dbo:musicComposer db
```

```
[...]
```

- URI / IRI refers to entities
- URL refers to locations



-
- curl -v http://dbpedia.org/resource/wolfgang_Amadeus_Mozart
 - curl -v "http://dbpedia.org/page/wolfgang_Amadeus_Mozart"
 - curl -v http://dbpedia.org/resource/wolfgang_Amadeus_Mozart
 - -H "Accept: text/turtle"
 - curl -v "http://dbpedia.org/data/wolfgang_Amadeus_Mozart.ttl"
 - curl -v http://dbpedia.org/resource/wolfgang_Amadeus_Mozart
 - -H "Accept: text/turtle" -L

How many Mozart?

http://dbpedia.org/resource/Wolfgang_Amadeus_Mozart

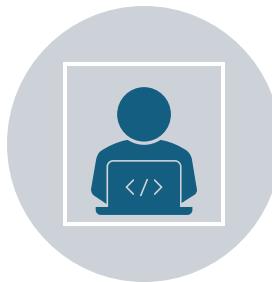
<http://www.wikidata.org/entity/Q254>

- In what formats is Mozart available?
 - text/html, application/rdf+xml, text/n-triples, text/turtle, application/ld+json
- Find Mozart's image (it's a jpg)
- When Mozart was born?
- Where Mozart died?
- How many inhabitants has the city where Mozart died, today?

Linked Data principles



Use **URIs** to identify the “things” in your data



Use **http:// URIs** so people (and machines) can look them up on the web



When a URI is **looked up, request/return a description** of the thing in RDF



Include **links to related things** (e.g. owl:sameAs)

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

The RDF model

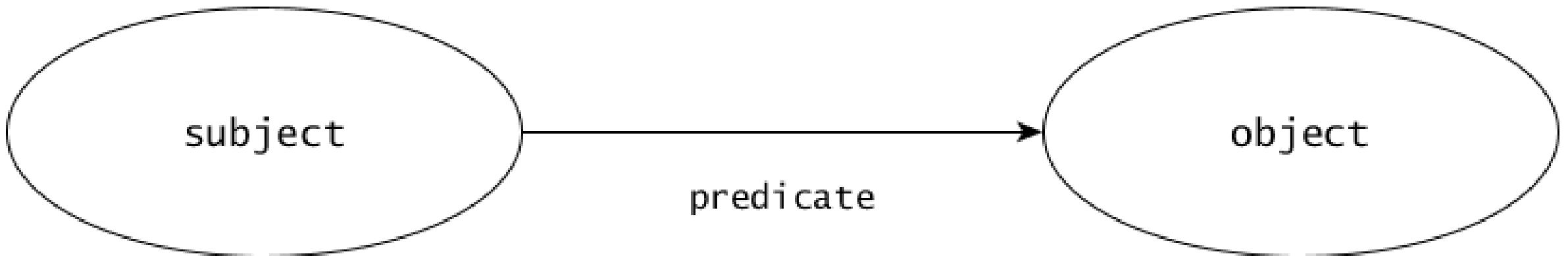
(the “content” of the HTTP body...)

Resource Description Framework

Relationships between things are expressed by the means of a *multi-directed, fully labelled graph*

where

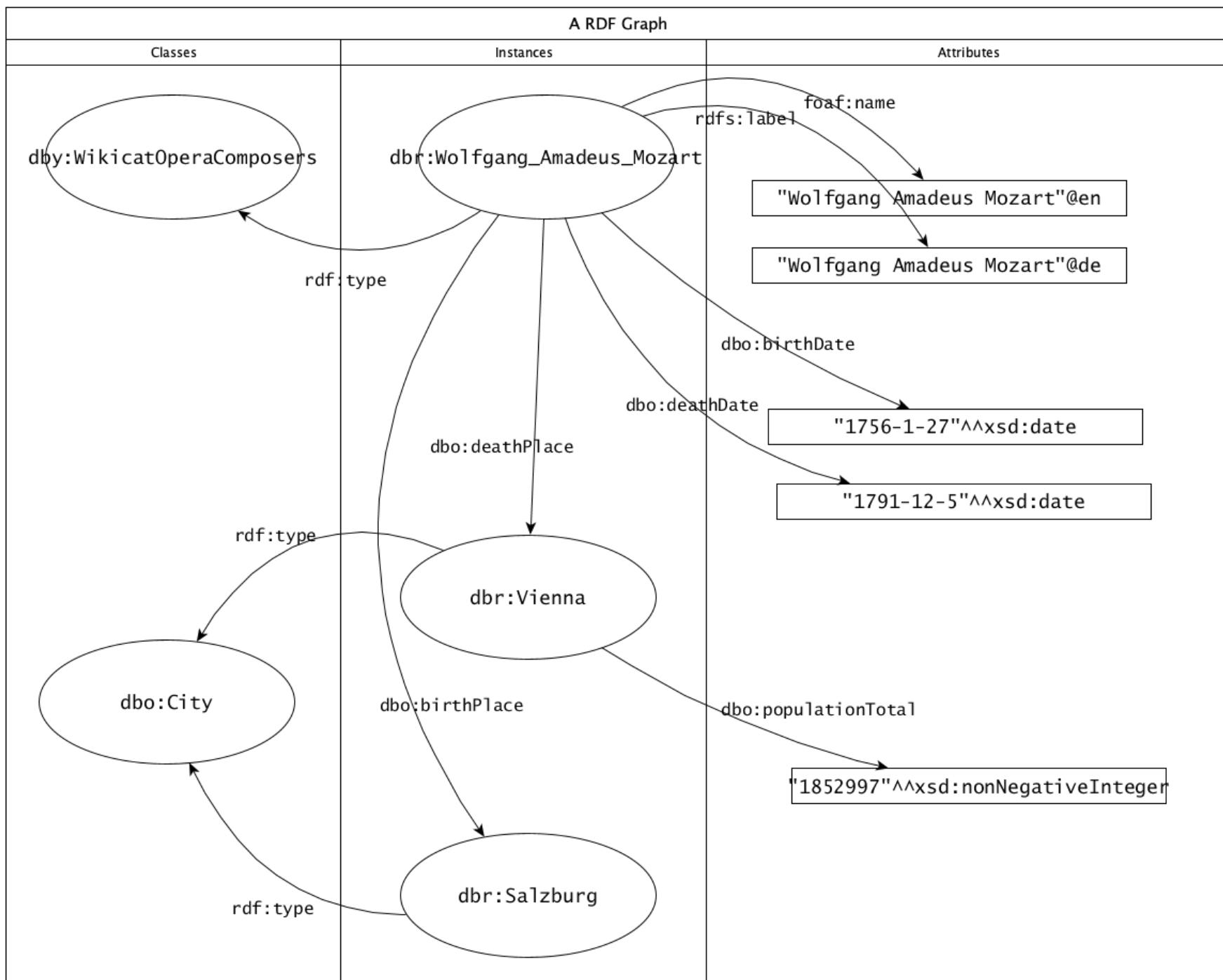
nodes could be resources identified by IRIs or XMLSchema-typed values; **relationships** are also identified by IRIs



RDF is based on an atomic element: the triple.

Triple: (subject predicate object)

- subject: a URI or a blank node
- predicate: MUST be a URI
- object: a URI, a blank node, or a literal



Literals

- **Representation of data values**
- **Serialization as strings**
- **Interpretation based on the *datatype***
- **Literals without Datatype** are treated as strings
- and can be annotated with a language (Alpha-2): @en

RDF serializations

N-Triples (application/n-triples)

Turtle (text/turtle)

RDF/XML (application/rdf+xml)

N-Quads (application/n-quads)

TriG (application/trig)

JSON-LD (application/ld+json)

[...]

N-Triples

<https://www.w3.org/TR/n-triples/>

EXAMPLE 1

```
<http://one.example/subject1> <http://one.example/predicate1> <http://one.example/object1> . # comments here  
# or on a line by themselves  
_:subject1 <http://an.example/predicate1> "object1" .  
_:subject2 <http://an.example/predicate2> "object2" .
```

EXAMPLE 4

```
_:alice <http://xmlns.com/foaf/0.1/knows> _:bob .  
_:bob <http://xmlns.com/foaf/0.1/knows> _:alice .
```

Namespaces

- Namespaces in XML: <https://www.w3.org/TR/xml-names/>
- Namespaces end either with # or /
- In some serialisations, are mapped to **prefixes**, for brevity
- <http://prefix.cc> to get help with namespaces and common prefixes

- http://dbpedia.org/resource/Wolfgang_Amadeus_Mozart
- <http://dbpedia.org/resource/> (namespace)
- dbr:Wolfgang_Amadeus_Mozart (using a “prefix”)

Turtle

<https://www.w3.org/TR/turtle/>

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix dbr: <http://dbpedia.org/resource/> .  
@prefix dbo: <http://dbpedia.org/ontology/> .  
@prefix yago: <http://dbpedia.org/class/yago/> .  
@prefix wikidata: <http://www.wikidata.org/entity/> .
```

```
dbr:wolfgang_Amadeus_Mozart  
    rdfs:label "Wolfgang Amadeus Mozart" ;  
    rdf:type owl:Thing , yago:wikiCatGermanClassicalComposers ,  
          yago:wikiCatGermanComposers , dbo:Person .
```

```
dbr:wolfgang_Amadeus_Mozart owl:sameAs wikidata:Q254 ;  
    dbo:deathPlace dbr:vienna .
```

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
dbr:wolfgang_Amadeus_Mozart dbo:deathDate "1791-12-5"^^xsd:date ;  
    dbo:birthPlace dbr:Salzburg ;  
    dbo:birthDate "1756-1-27"^^xsd:date .
```

Named Graphs

- RDF1.1 **named graphs** allow to integrate multiple RDF documents preserving the context of each triple: **g s p o**
- Syntax: N-Quads
- Triple Stores: database management systems that allow to query RDF

-
- **SPARQL**: <https://www.w3.org/TR/sparql11-overview/>
 - OWL: <https://www.w3.org/OWL/>
 - SHACL: <https://www.w3.org/TR/shacl/>
 - RDF-Star: <https://w3c.github.io/rdf-star/cg-spec/2021-12-17.html>
 - SPIN: <https://spinrdf.org/>
 - RDFa: <https://www.w3.org/TR/rdfa-syntax/>
 - RIF: <https://www.w3.org/TR/rif-overview/>
 - JSON-LD: <https://json-ld.org/>

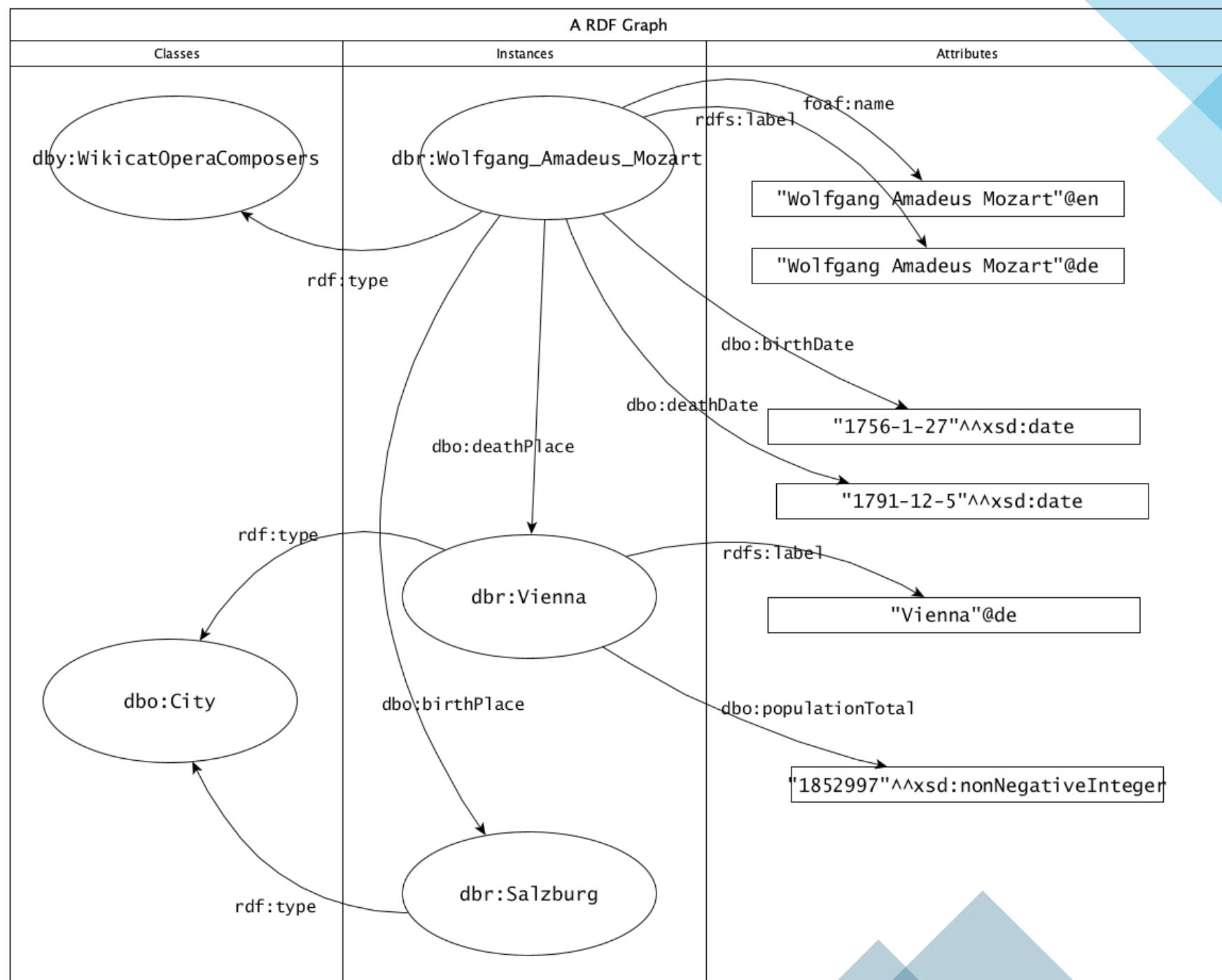
“Find all grand operas in DBpedia”

- *This type of task is possible using a SPARQL endpoint:*
- <http://dbpedia.org/sparql>

```
SELECT * WHERE {  
    ?entity  
    <http://purl.org/dc/terms/subject>  
    <http://dbpedia.org/resource/Category:Grand\_operas> .  
}
```

Triple and Graph Patterns

How do we describe the structure of the RDF graph which we're interested in?



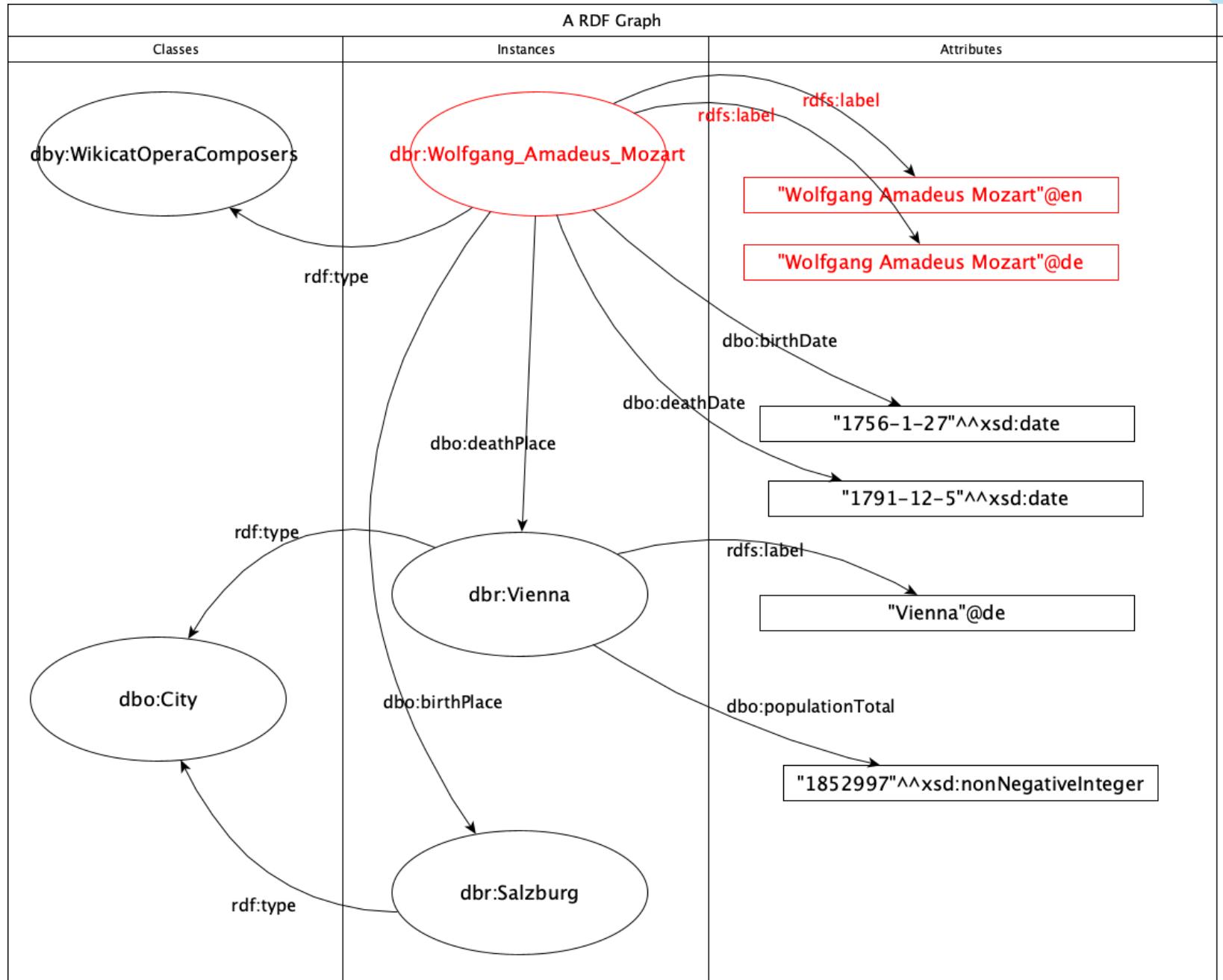
An RDF triple in Turtle syntax

```
@PREFIX dbr: <http://dbpedia.org/resource/> .  
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
  
dbr:Wolfgang_Amadeus_Mozart rdfs:label "Wolfgang  
Amadeus Mozart"@en .
```

A SPARQL triple pattern, with a single variable

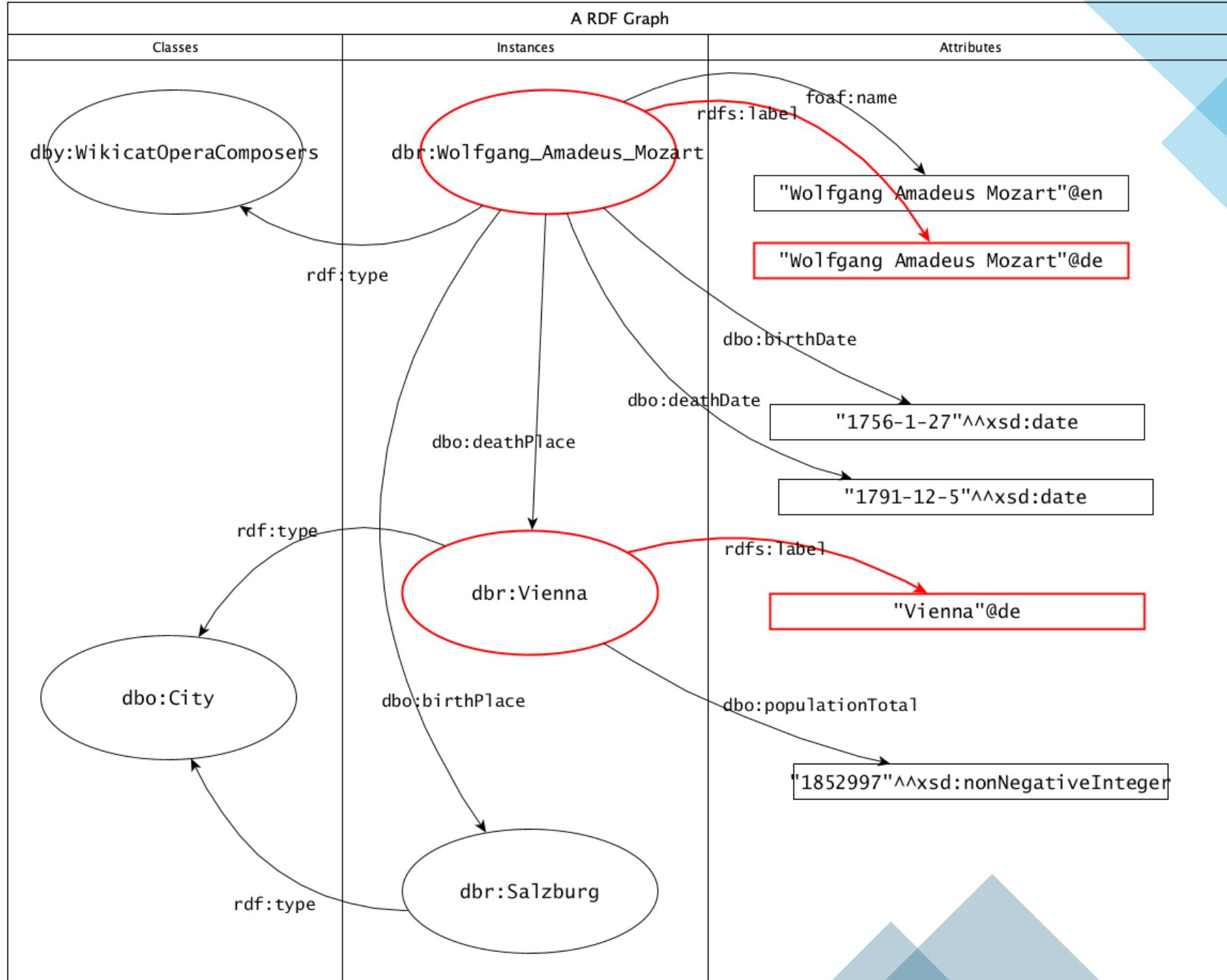
PREFIX dbr: <<http://dbpedia.org/resource/>>
PREFIX rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>

dbr:Wolfgang_Amadeus_Mozart rdfs:label ?name .



- # All parts of a triple pattern can be variables
- ?subject ?predicate ?object .

- # Matching labels of resources
- PREFIX rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>
- ?subject rdfs:label ?label.



```
# Combine triples patterns to create a graph pattern
```

```
PREFIX dby: <http://dbpedia.org/class/yago/>
```

```
?subject rdfs:label ?label .
```

```
?subject rdf:type dby:wikicatOperaComposers .
```

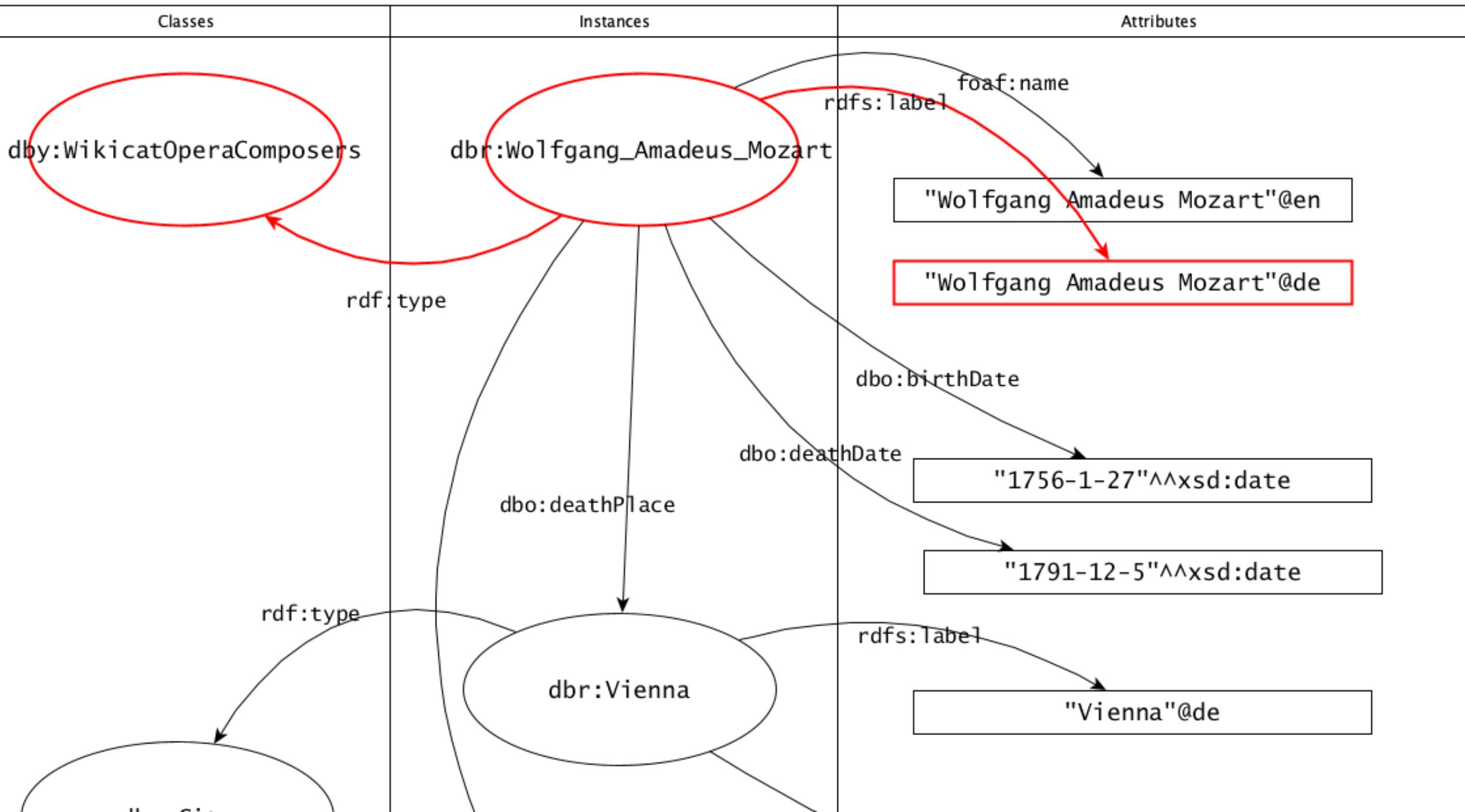
```
# SPARQL is based on Turtle, which allows abbreviations
```

```
# e.g. predicate-object lists:
```

```
?subject rdfs:label ?label;
```

```
    rdf:type dby:wikicatOperaComposers .
```

A RDF Graph



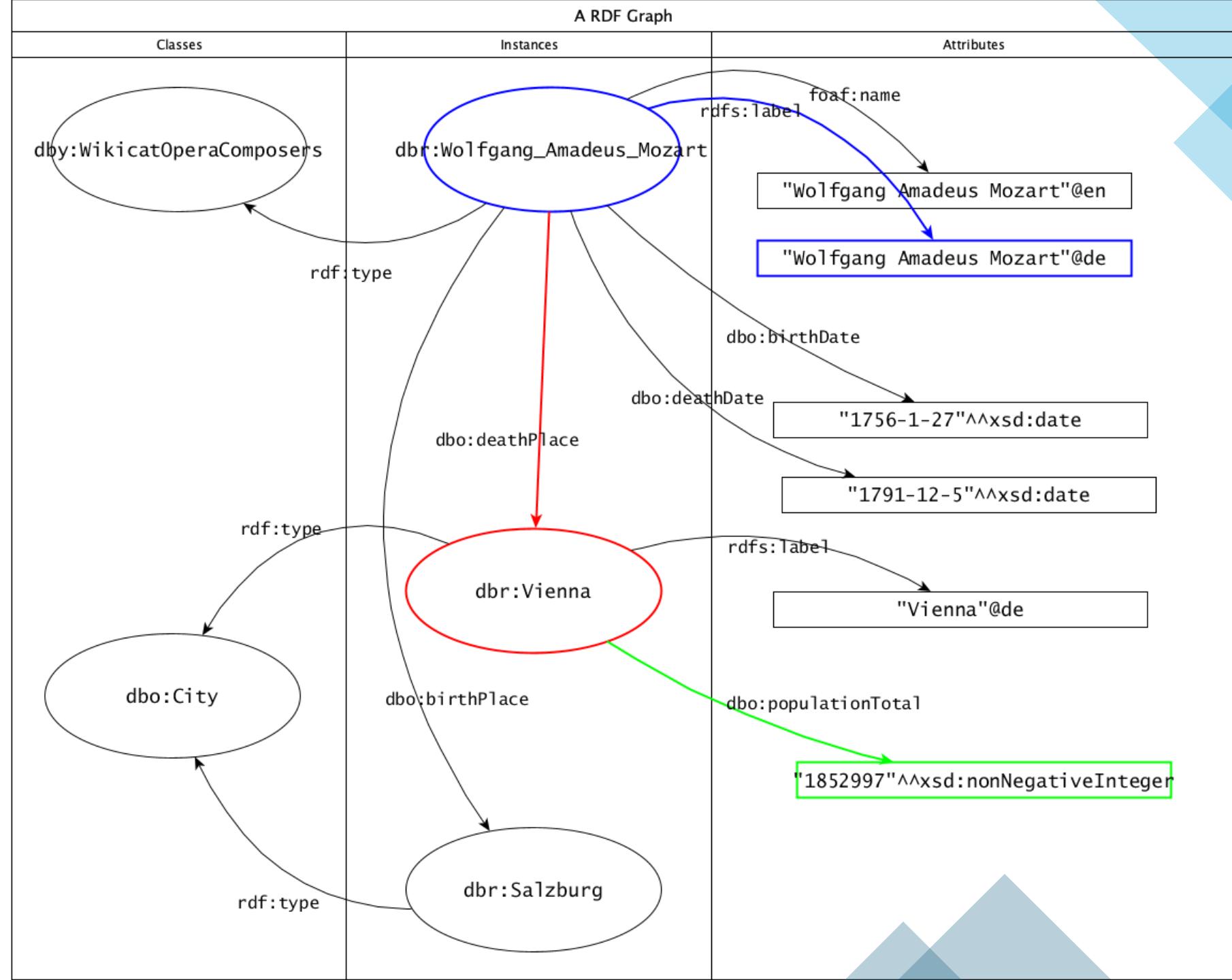
```
# Graph patterns allow us to traverse a graph  
  
?person rdfs:label "Wolfgang Amadeus Mozart"@de .  
  
?person dbo:deathPlace ?place .  
  
?place dbo:populationTotal ?population .
```

#graph patterns allow us to query the graph

?person rdfs:label "Wolfgang Amadeus Mozart"@de .

?person dbo:deathPlace ?place .

?place dbo:populationTotal ?population .



Operators

projection (π ,
a.k.a. SELECT)

selection (σ ,
a.k.a. WHERE or
FILTER)

union (U , a.k.a.
UNION)

difference ($-$,
a.k.a. MINUS)

inner joins (,
a.k.a. NATURAL
JOIN)

left outer join (,
a.k.a. LEFT
OUTER JOIN or
OPTIONAL)

anti-join (, a.k.a.
NOT EXISTS)

Structure of a Query

What does a basic SPARQL query look like?

```
# Query. 1  
# Associate URIs with prefixes
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>  
PREFIX dbo: <http://dbpedia.org/ontology/>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
# Example of a SELECT query, retrieving 2 variables  
# Variables selected MUST be bound in graph pattern
```

```
SELECT ?place ?population  
WHERE {  
  
  #This is our graph pattern  
  ?person rdfs:label "Cab Calloway"@en ;  
           dbo:deathPlace ?place .  
  ?place dbo:populationTotal ?population  
}
```

- We will use this UI: <http://yasgui.org/>
- Credits: <http://laurensrietveld.nl/>



Yet Another SPARQL GUI

Funded by  Data²Semantics
From Data to Semantics for Scientific Data Publishers

```
# Query. 2
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
# Example of a SELECT query, retrieving all variables
```

```
SELECT *
```

```
WHERE {
```

```
?person rdfs:label "Cab Calloway"@en ;
        dbo:deathPlace ?place .
?place dbo:populationTotal ?population .
```

```
}
```

OPTIONAL bindings

How do we allow for missing or unknown information?

Query. 3

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?person ?population

WHERE {

#This pattern must be bound

?person rdfs:label "Cab Calloway"@en ;
 dbo:birthPlace ?place .

#Anything in this block doesn't have to be bound (dbo:whatever does not exist)

OPTIONAL {

?place dbo:whatever ?population .

}

}

UNION queries

How do we allow for alternatives or variations in the graph?

Query. 4

```
PREFIX dbo:  
<http://dbpedia.org/ontology/>  
  
SELECT (COUNT(*) as ?c)  
WHERE {  
    {  
        ?person dbo:deathPlace  
        ?place .  
    }  
    UNION  
    {  
        ?person dbo:birthPlace  
        ?place .  
    }  
}  
  
# Check the difference  
  
PREFIX dbo:  
<http://dbpedia.org/ontology/>  
  
SELECT (COUNT(*) as ?c)  
WHERE {  
    {  
        ?person dbo:deathPlace  
        ?place .  
    }  
    # }  
    # UNION  
    # {  
        ?person dbo:birthPlace  
        ?place .  
    }  
}
```

Sorting & Restrictions



- How do we apply a sort order to the results?
- How can we add restrictions?
- How can we restrict the number of results returned?

```
# Query. 5  
# Select the URI and population of all places
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
SELECT ?place ?population  
WHERE {
```

```
    ?place dbo:populationTotal ?population .
```

```
}
```

```
# Ex. 6
# Select the URI and population of all places
# with highest first

PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?place ?population
WHERE {
    ?place dbo:populationTotal ?population .
}

# Use an ORDER BY clause to apply a sort.
# Can be ASC or DESC
ORDER BY DESC(?population)
```

```
# Ex. 7
# Select the URI and population of a city
# with highest first

PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT ?place ?population
WHERE {
    ?place dbo:populationTotal ?population .
    FILTER EXISTS {
        ?place dbp:countryCode []
    }
}
# Use an ORDER BY clause to apply a sort.
# Can be ASC or DESC
ORDER BY DESC(?population)
```

```
# Ex. 8
# Select the URI and population of the 11-20th most populated countries

PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT ?place ?population
WHERE {
    ?place dbo:populationTotal ?population .
    FILTER EXISTS {
        ?place dbo:countryCode []
    }
}
ORDER BY DESC(?population) # Use an ORDER BY clause to apply a sort.
LIMIT 10 # Limit to first ten results
OFFSET 10 # Apply an offset to get next “page”
```

Filtering

How do we restrict results based on aspects of the data rather than the graph, e.g. string matching?



In the following triple the literal has assigned a
datatype to indicate it is a date

PREFIX dbr: <<http://dbpedia.org/resource/>>
PREFIX dbo: <<http://dbpedia.org/ontology/>>
PREFIX xsd: <<http://www.w3.org/2001/XMLSchema#>>

dbr:wolfgang_Amadeus_Mozart
dbo:birthDate "1756-1-27"^^xsd:date

```
# Query. 9
# Select name of persons born between 1st Jan 1756 and 1st Jan 1757

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?name
WHERE {
    ?person dbo:birthDate ?date;
             rdfs:label ?name.

    FILTER (?date > "1756-01-01"^^xsd:date &&
            ?date < "1757-01-01"^^xsd:date)
}
```

```
# Query. 10
# Select the URI and population of places with an area below 20km^2, with
most populated first
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX dbpp: <http://dbpedia.org/ontology/PopulatedPlace>
```

```
SELECT ?place ?population
WHERE {
    ?place dbo:populationTotal ?population ;
           dbpp:areaTotal ?area .
```

```
# Note that we have to cast the data to the right type
# As it is not declared in the data
FILTER( xsd:double(?area) < 20 )
} ORDER BY DESC(?population)
```

```
# Query. 11
```

```
# Select persons named wolfgang
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
SELECT ?subject ?name
```

```
WHERE {
```

```
    ?subject rdfs:label ?name ;  
        dbo:deathPlace dbr:vienna .
```

```
    FILTER( regex(?name, "wolfgang", "i" ) )
```

```
}
```

Built-In Filters

- Logical: !, &&, ||
- Math: +, -, *, /
- Comparison: =, !=, >, <, ...
- Variable tests: isURI, isBlank, isLiteral, bound
- Accessors: str, lang, datatype
- Other: sameTerm, langMatches, regex

DISTINCT

How do we remove duplicate results?

```
# Query. 12
```

```
# Select list of places that gave birth to german classical composers
```

```
PREFIX space: <http://purl.org/net/schemas/space/>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

```
SELECT DISTINCT ?place
```

```
WHERE {
```

```
[] dbo:birthPlace ?place ;
```

```
    dct:subject dbc:German_classical_composers
```

```
}
```

SPARQL Query forms

- SELECT
- ASK
- DESCRIBE
- CONSTRUCT

ASK

Test whether the graph contains some data of interest

```
# Query . 13
```

```
# Is Mozart's date of birth 1756-1-27?
```

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
PREFIX xsd: http://www.w3.org/2001/XMLSchema#
```

```
PREFIX space: <http://purl.org/net/schemas/space/>
```

```
ASK WHERE {
```

```
    dbr:Mozart space:launched "1756-1-27"^^xsd:date .
```

```
}
```

```
# ASK returns a boolean value
```

DESCRIBE

Generate an RDF description of a resource(s)

```
# Query. 14

# Describe persons born in 1757

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dbp: <http://dbpedia.org/property/>

DESCRIBE ?person {
    ?person dbp:birthDate ?date .
    FILTER ( ?date < "1958-01-01"^^xsd:date &&
             ?date >= "1757-01-01"^^xsd:date )
}
```

CONSTRUCT

The diagram illustrates the process of using the CONSTRUCT query to transform RDF data. It features two large circles connected by an orange arrow pointing from left to right. The left circle is orange and contains the text "CREATE A CUSTOM RDF GRAPH BASED ON QUERY CRITERIA". The right circle is green and contains the text "CAN BE USED TO TRANSFORM RDF DATA". Above the circles, the word "CONSTRUCT" is written in large, bold, black capital letters.

CREATE A
CUSTOM RDF
GRAPH BASED
ON QUERY
CRITERIA



CAN BE USED
TO
TRANSFORM
RDF DATA

Example

```
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

CONSTRUCT {
    ?person foaf:name ?name ; a foaf:Person
} WHERE {
    ?person rdfs:label ?label ; a dbo:Person ; dbo:deathPlace dbr:Vienna
}
```

```
# Example.
```

```
# Remodelling! Change the identifier
```

```
PREFIX dbr: <http://dbpedia.org/resource/>
```

```
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX kgc: <urn:knowledge-graphs-course#>
```

```
CONSTRUCT {
```

```
    ?person foaf:name ?name ; a foaf:Person
```

```
} WHERE {
```

```
    ?p rdfs:label ?label ; a dbo:Person ; dbo:deathPlace dbr:Vienna .
```

```
    BIND( IRI( CONCAT( STR(kgc:), MD5(str(?p)) ) ) AS ?person) .
```

```
}
```

NAMED GRAPHS

SPARQL can query multiple RDF graphs together!

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#
```

```
SELECT distinct ?g WHERE {
  graph ?g {
    ?sub ?pred ?obj
  }
}
```

[data.CNR.it](#)[SPARQL Endpoint](#)[Browse](#)[CNR.it](#)

<http://www.cnr.it/ontology/cnr/individuo/CNR>

Italian National Research Council

Label

- Italian National Research Council (literal)
- Consiglio Nazionale delle Ricerche (literal)

Has part

- [Rete scientifica](#) (Rete scientifica CNR)

Logo

- //www.cnr.it/documenti/logo/logo.zip (literal)

Homepage

- //www.cnr.it (literal)

Incoming links:

Example resource of dataset

- [data.CNR.it dataset](#) ↗

Publisher

- [data.CNR.it :: Research results and publications](#) ↗
- [data.CNR.it :: People](#) ↗
- [data.CNR.it :: Attività di ricerca](#) ↗
- [data.CNR.it :: Units](#) ↗

Data.cnr.it

- Quanti articoli ci sono?
- Quali sono le proprietà con cui sono descritti i brevetti?
- Conta gli istituti, i dipartimenti, gli articoli
- Mostra una tabella con tutti i tipi e il numero di entita' per ciascun tipo