Semantics, knowledge graphs and ontologies in practice

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain

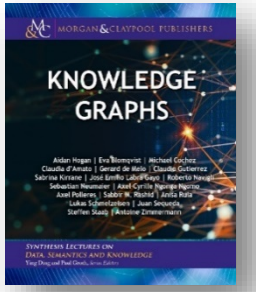
About me…

- Founded WESO (Web Semantics Oviedo) research group
  - Practical applications of semantic technologies since 2004
  - Several domains: e-Government, e-Health
- Some books:
  - "*Web semántica*" (in Spanish), 2012
  - "*Validating RDF data*", 2017
  - "Knowledge Graphs", 2021
- …and software:
  - SHaclEX (Scala library, implements ShEx & SHACL)
  - RDFShape (RDF playground)

http://labra.weso.es

2012

2017 HTML version:
http://book.validatingrdf.com

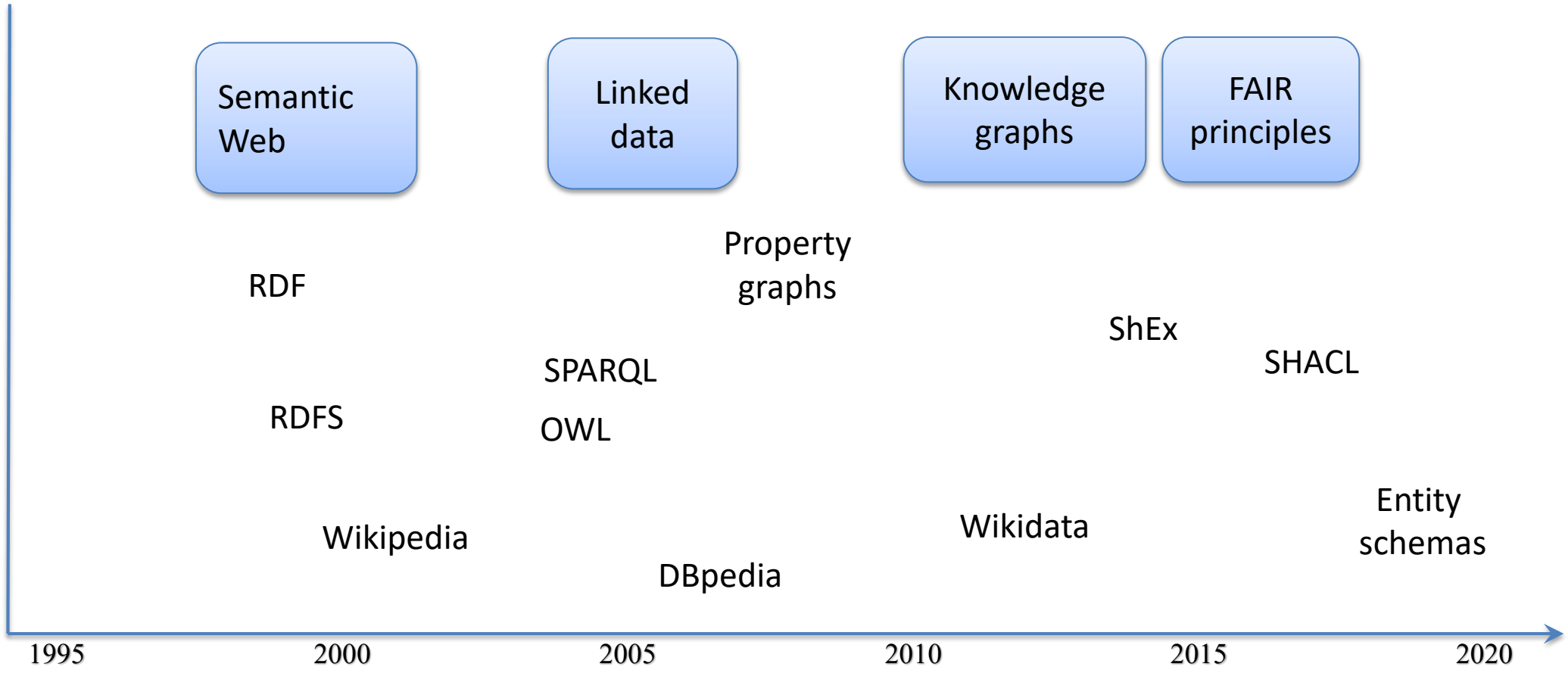


2021, HTML version
https://kgbook.org/

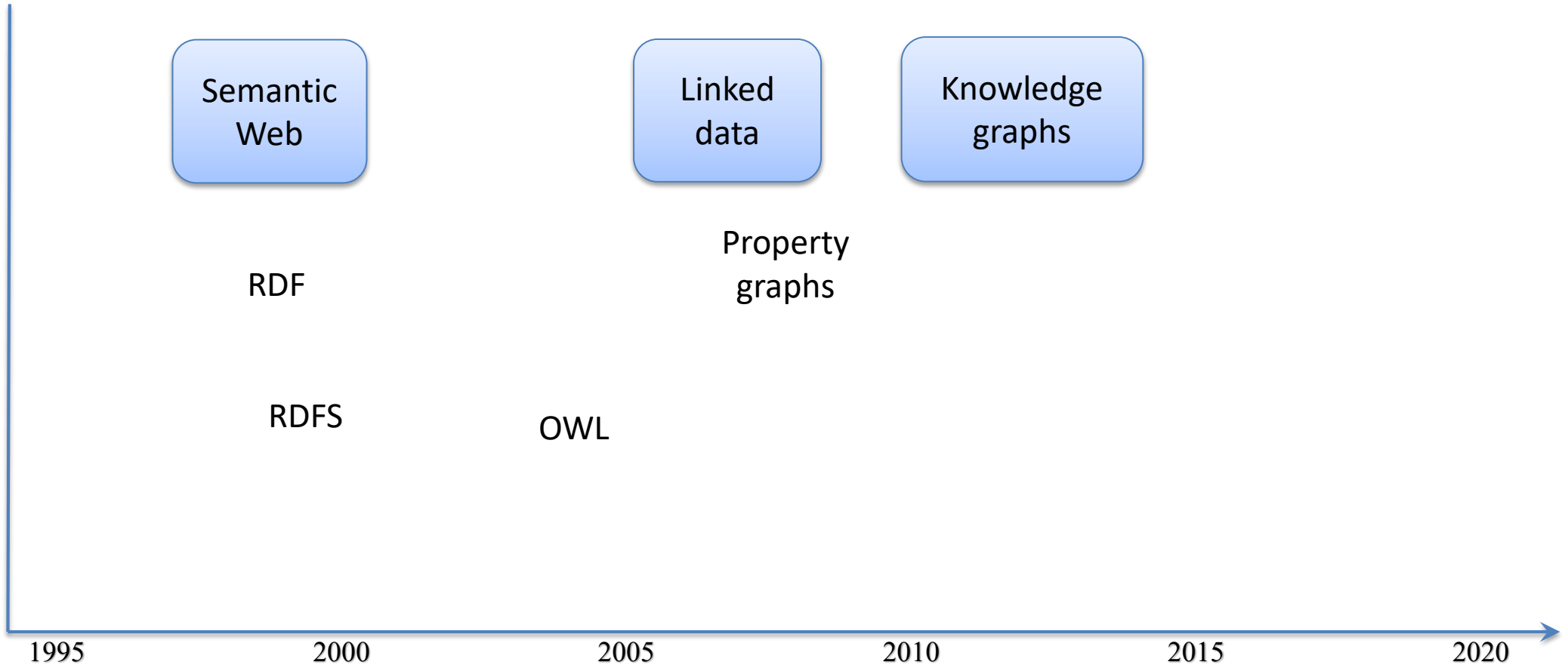
Schedule

Day	Title	Topics
Day 1.	Semantic Technologies and Knowledge graphs	Semantic Web Linked data Knowledge graphs RDF data model Property graphs Wikibase graphs Examples and applications
Day 2.	RDF data modelling and SPARQL	Data modelling exercises with RDF and turtle SPARQL
Day 3.	Validating RDF data	Shape Expressions (ShEx) SHACL Validating Knowledge Graphs
Day 4.	Advanced topics	ShEx and SHACL compared Reasoning RDFS OWL Nanopublications

Roadmap


Semantic Web
Linked data
Knowledge graphs
FAIR principles
Property graphs
RDF
ShEx
SHACL
SPARQL
RDFS
OWL
Entity schemas
Wikipedia
Wikidata
DBpedia
1995
2000
2005
2010
2015
2020

Roadmap


Semantic Web
Linked data
Knowledge graphs
Property graphs
RDF
RDFS
OWL
1995
2000
2005
2010
2015
2020

Semantic Web

Semantic Web = vision of the data Web

Goal: Automatically share and reuse data on the Web

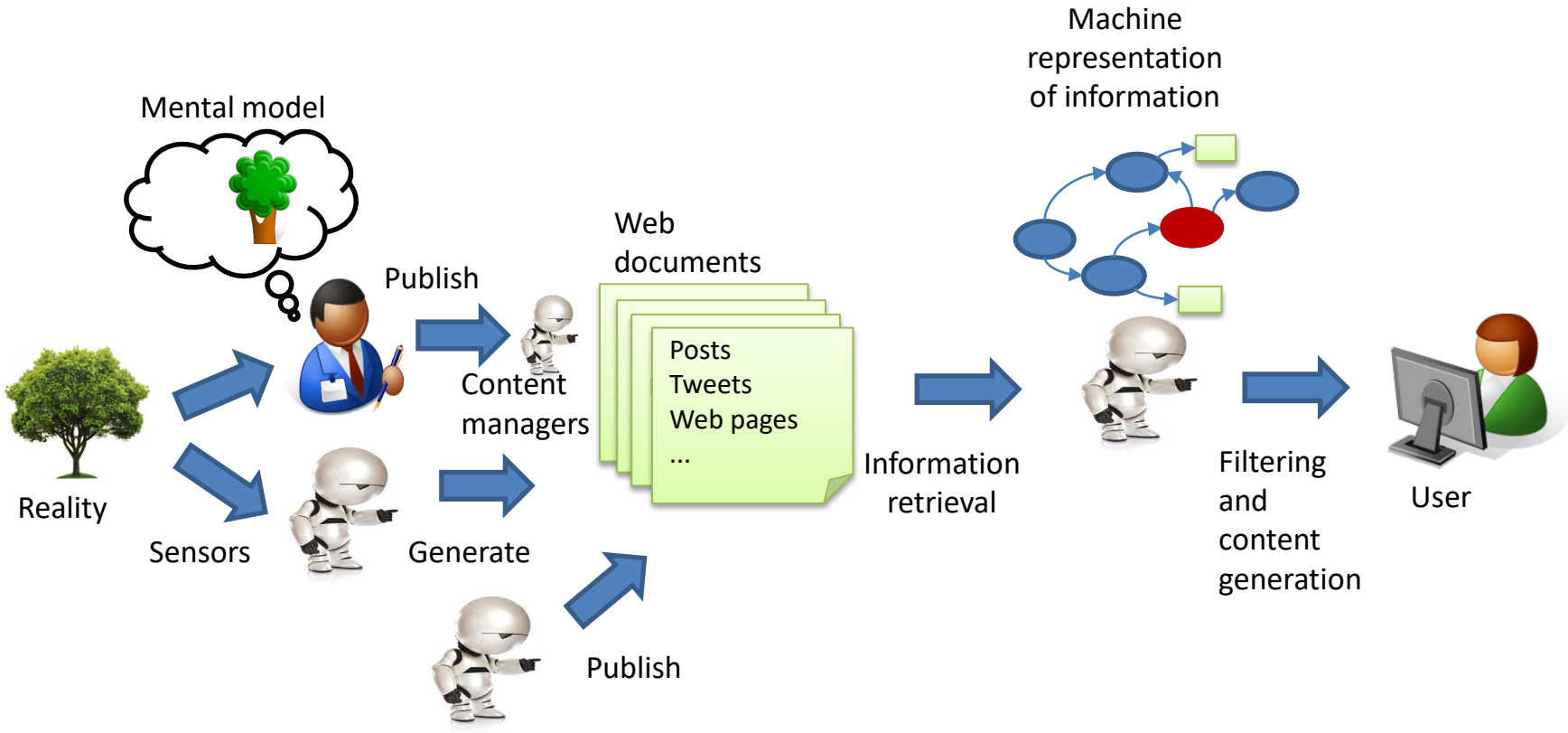
Not only documents/web pages, but also data

Long term project

- The goal is not to destroy the current web
  - Gradually adopt better practices
- Improve publication/reuse techniques
- Let machines process (understand?) more data
- Give more value to information

Tim Berners-Lee
Source: Wikipedia

Machines in the Web



Web information is constantly manipulated by machines
Semantic web goal ⇒ facilitate that manipulation

¿People vs Machines?

Creativity, imagination
Unpredictable (make errors)
Get tired with repetitive tasks
Understanding based on context

Programmed for certain tasks
Predictable (without errors*)
No problem for repetitive tasks
Dificulties to understand the context

*when they are well programmed

Representing context for machines

Example: "Oviedo has a temperature of 36"

Decomposed in:

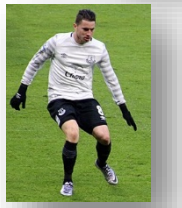
Oviedo...has a temperature of...36

Oviedo?

Oviedo, a city in Spain

Oviedo, another city in Florida, USA

Bryan Oviedo, a football player



How can we remove ambiguity?

Using URIS

https://www.oviedo.es/

https://www.cityofoviedo.net/

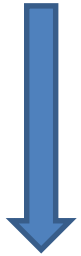
https://twitter.com/bryan_oviedo

Representing context for machines

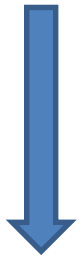
Example: "Oviedo has a temperature of 36"

Decomposed in:

Oviedo...has a temperature of...36



...has a temperature of... http://example.org/hasTemperature



Is there an existing URI
for "hasTemperature" property?

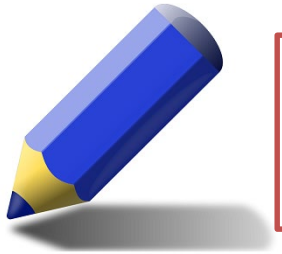
http://purl.obolibrary.org/obo/PATO_0000146

Representing context for machines

Example: "Oviedo has a temperature of 36"

Decomposed in:

Oviedo...has a temperature of...36



Is the previous representation accurate enough?
- Identify some problems about the previous definition...
- Identify several possibilities to improve that representation

RDF

RDF = Resource description framework

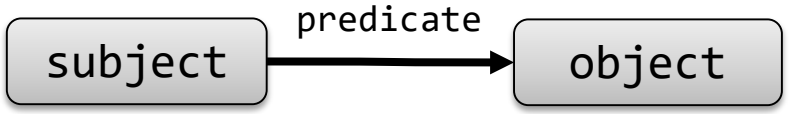
Based on triples and URIs to represent properties and nodes

Short history

- Around 1997 - PICS, Dublin core, Meta Content Framework
- 1997 1st Working draft https://www.w3.org/TR/WD-rdf-syntax-971002, RDF/XML
- 1999 1st W3C Rec https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, XML Syntax, first applications RSS, EARL
- 2004 - RDF Revised https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, SPARQL, Turtle, Linked Data
- 2014 - RDF 1.1 https://www.w3.org/TR/rdf11-concepts/, SPARQL 1.1, JSON-LD

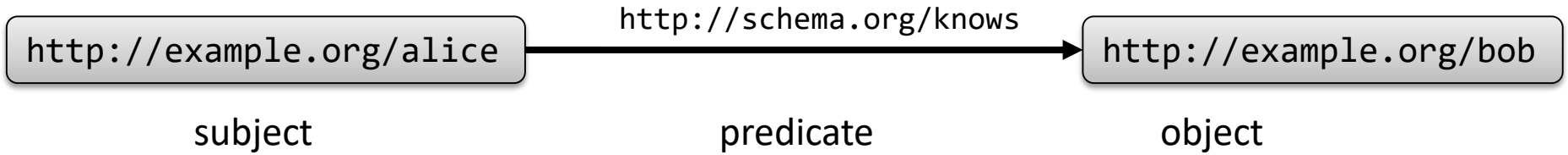
RDF Data Model

RDF is made from statements



Statment = a triple (subject, predicate, object)

Example:



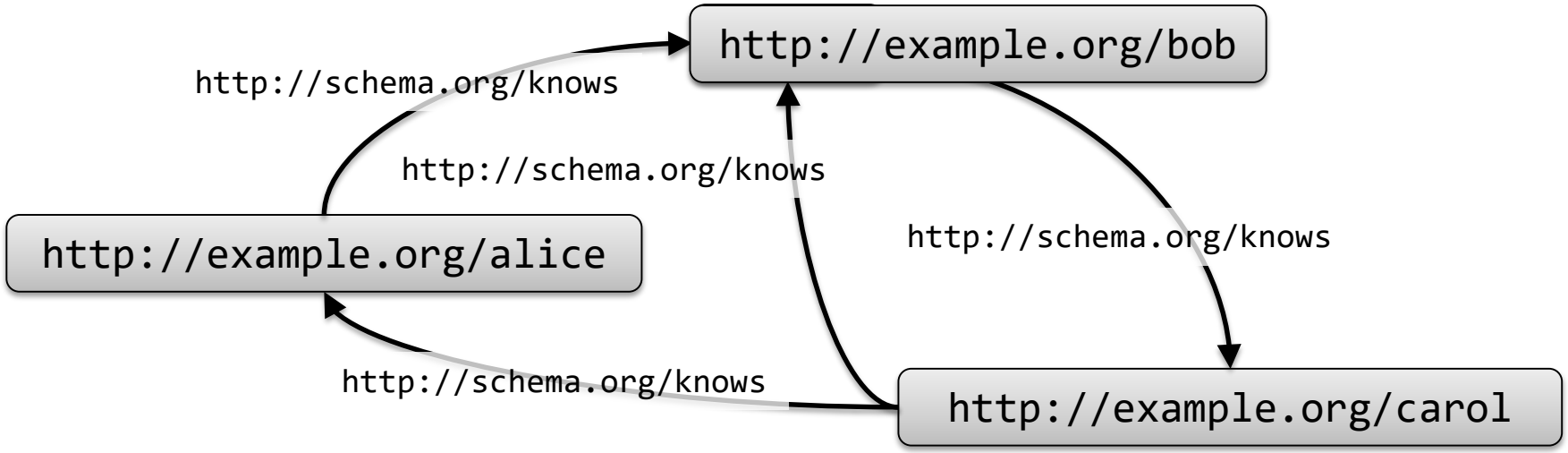
N-Triples representation

```
<http://example.org/alice> <http://schema.org/knows> <http://example.org/bob> .
```

Set of statements = RDF graph

RDF data model = directed graph

Example:



N-triples representation

```
<http://example.org/alice>   <http://schema.org/knows>   <http://example.org/bob> .
<http://example.org/bob>     <http://schema.org/knows>   <http://example.org/carol> .
<http://example.org/carol>   <http://schema.org/knows>   <http://example.org/alice> .
<http://example.org/carol>   <http://schema.org/knows>   <http://example.org/bob> .
```

subject predicate object

Turtle notation

Human readable notation that simplifies N-Triples

Allows namespace declarations

N-Triples

```
<http://example.org/alice>  <http://schema.org/knows>  <http://example.org/bob> .
<http://example.org/bob>    <http://schema.org/knows>  <http://example.org/carol> .
<http://example.org/carol>  <http://schema.org/knows>  <http://example.org/alice> .
<http://example.org/carol>  <http://schema.org/knows>  <http://example.org/bob> .
```

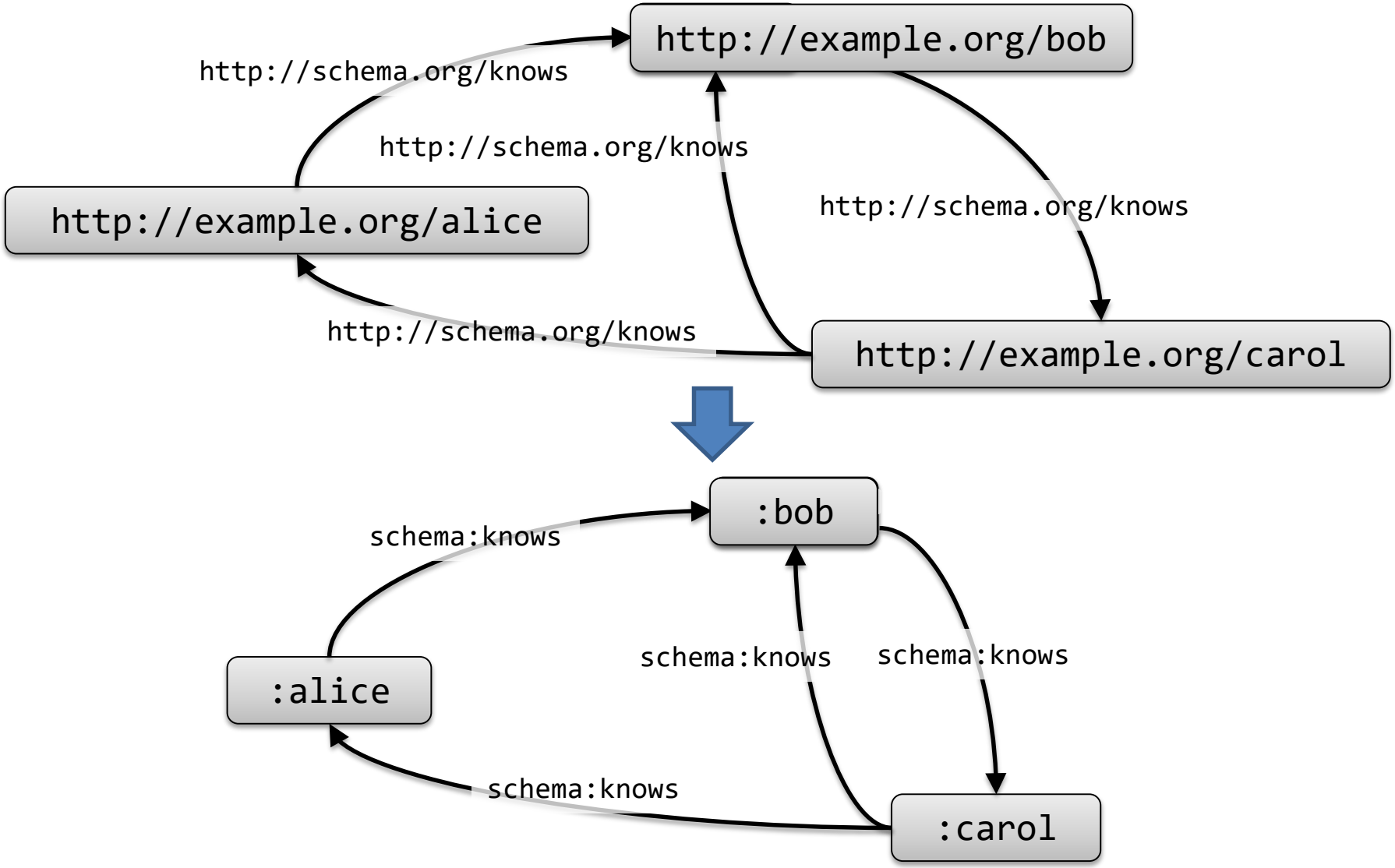
Turtle

```
prefix :       <http://example.org/>
prefix schema: <http://schema.org/>

:alice schema:knows :bob .
:bob   schema:knows :carol .
:carol schema:knows :bob .
:carol schema:knows :alice .
```

Note:
We will see later other Turtle simplifications

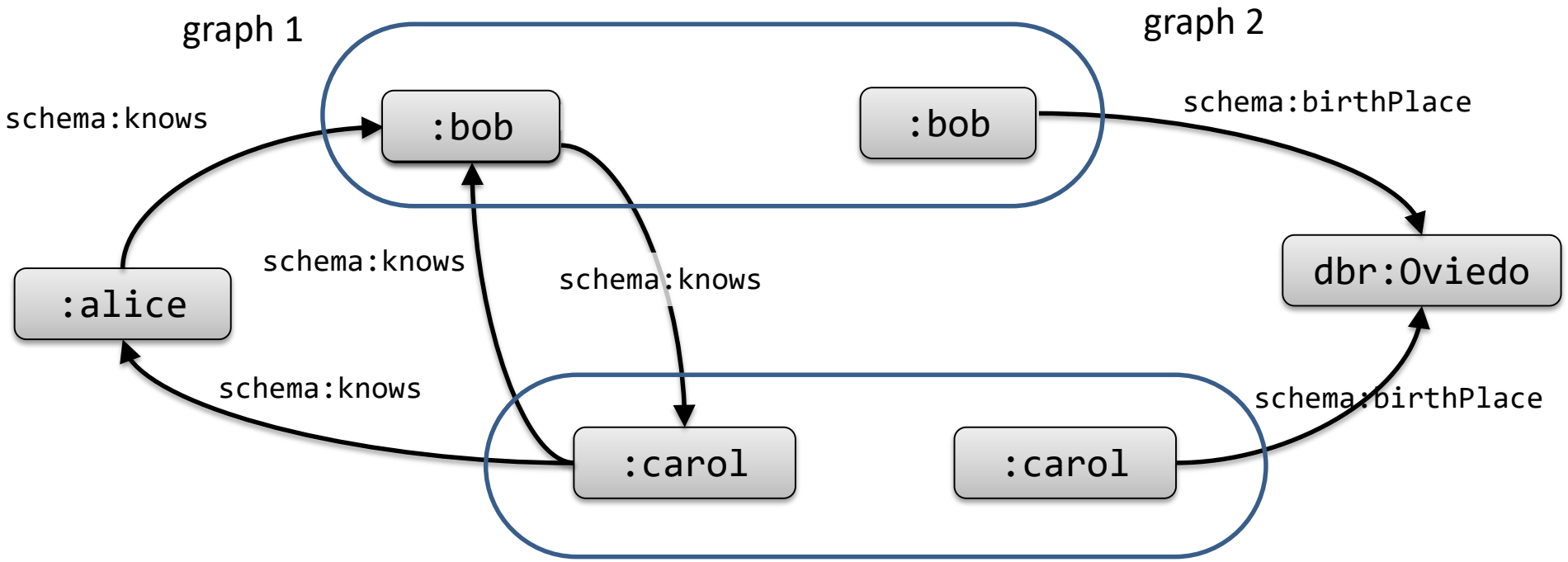
Namespaces simplification



RDF is compositional

RDF graphs can be merged to obtain a bigger graph

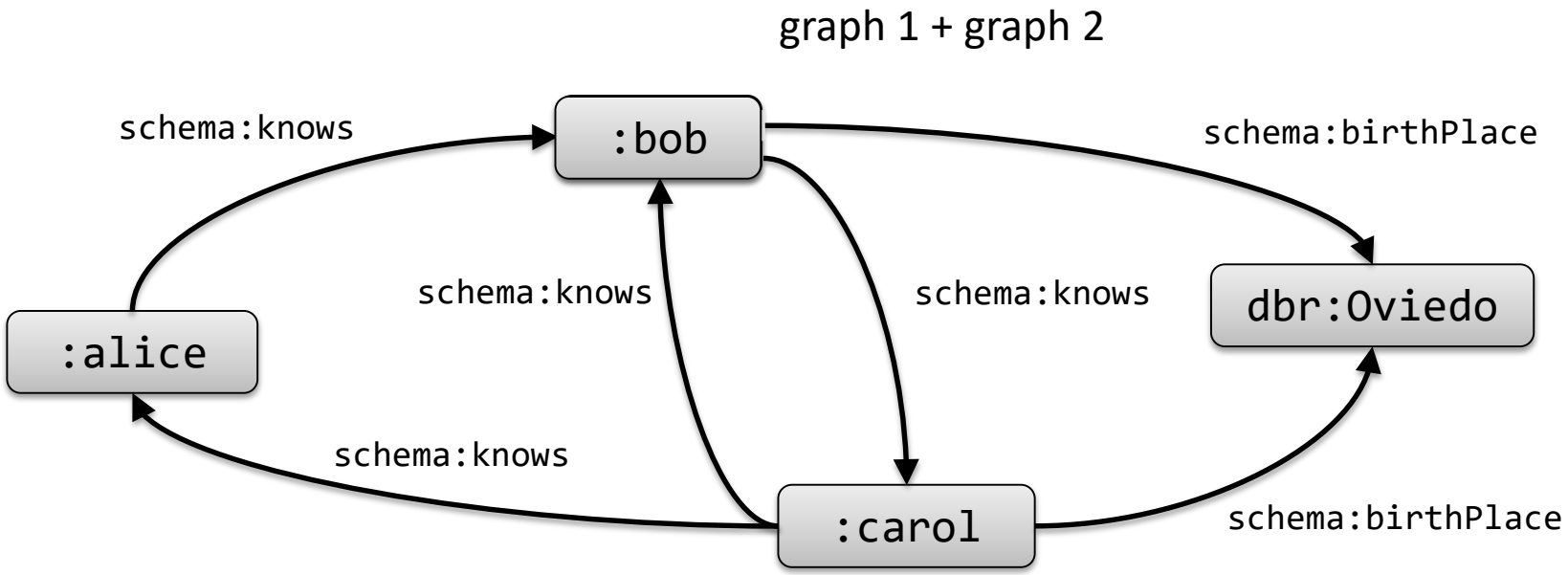
Automatic data integration



RDF is compositional

RDF graphs can be merged to obtain a bigger graph

Automatic data integration



Turtle syntax

Some simplifications

Prefix declarations

; when triples share the subject

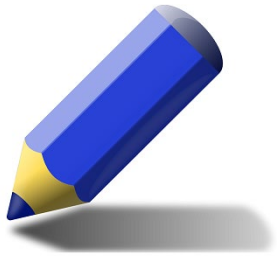
```
:alice schema:birthPlace dbr:Oviedo .
:alice schema:knows      :bob .
```

```
:alice schema:birthPlace dbr:Oviedo ;
       schema:knows      :bob .
```

, when triples share subject and object

```
:carol  schema:knows      :alice .
:carol  schema:knows      :bob .
```

```
:carol schema:knows :alice, :bob .
```



Turtle syntax

Exercise: simplify

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


:alice schema:knows      :bob .
:bob   schema:knows      :carol .
:carol schema:knows      :bob .
:carol schema:knows      :alice .
:bob   schema:birthPlace dbr:Spain .
:carol schema:birthPlace dbr:Spain .
```

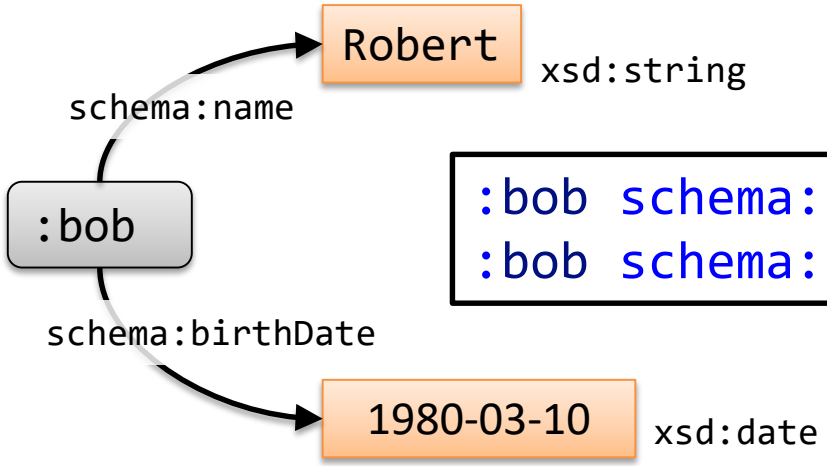
```
prefix ex:     <http://example.org/>
prefix schema: <http://schema.org/>
prefix dbr:    <http://dbpedia.org/resource>

:alice schema:knows      :bob , :carol.
:bob   schema:knows      :carol ;
       schema:birthPlace dbr:Spain .
:carol schema:knows      :bob, :alice ;
       schema:birthPlace dbr:Spain .
```

Try it: https://tinyurl.com/y9wbdycp

RDF Literals

- Objects can also be literals
  - Literals contain a lexical form and a datatype
  - Typical datatypes = XML Schema primitive datatypes
    - If not specified, a literal has datatype `xsd:string`

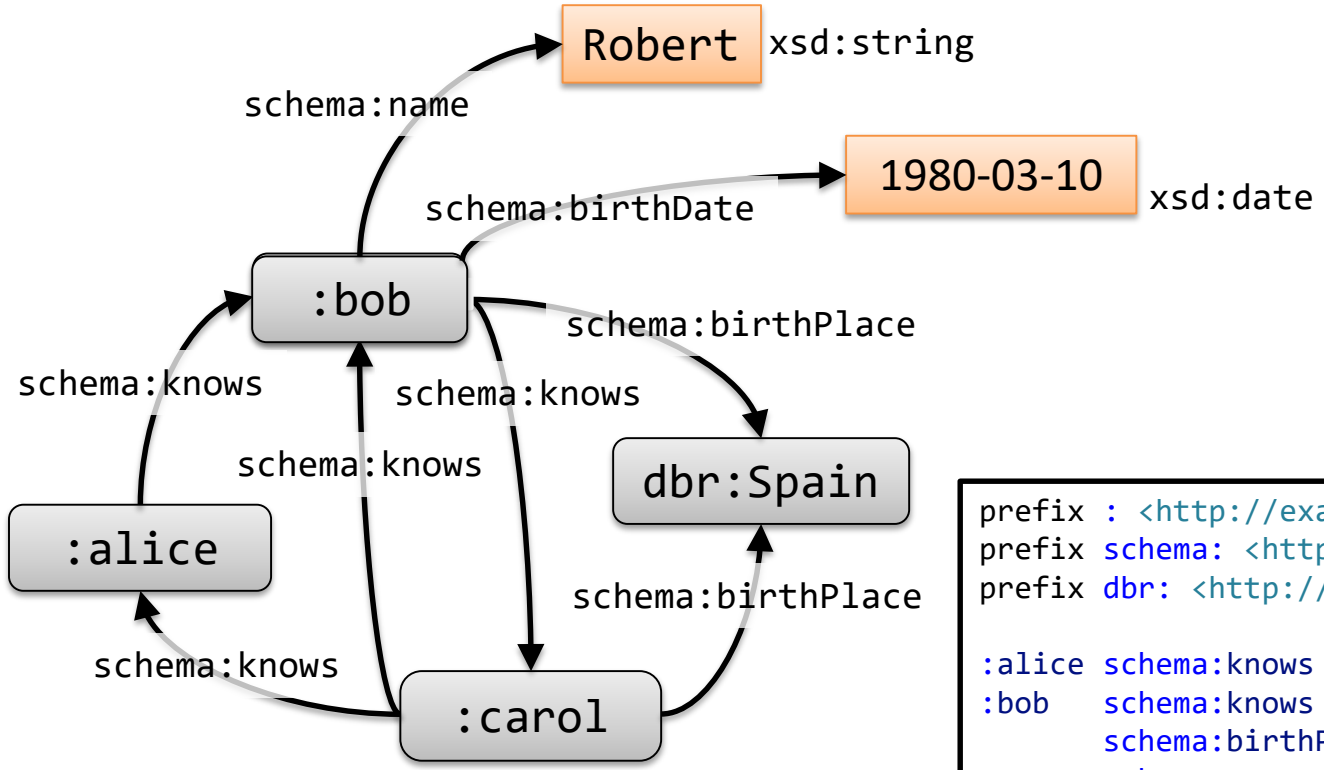


Turtle notation

```
:bob schema:name       "Robert" ;
:bob schema:birthDate "1980-03-10"^^<xsd:date>.
```

Remember...RDF is compositional

Merging previous data

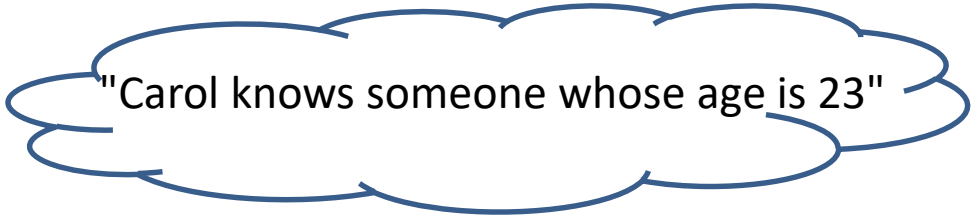



```
prefix : <http://example.org/>
prefix schema: <http://schema.org/>
prefix dbr: <http://dbpedia.org/resource>

:alice schema:knows      :bob , :carol.
:bob   schema:knows      :carol ;
       schema:birthPlace dbr:Spain;
       schema:name       "Robert";
       schema:birthDate  "1980-03-10"^^<xsd:date>.
:carol schema:knows      :bob, :alice ;
       schema:birthPlace dbr:Spain .
```

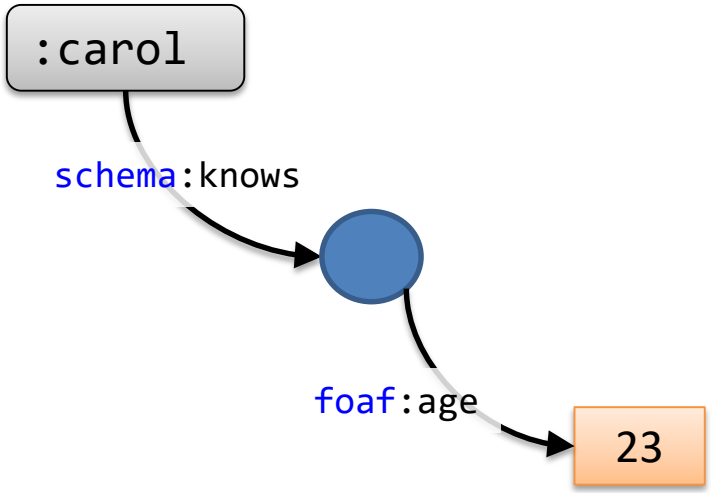
Blank nodes

Subjects and objects can also be Blank nodes



Turtle notation with local identifier

```
:carol schema:knows    _:x .
_:x    foaf:age        23 .
```



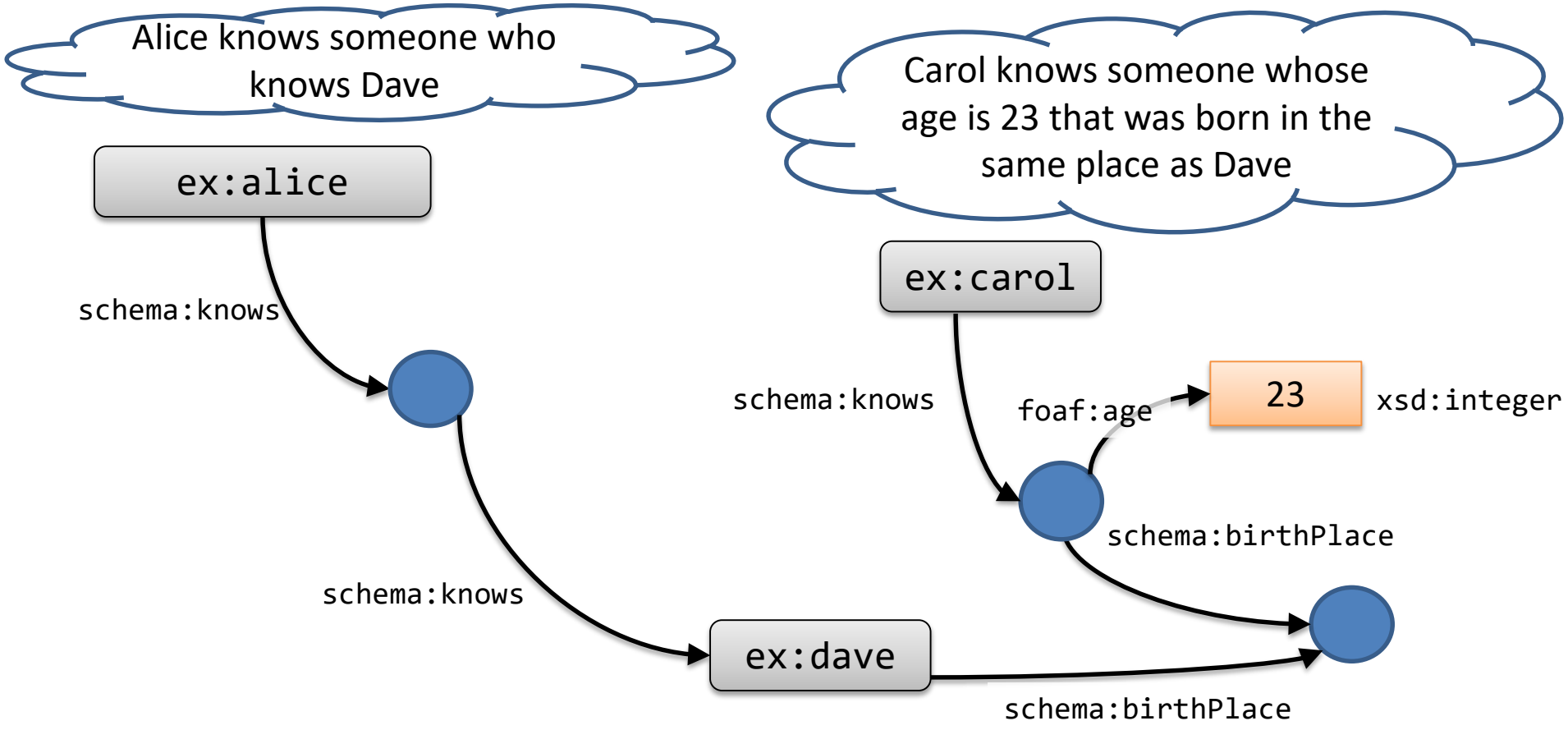
Turtle notation with square brackets

```
:carol schema:knows [foaf:age 23 ] .
```

Mathematical meaning:

$\exists x(\text{schema:knows}(\text{:carol}, x) \wedge \text{foaf:age}(x, 23))$

Blank nodes

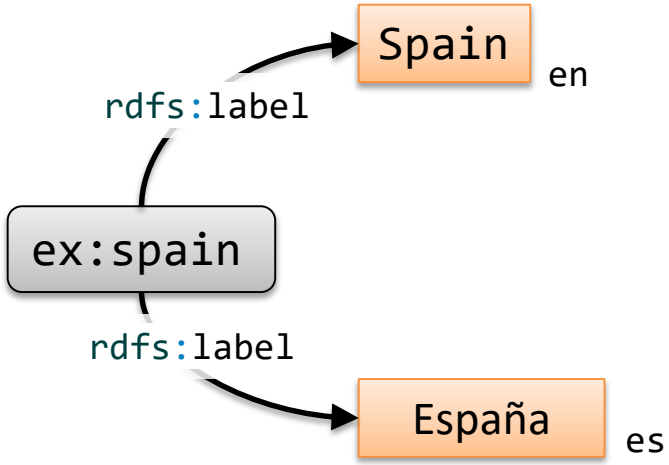



```
:alice schema:knows      [ schema:knows :dave ] .
:carol schema:knows      [ :age                23 ;
                           schema:birthPlace _:p ] .
:dave schema:birthPlace _:p .
```

Language tagged strings

String literals can be qualified by a language tag

They have datatype `rdfs:langString`

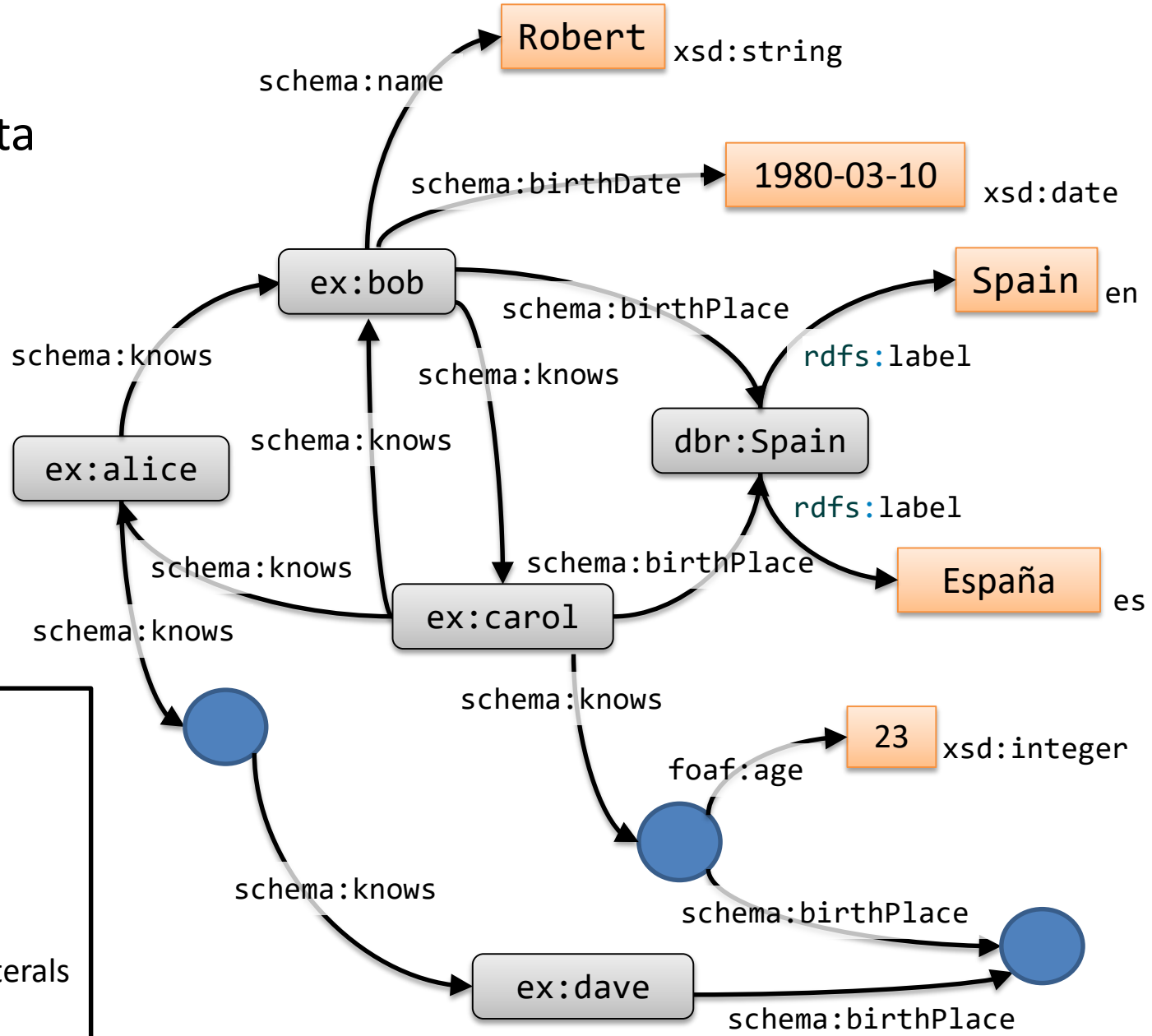


Turtle notation

```
ex:spain rdfs:label "Spain"@en .
ex:spain rdfs:label "España"@es .
```

RDF data model

Example of RDF data



3 types of nodes
- IRIs
- Blank nodes
- Literals

Subjects: URIs or Blank nodes
Objects: URIs, Blank nodes or literals
Predicates always URIs

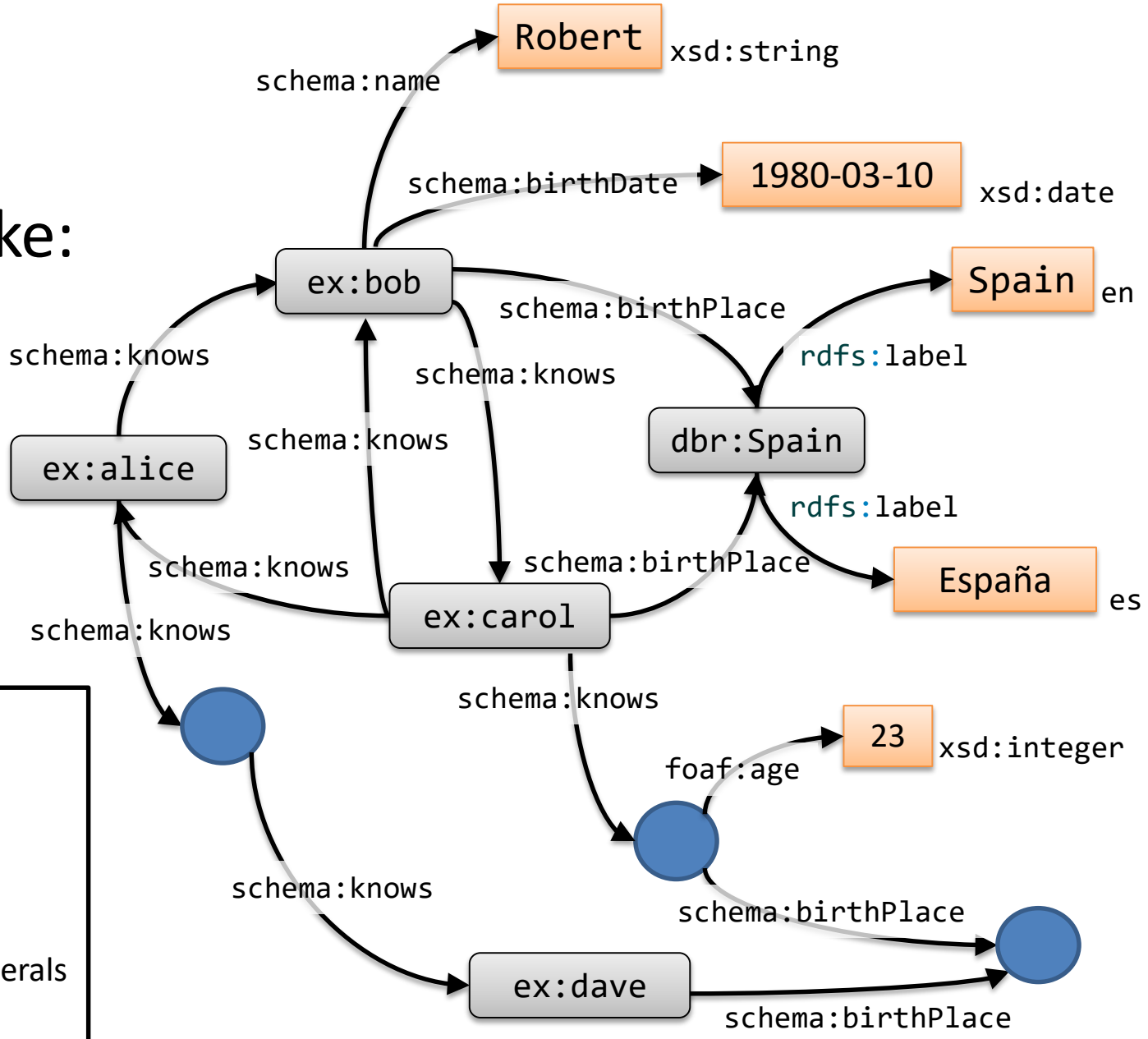
Formal definition of RDF data model

Most papers have something like:

Given a set of IRIs $\mathcal{I}$,
a set of blank nodes $\mathcal{B}$
and a set of literals Lit
an *RDF graph* is a tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{P}, \mathcal{O}, \rho \rangle$
where
$\mathcal{S} = \mathcal{I} \cup \mathcal{B}$,
$\mathcal{P} = \mathcal{I}$,
$\mathcal{O} = \mathcal{I} \cup \mathcal{B} \cup Lit$
$\rho \subseteq \mathcal{S} \times \mathcal{P} \times \mathcal{O}$

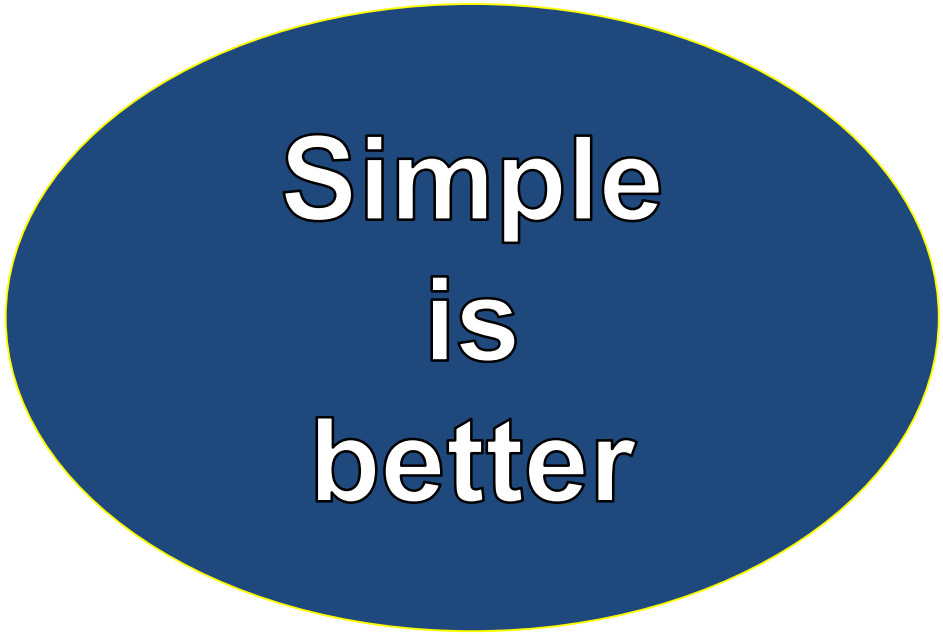
3 types of nodes
- IRIs
- Blank nodes
- Literals

Subjects: URIs or Blank nodes
Objects: URIs, Blank nodes or literals
Predicates always URIs



...and that's all about the RDF data model

The RDF Data model is very simple



RDF ecosystem

RDF Syntaxes (RDF/XML, N-Triples, Turtle, JSON-LD,...)

Shared entities, vocabularies, ontologies (RDFS, OWL, SKOS,...)

Query language (SPARQL, TripleStores...)

RDF description and validation (ShEx, SHACL)

RDF syntaxes

First syntax based on XML: RDF/XML

N-Triples (enumerates all triples separated by dots)

Turtle (human readability)

JSON-LD

...other syntaxes...

....lots of syntaxes but a unique data model

RDF/XML

First syntax

Not very popular

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://example.org/"
         xmlns:schema="http://schema.org/">
<rdf:Description rdf:about="http://example.org/carol">
 <schema:knows>
  <rdf:Description rdf:about="http://example.org/bob">
   <schema:knows rdf:resource="http://example.org/carol"/>
   <schema:name>Robert</schema:name>
   <schema:birthDate rdf:datatype="xsd:date">1980-03-10</schema:birthDate>
  </rdf:Description>
 </schema:knows>
 <schema:knows>
  <rdf:Description rdf:about="http://example.org/alice">
   <schema:knows rdf:resource="http://example.org/bob"/>
   <schema:knows rdf:resource="http://example.org/carol"/>
  </rdf:Description>
 </schema:knows>
 <schema:knows rdf:parseType="Resource">
  <age rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">23</age>
 </schema:knows>
 </rdf:Description>
</rdf:RDF>
```

N-Triples

For testing and easy parsing

...just triples separated by dots

```
<http://example.org/carol> <http://schema.org/knows>     <http://example.org/bob> .
<http://example.org/carol> <http://schema.org/knows>     <http://example.org/alice> .
<http://example.org/carol> <http://schema.org/knows>     _:x .
_:x                        <http://example.org/age>      "23"^^<http://www.w3.org/2001/XMLSchema#integer> .
<http://example.org/alice> <http://schema.org/knows>     <http://example.org/bob> .
<http://example.org/alice> <http://schema.org/knows>     <http://example.org/carol> .
<http://example.org/bob>   <http://schema.org/knows>     <http://example.org/carol> .
<http://example.org/bob>   <http://schema.org/name>      "Robert" .
<http://example.org/bob>   <http://schema.org/birthDate> "1980-03-10"^^<xsd:date> .
```

Turtle

Concise

Designed to be human-readable

```
prefix :       <http://example.org/>
prefix schema: <http://schema.org/>

:alice schema:knows :bob , :carol .
:bob   schema:knows :carol ;
       schema:name "Robert";
       schema:birthDate "1980-03-10"^^<xsd:date>.
:carol schema:knows :bob, :alice ;
       schema:knows [ :age 23 ] .
```

JSON-LD

Json for Linked Data

```
{
"@context" : {
"knows" : { "@id" : "http://schema.org/knows", "@type" : "@id" },
"age" : { "@id" : "http://example.org/age",
          "@type" : "http://www.w3.org/2001/XMLSchema#integer" },
"name" : { "@id" : "http://schema.org/name" },
"birthDate" : { "@id" : "http://schema.org/birthDate", "@type" : "xsd:date" },
"@vocab" : "http://example.org/",
"schema" : "http://schema.org/"
},
"@graph" : [
 { "@id" : "http://example.org/alice",
  "knows" : [ "http://example.org/bob", "http://example.org/carol" ] },
 { "@id" : "http://example.org/bob",
  "birthDate" : "1980-03-10",
  "knows" : "http://example.org/carol",
  "name" : "Robert" },
 { "@id" : "http://example.org/carol",
  "knows" : [ "http://example.org/bob", "http://example.org/alice", "_:x" ] },
 { "@id" : "_:x",
  "http://example.org/age" : 23 }
 ]
}
```

Other Turtle simplifications

RDF type property

Numbers

Collections

RDF type property

The `rdf:type` property declares the type of a resource

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

e:alice rdf:type schema:Person .
e:bob   rdf:type schema:Person .
```

`rdf:type` can be simplified as `a`

```
@prefix schema: <http://schema.org/> .

:alice a schema:Person .
:bob   a schema:Person .
```

Constants

Numbers and boolean values can be represented without quotes

They are parsed as XML Schema datatypes

Datatype	Shorthand example	Lexical example
xsd:integer	3	"3"^^xsd:integer
xsd:decimal	-3.14	"-3.14"^^xsd:decimal
xsd:double	3.14e2	"3.14e2"^^xsd:double
xsd:boolean	true	"true"^^xsd:boolean

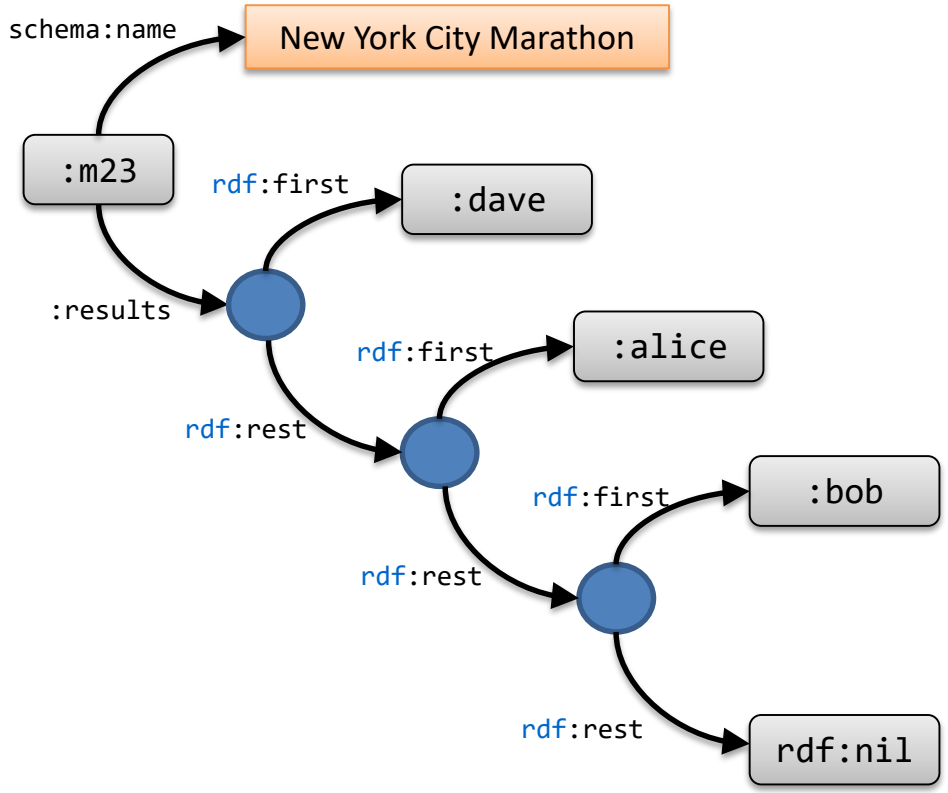
Collections

Ordered lists

```
:m23 schema:name "New York City Marathon ";
     :results    ( :dave :alice :bob ) .
```

Internally, represented as linked lists

```
:m23 schema:name "New York City Marathon ";
     :results _:1 .
_:1 rdf:first :dave ;
    rdf:rest _:2 .
_:2 rdf:first :alice ;
    rdf:rest _:3 .
_:3 rdf:first :bob ;
    rdf:rest rdf:nil .
```



SPARQL

SPARQL Protocol And RDF Query Language

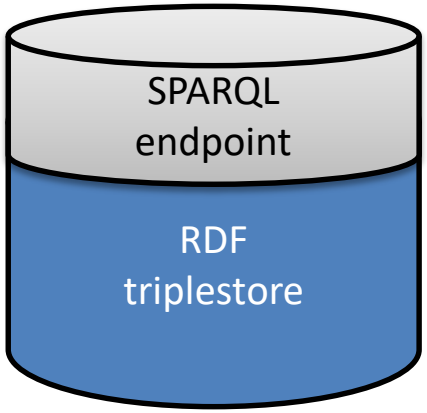
SPARQL 1.0 (2008), 1.1 (2013)

Syntax inspired by Turtle

Semantics based on graph patterns

SPARQL endpoints = services that implement SPARQL protocol

Triplestores = RDF databases

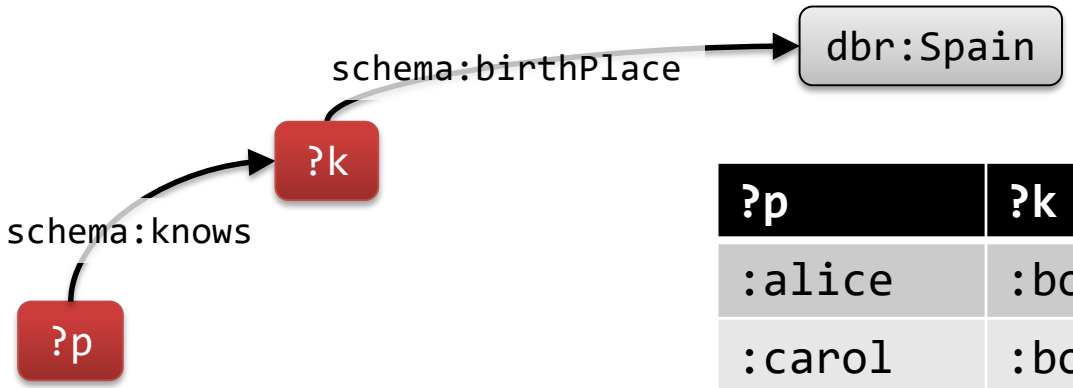


SPARQL example

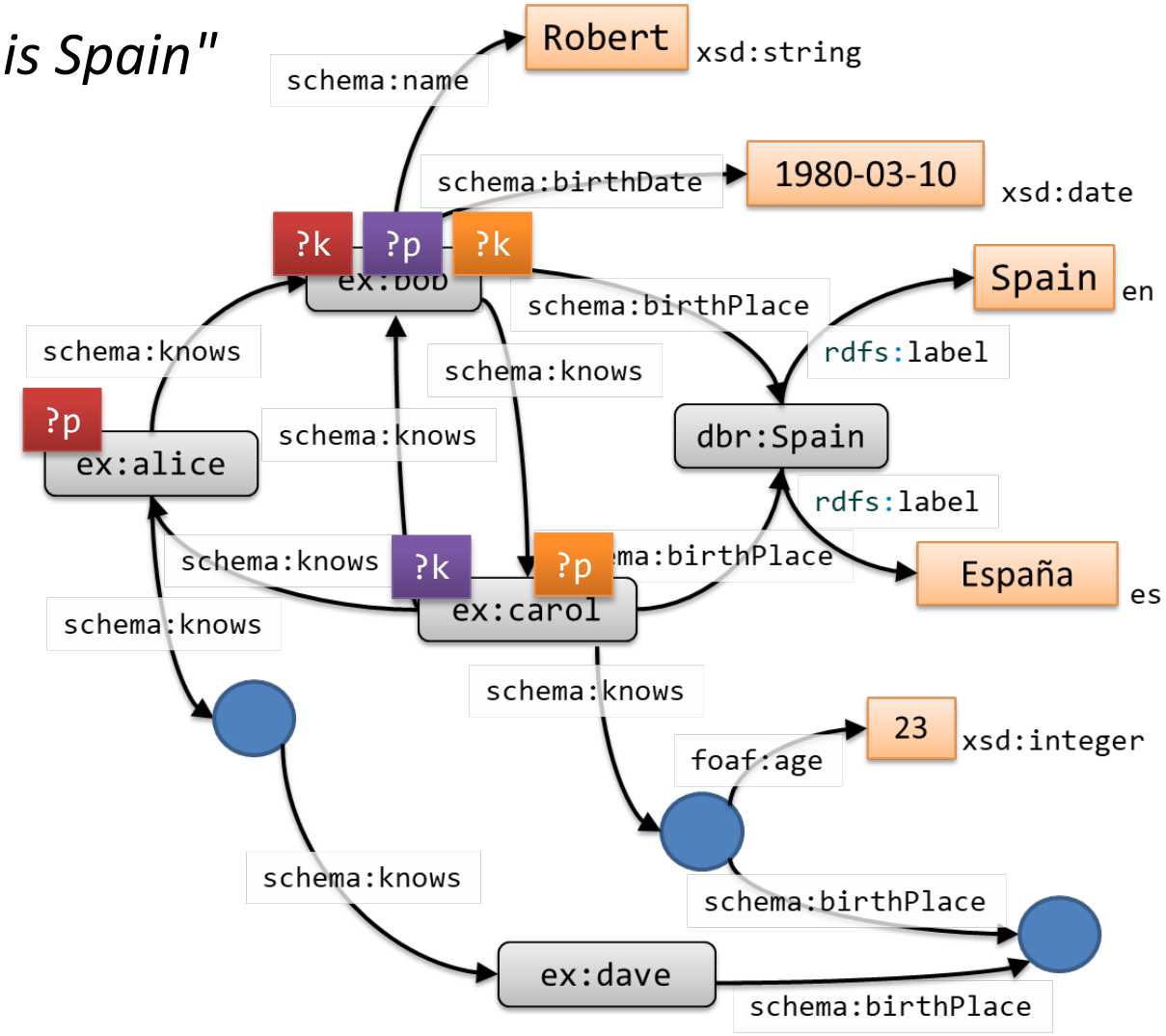
"People who knows someone whose birth place is Spain"

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

select ?p ?k where {
  ?p schema:knows      ?k  .
  ?k schema:birthPlace dbr:Spain
}
```



?p	?k
:alice	:bob
:carol	:bob
:bob	:carol



Shared entities and vocabularies

The use of URIs instead of plain strings facilitates:

- Merging data from heterogeneous sources
- Avoid ambiguity

Challenge: Agreeing on common entities and properties

Some popular vocabularies:

- schema.org: Joint effort from Google, Yahoo, Microsoft, Yandex
- Linked open vocabularies Project: http://lov.okfn.org/

Some popular vocabularies and namespaces

Alias	URL	Name	Some properties
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#	RDF	type, subject, predicate, object,…
rdfs:	http://www.w3.org/2000/01/rdf-schema#	RDF Schema	domain, range, Class, Property, subClassOf,…
owl:	http://www.w3.org/2002/07/owl#	OWL Ontologies	sameAs, intersectionOf, unionOf, …
dc:	http://purl.org/dc/elements/1.1/	Dublin Core	author, date, creator, …
Schema:	http://schema.org/	Schema.org	name, knows, etc.
skos:	http://www.w3.org/2008/05/skos#	SKOS	broader, narrower, …

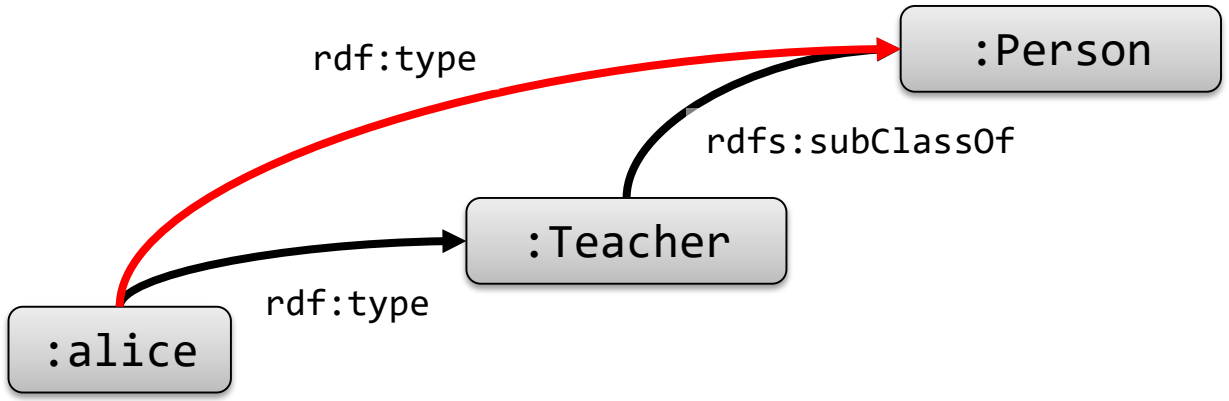
Service http://prefix.cc can be used to find the most popular prefix for some URI

The power of shared vocabularies

RDFS (Originally RDF Schema) defines common concepts

- Classes: `rdfs:Class`, `rdfs:Property`, `rdfs:Literal`
- Properties: `rdfs:domain`, `rdfs:range`, `rdfs:subClassOf`, …

RDFS processors can infer new triples



IF x `rdf:type` A AND A `rdfs:subClassOf` B
THEN x `rdf:type` B

From shared vocabularies to ontologies

OWL = Web Ontology Language.

OWL 1 (2004), OWL 2 (2009)

Based on description logics

Describe classes, properties, individuals and their relationships

Highly expressive and powerful inference mechanism

OWL example

Simple ontology, Terminological part (TBox)

$$Person \sqsubseteq = 2\, hasParent$$
$$Person \sqsubseteq \exists\, hasParent\, Male$$
$$Person \sqsubseteq \exists\, hasParent\, Female$$

$$Male \sqsubseteq \neg Female$$
$$Female \sqsubseteq \neg Male$$

```
:Person rdf:type owl:Class ;
 rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty :hasParent ; owl:cardinality 2
 ], [
 rdf:type owl:Restriction ;
 owl:onProperty :hasParent ; owl:someValuesFrom :Male
 ], [
 rdf:type owl:Restriction ;
 owl:onProperty :hasParent ; owl:someValuesFrom :Female
 ] .

:Female owl:disjointWith :Male .
```

Instance data, assertional part = ABox

$$Person(alice)$$
$$hasParent(alice, bob)$$
$$hasParent(alice, carol)$$
$$Female(carol)$$

```
:alice rdf:type :Person ;
 :hasParent :bob,
            :carol .
:carol rdf:type :Female .
```

Inferred data

$$Male(bob)$$

```
:bob rdf:type :Male .
```

The example is not complete... what is missing?

We need to declare that :bob ≠ :carol

```
[ rdf:type owl:AllDifferent ;
  owl:distinctMembers ( :bob
                        :carol
                      ) ] .
```

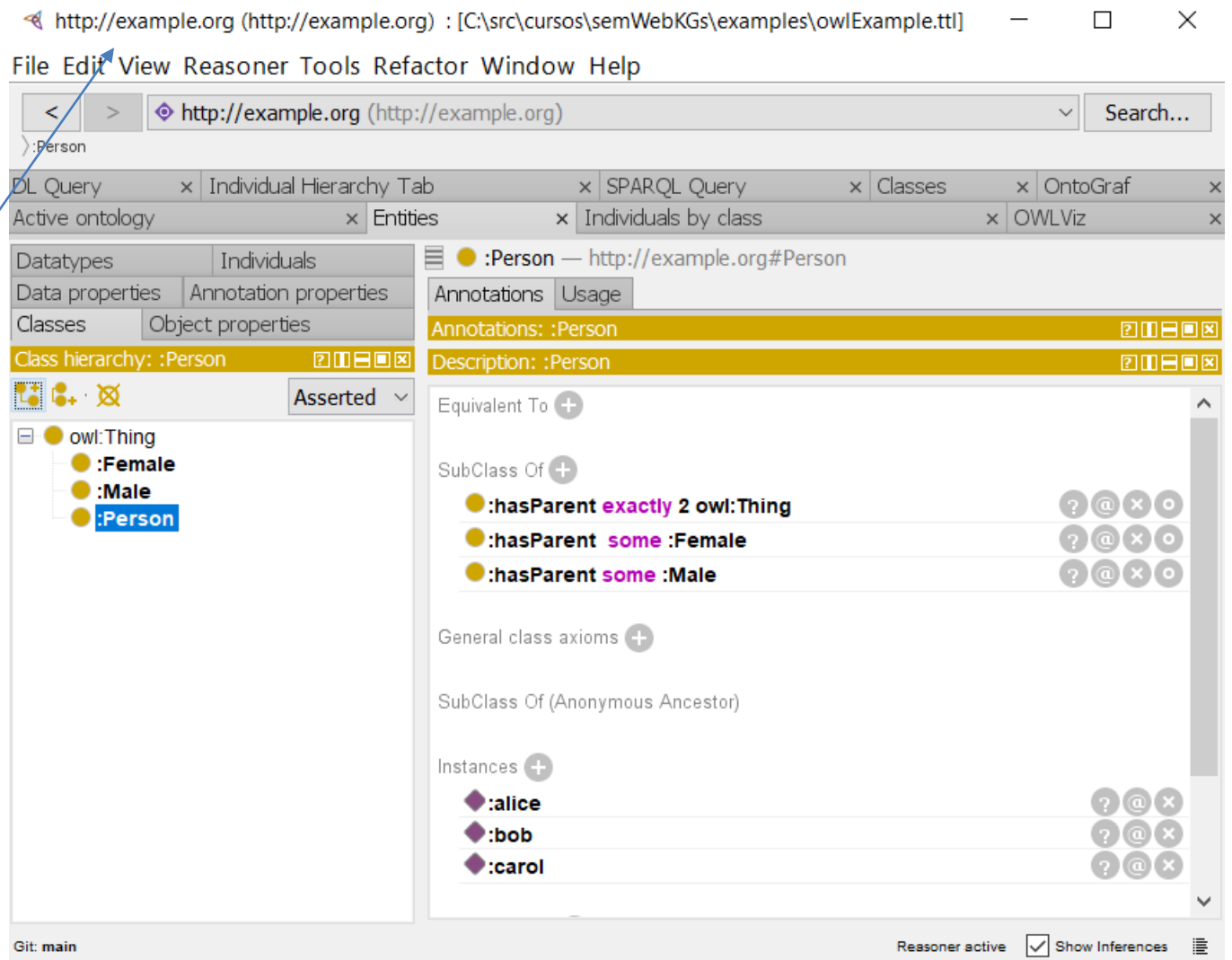
OWL

OWL = language to describe ontologies

Different kinds of ontologies:

- Upper level ontologies (SUMO, BFO, ...)
- Domain specific ontologies

Ontology editors like [Protégé editor]



Ontology concepts have a URI
They can be defined/stored in local files
How should we publish them?

Linked Data

Principles proposed by Tim Berners-Lee to publish data*:

1. Use URIs to denote things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs. so that they can discover more things

* https://www.w3.org/DesignIssues/LinkedData.html

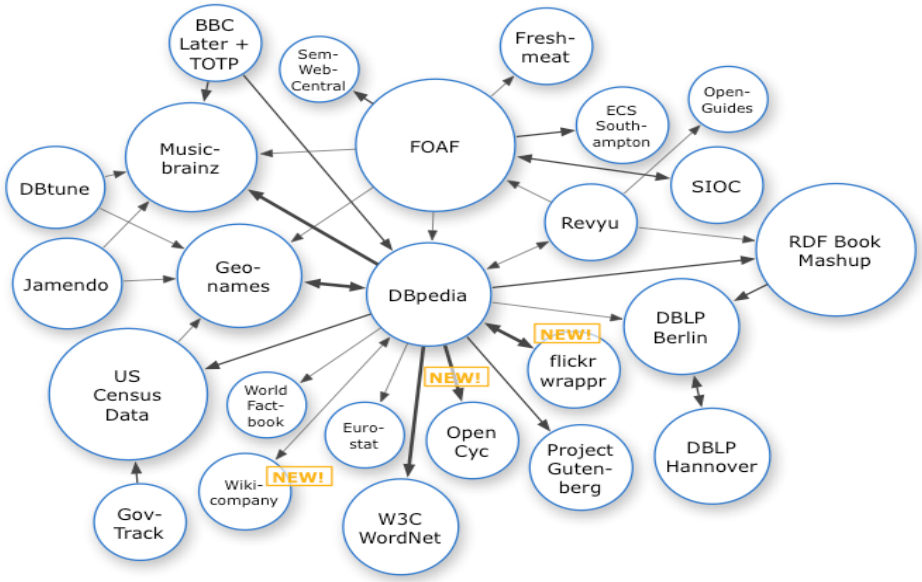
Why Linked data?

Best practices to publish data on the Web

To avoid the use of URIs as plain identifiers

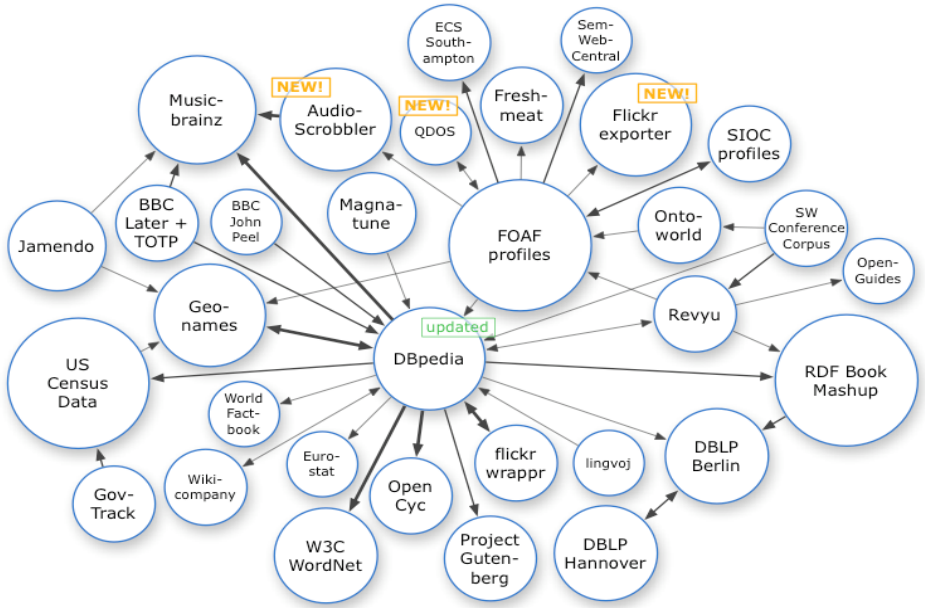
- A pattern very common when designing OWL ontologies
- URIs were mainly used as identifiers
- When dereferencing such URIs no useful information was retrieved
- It was breaking the idea of the web as an interlinked space

LOD (2007)

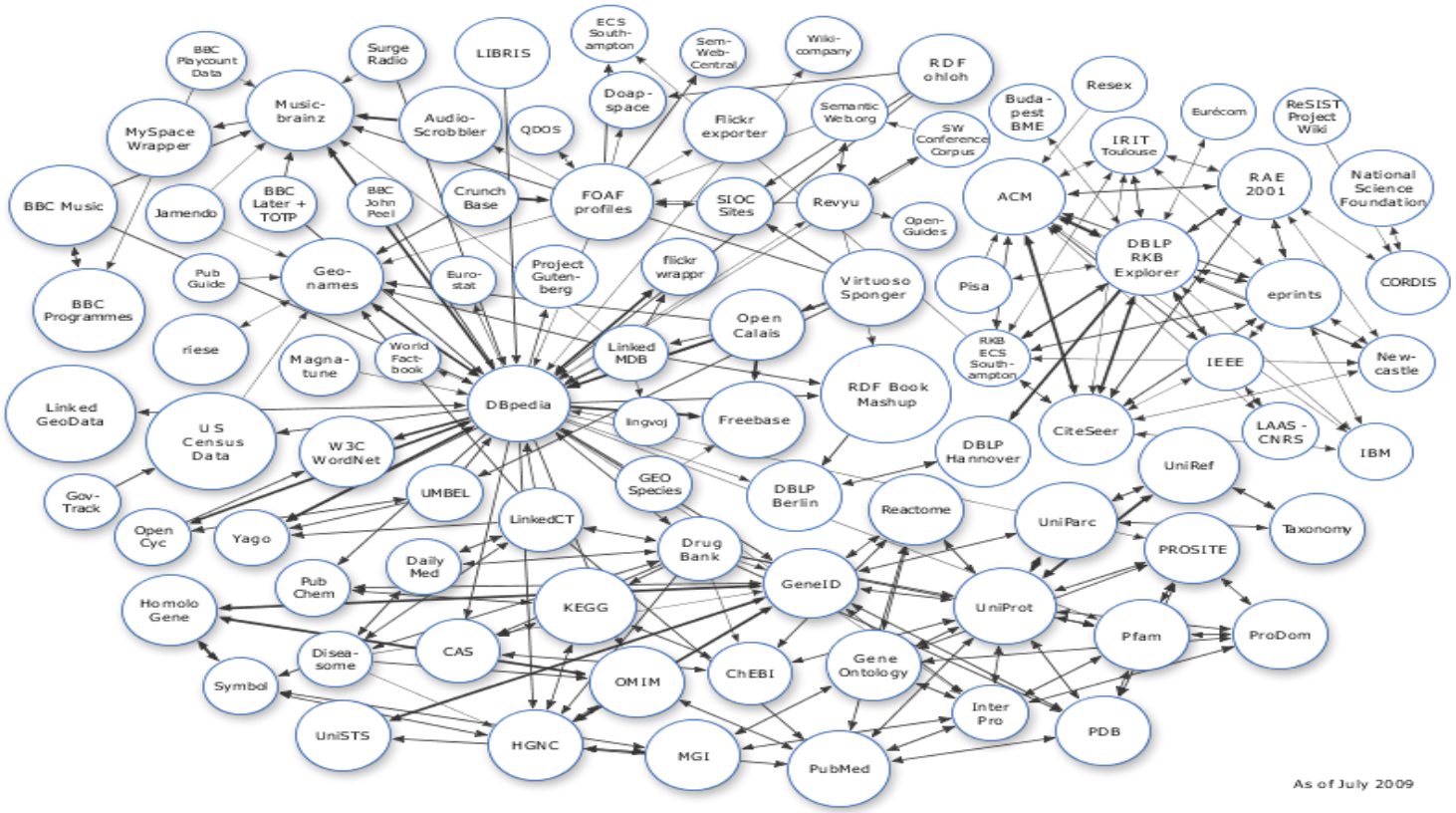


Evolución: https://lod-cloud.net/

LOD (2008)

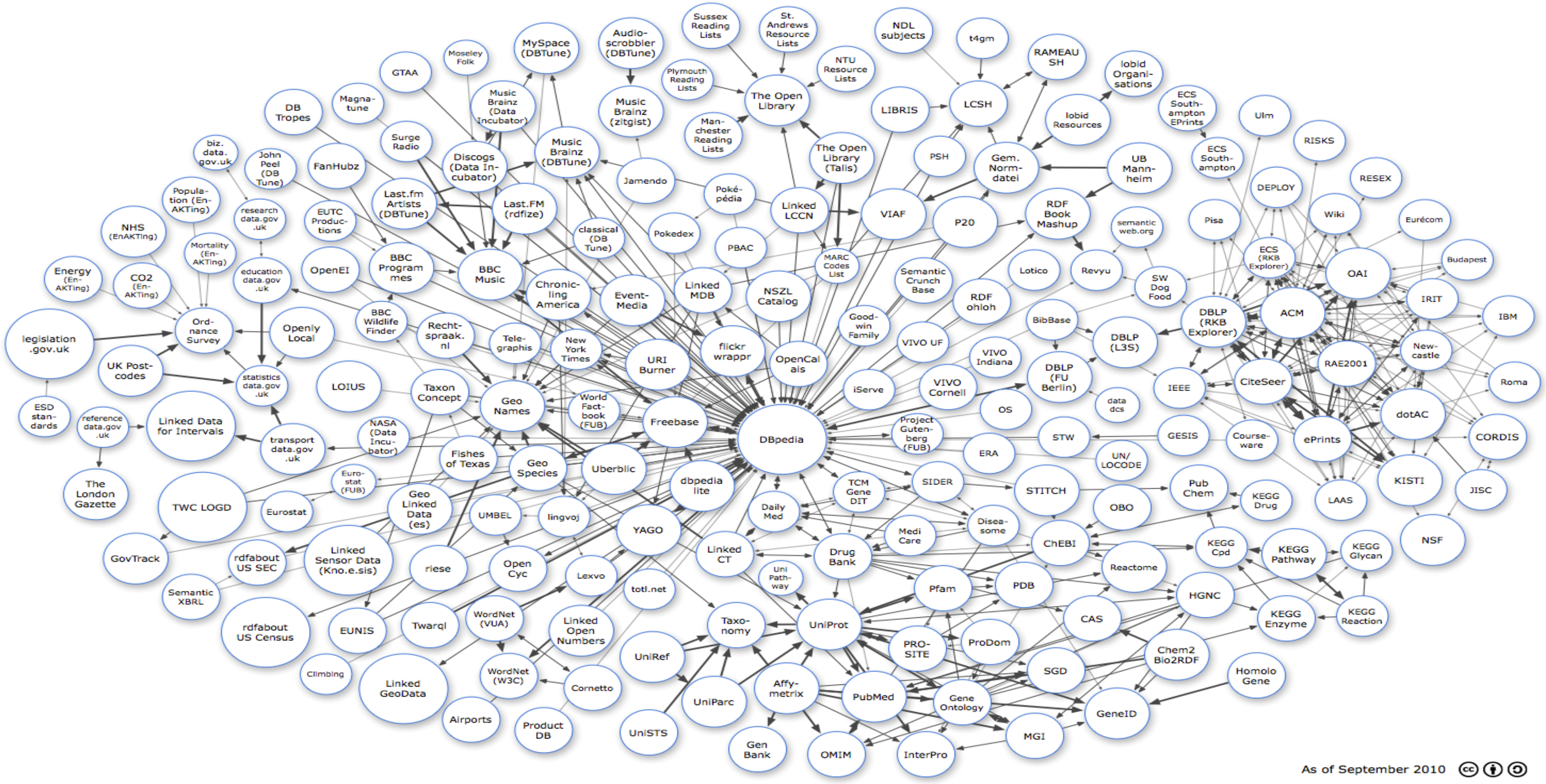

ECS South-ampton
Sem-Web-Central
Music-brainz
NEW!
Audio-Scrobbler
NEW!
QDOS
Fresh-meat
NEW!
Flickr exporter
SIOC profiles
BBC Later + TOTP
BBC John Peel
Magna-tune
FOAF profiles
Onto-world
SW Conference Corpus
Jamendo
Open-Guides
Geo-names
updated
Revyu
US Census Data
DBpedia
RDF Book Mashup
World Fact-book
Euro-stat
flickr wrappr
lingvoj
DBLP Berlin
Wiki-company
Open Cyc
Gov-Track
W3C WordNet
Project Guten-berg
DBLP Hannover

LOD (2009)

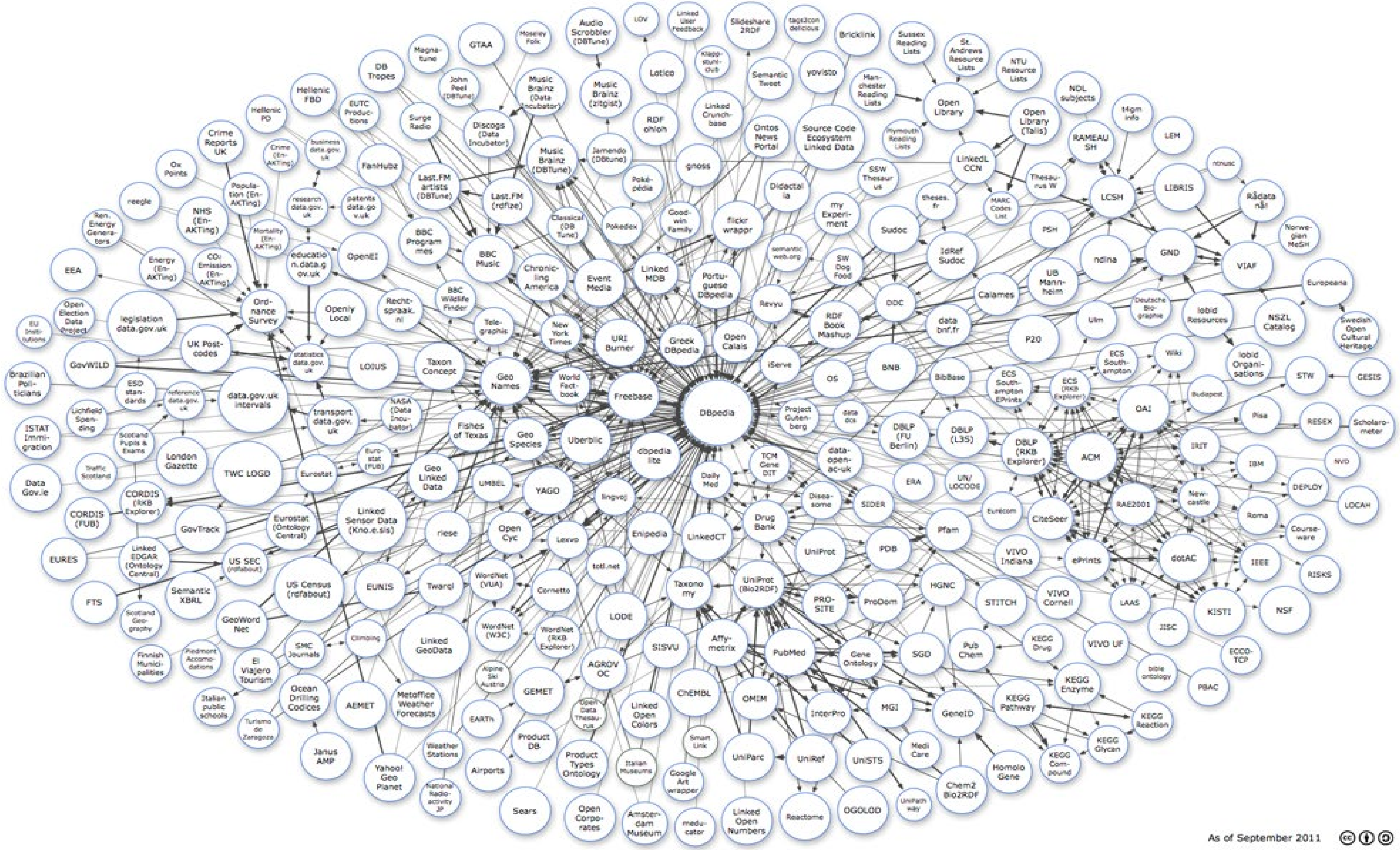

BBC Playcount Data
Surge Radio
LIBRIS
ECS South-ampton
Sem-Web-Central
Wiki-company
RDF ohloh
Resex
Buda-pest BME
Eurécom
ReSIST Project Wiki
Music-brainz
Audio-Scrobbler
QDOS
Doap-space
Flickr exporter
Semantic Web.org
SW Conference Corpus
IRIT Toulouse
MySpace Wrapper
BBC Music
Jamendo
BBC Later + TOTP
BBC John Peel
Crunch Base
FOAF profiles
SIOC Sites
Revyu
Open-Guides
ACM
RAE 2001
National Science Foundation
DBLP RKB Explorer
Pub Guide
Geo-names
Euro-stat
Project Guten-berg
flickr wrappr
Virtuoso Sponger
Pisa
eprints
CORDIS
BBC Programmes
riese
Magna-tune
World Fact-book
Open Calais
Linked MDB
RKB ECS South-ampton
IEEE
New-castle
Linked GeoData
US Census Data
W3C WordNet
DBpedia
lingvoj
Freebase
RDF Book Mashup
CiteSeer
LAAS - CNRS
IBM
Gov-Track
UMBEL
GEO Species
DBLP Hannover
UniRef
Open Cyc
Yago
LinkedCT
DBLP Berlin
Reactome
UniParc
Taxonomy
Daily Med
Drug Bank
PROSITE
Pub Chem
GeneID
Homolo Gene
KEGG
UniProt
Pfam
ProDom
Disea-some
CAS
Gene Ontology
Symbol
OMIM
ChEBI
Inter Pro
PDB
UniSTS
HGNC
MGI
PubMed
As of July 2009

LOD (2010)

WESO

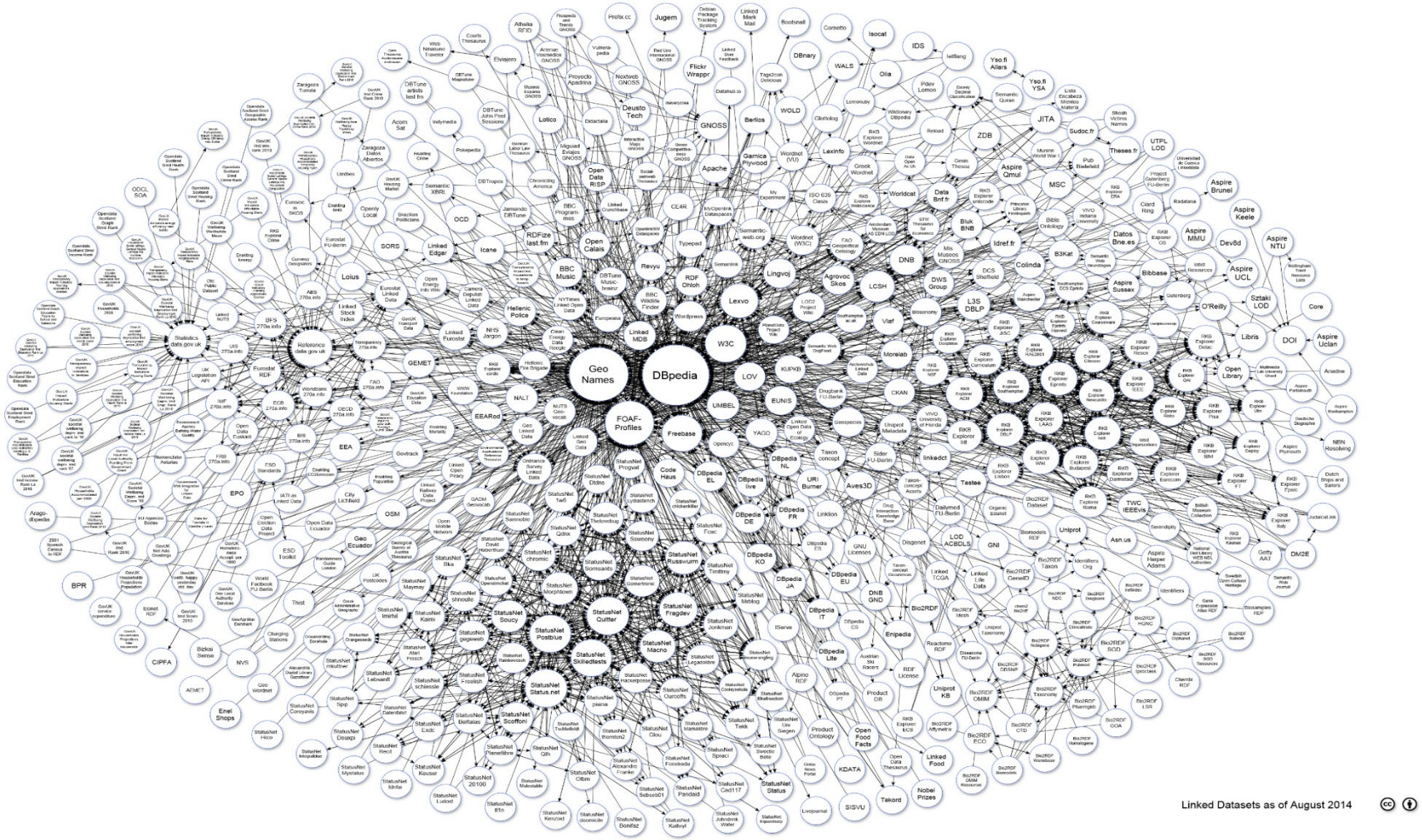


LOD (2011)


DBpedia
Freebase
Geo Names
YAGO
Uberblic
Linked MDB
Event Media
URI Burner
Greek DBpedia
Open Calais
Revyu
BBC Music
BBC Program mes
Last.FM (rdfize)
Music Brainz (DBTune)
Audio Scrobbler (DBTune)
GTAA
LOV
Bricklink
Open Library
Open Library (Talis)
LCSH
LIBRIS
GND
VIAF
ndlna
Europeana
NSZL Catalog
P20
BNB
DDC
Sudoc
OAI
ACM
DBLP (L3S)
DBLP (FU Berlin)
DBLP (RKB Explorer)
CiteSeer
RAE2001
dotAC
IEEE
KISTI
NSF
ePrints
VIVO Indiana
VIVO Cornell
VIVO UF
STITCH
HGNC
PDB
UniProt
Drug Bank
LinkedCT
SIDER
Diseasome
Daily Med
PubMed
Gene Ontology
SGD
GeneID
MGI
InterPro
UniRef
UniParc
UniSTS
OMIM
CHEMBL
SISVU
Affymetrix
ProDom
PROSITE
Pub Chem
KEGG Enzyme
KEGG Pathway
KEGG Compound
KEGG Glycan
KEGG Reaction
Homolo Gene
Chem2 Bio2RDF
OGOLOD
Reactome
Linked Open Numbers
Open Corporates
Sears
Airports
Product DB
Product Types Ontology
Linked Open Colors
GEMET
AGROVOC
EARTh
LODE
Taxonomy
Cornetto
Lexvo
lingvoj
Enipedia
riese
Open Cyc
UMBEL
Geo Linked Data
Geo Species
Fishes of Texas
Taxon Concept
LOIUS
data.gov.uk intervals
TWC LOGD
Eurostat
US SEC (rdfabout)
US Census (rdfabout)
EUNIS
Twarql
Linked GeoData
Linked Sensor Data (Kno.e.sis)
GovTrack
Semantic XBRL
GeoWord Net
El Viajero Tourism
Ocean Drilling Codices
AEMET
Metoffice Weather Forecasts
Yahoo! Geo Planet
Janus AMP
EURES
FTS
Data Gov.ie
CORDIS (FUB)
ISTAT Immigration
Brazilian Politicians
GovWILD
legislation data.gov.uk
UK Postcodes
EEA
Ordnance Survey
Openly Local
OpenEI
London Gazette
Crime Reports UK
Hellenic FBD
DB Tropes
Flickr wrappr
Source Code Ecosystem Linked Data
Ontos News Portal
Semantic Tweet
RDF Book Mashup
my Experiment
Calames
OS
ECS Southampton
ERA
Pfam
Wiki
STW
GESIS
RESEX
Scholarometer
IBM
DEPLOY
LOCAH
NVD
IRIT
Newcastle
Roma
Courseware
RISKS
As of September 2011

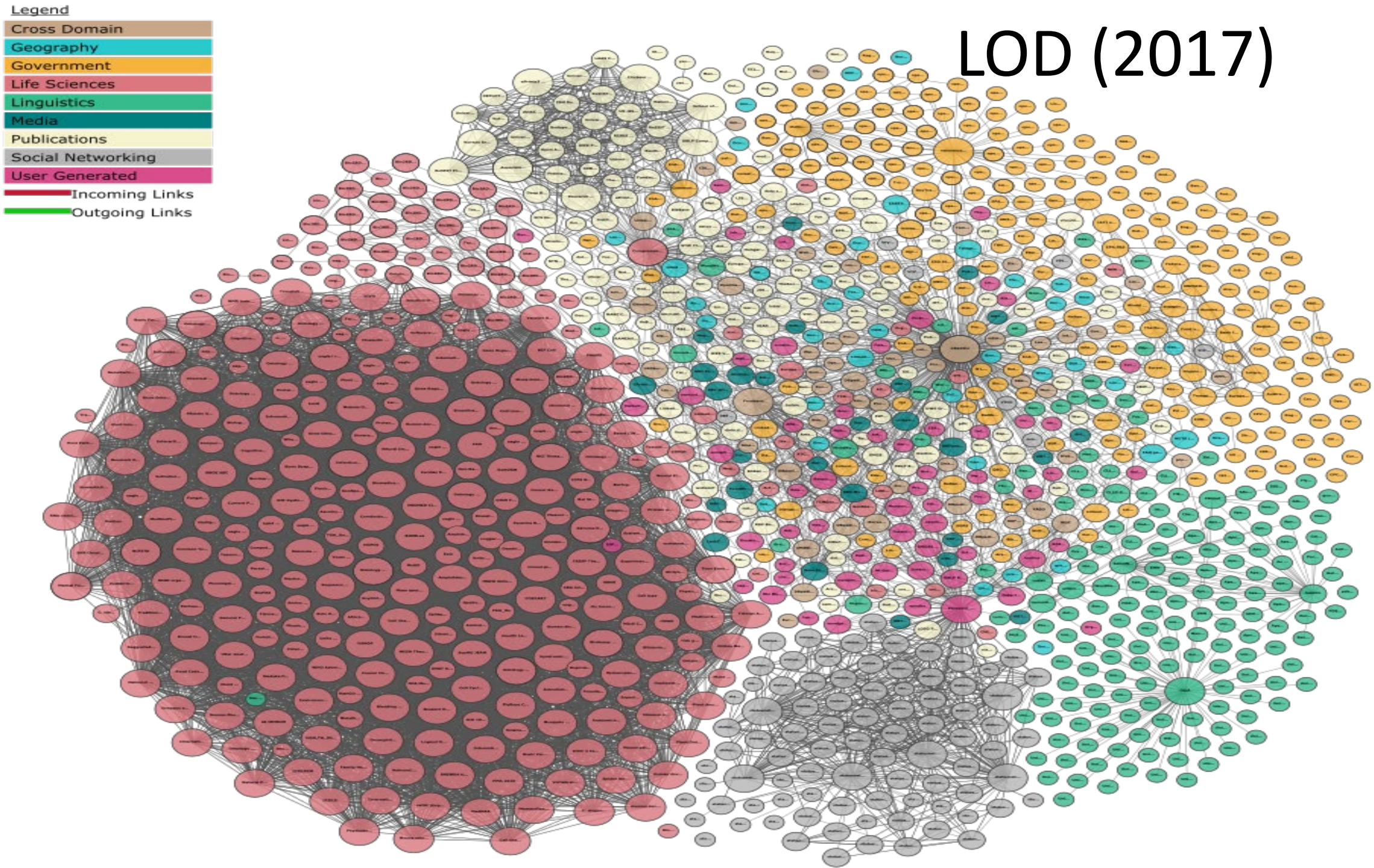
LOD (2014)

WESO

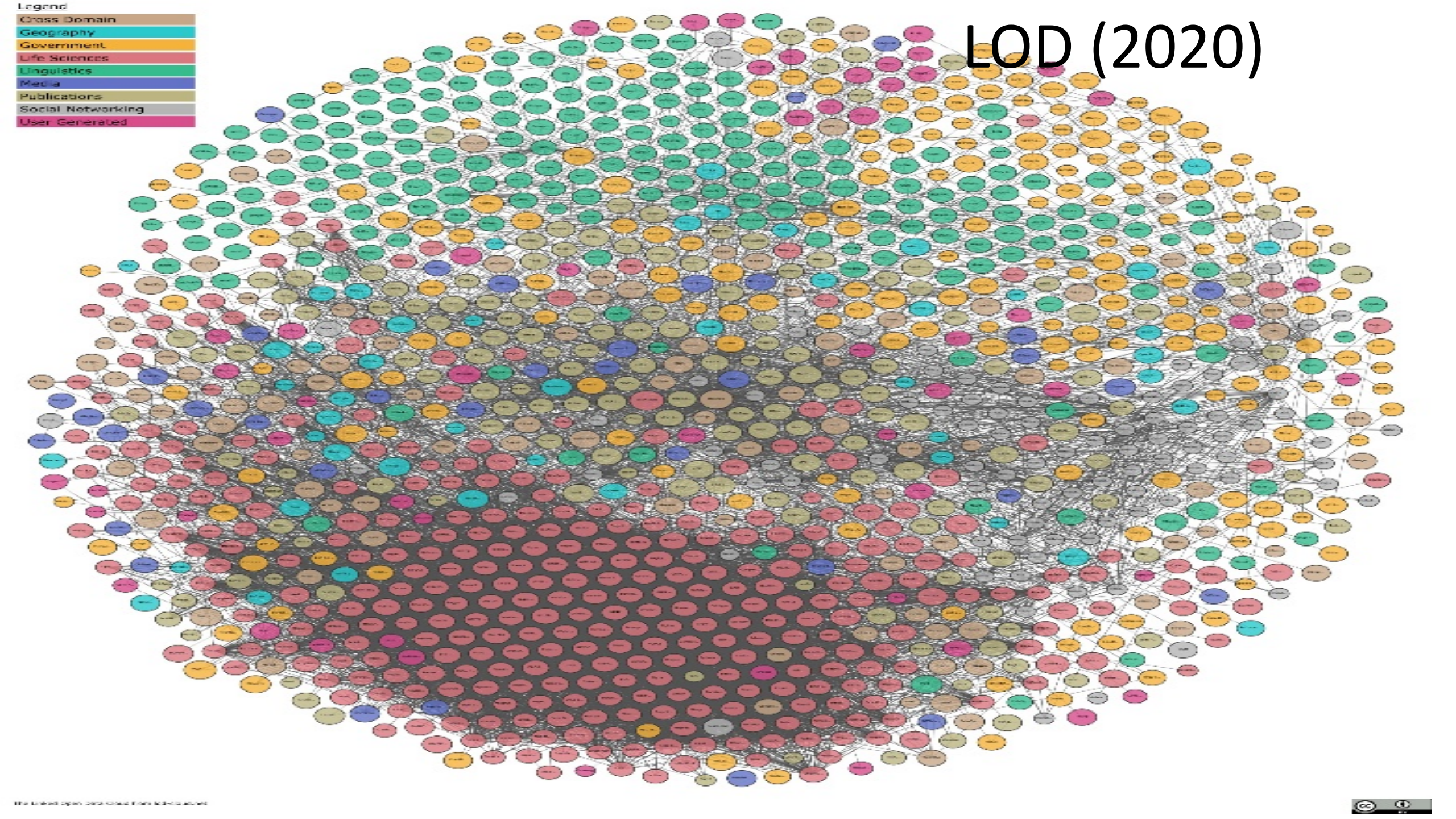


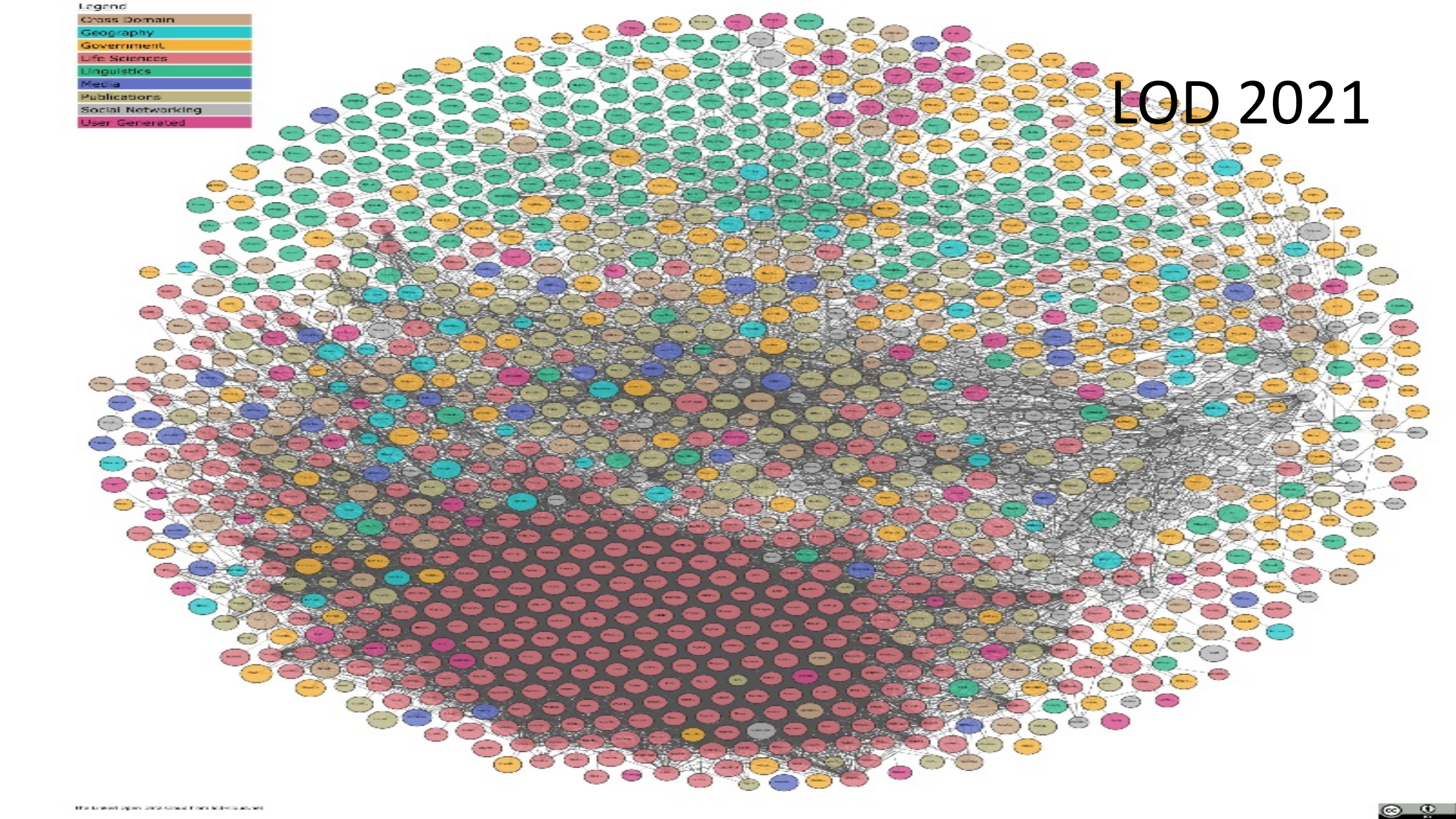
LOD (2017)

WESO



LOD (2020)


Legend
Cross Domain
Geography
Government
Life Sciences
Linguistics
Media
Publications
Social Networking
User Generated


Legend
Cross Domain
Geography
Government
Life Sciences
Linguistics
Media
Publications
Social Networking
User Generated
LOD 2021

Knowledge Graphs

Knowledge graph = a **graph of data** intended to represent knowledge about some domain

- Nodes represent entities of interest
- Edges represent relations between these entities

Popularized by Google in 2012

Different types of knowledge graphs models

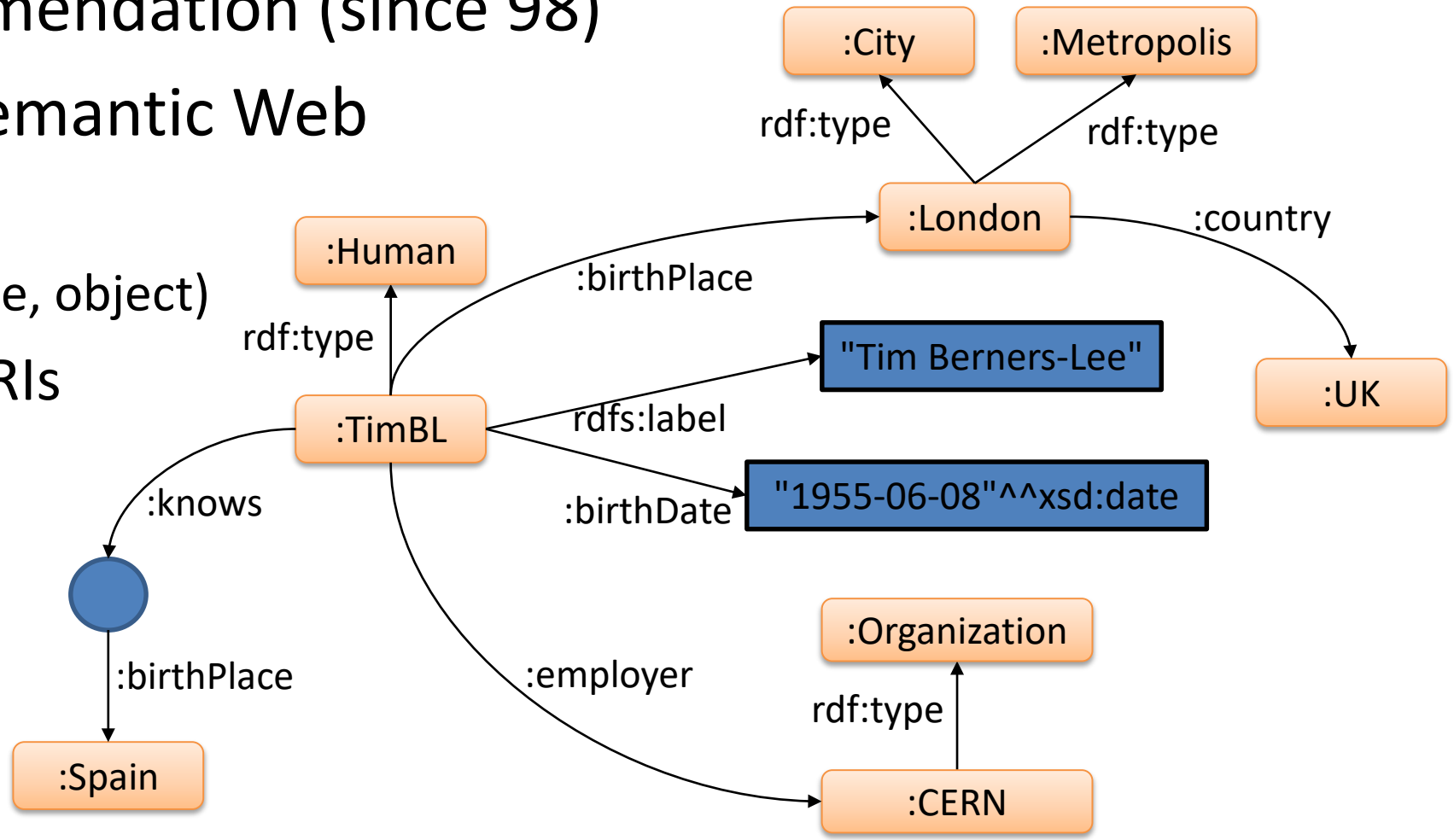
- RDF based
- Property graphs
- Wikibase graphs

RDF graphs

RDF = W3C recommendation (since 98)

Lingua franca of Semantic Web

- Based on triples
  - (subject, predicaje, object)
- Most nodes are URIs
  - Interoperability



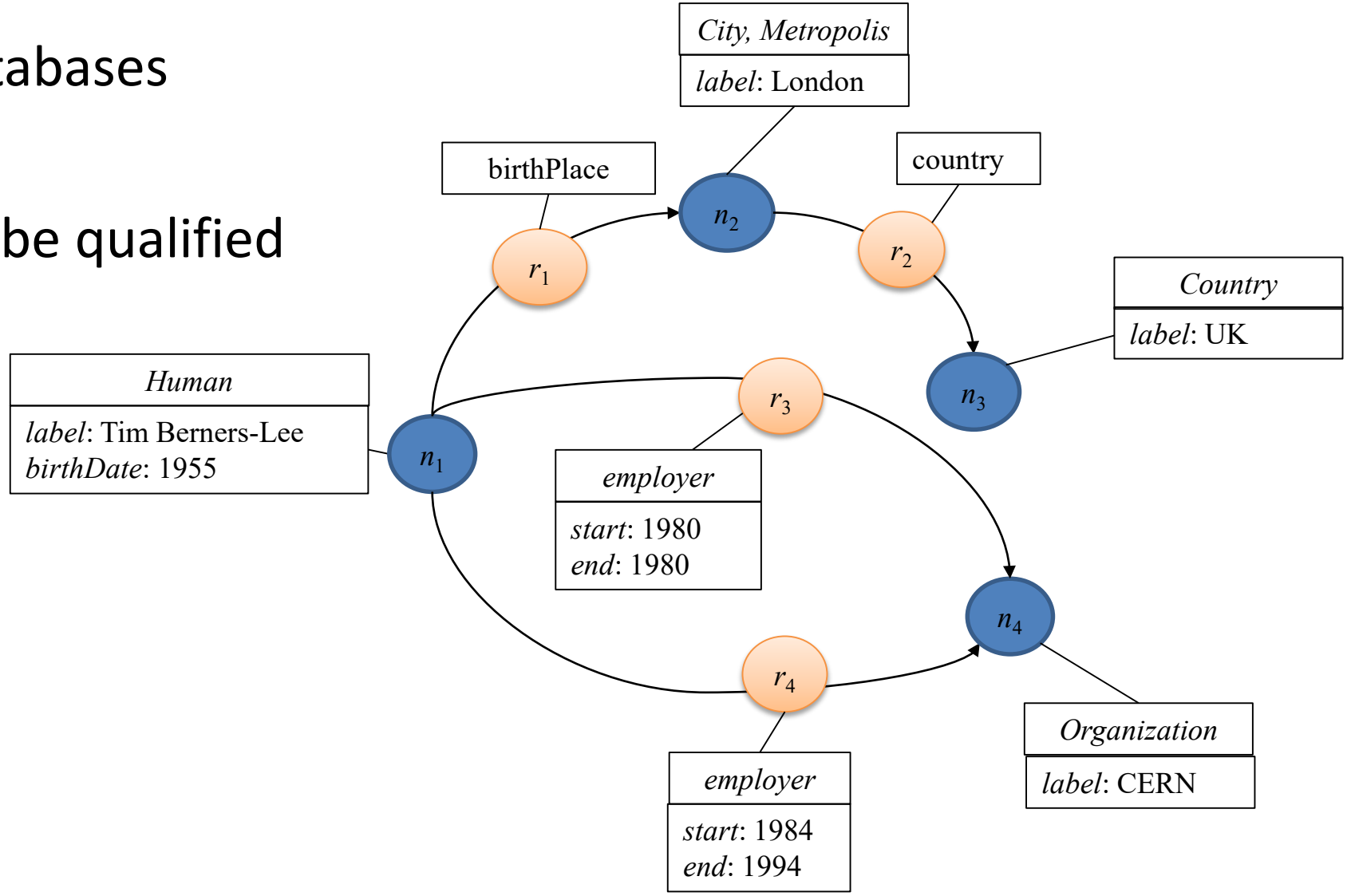
Property graphs

Popularized by graph databases

Example: Neo4J

Nodes and relations can be qualified

Nodes can have types



Wikibase graphs

Popularized by Wikidata

Wikibase = software supporting Wikidata

The values can be nodes in the graph

Example:

Tim Berners Lee

http://www.wikidata.org/entity/Q80