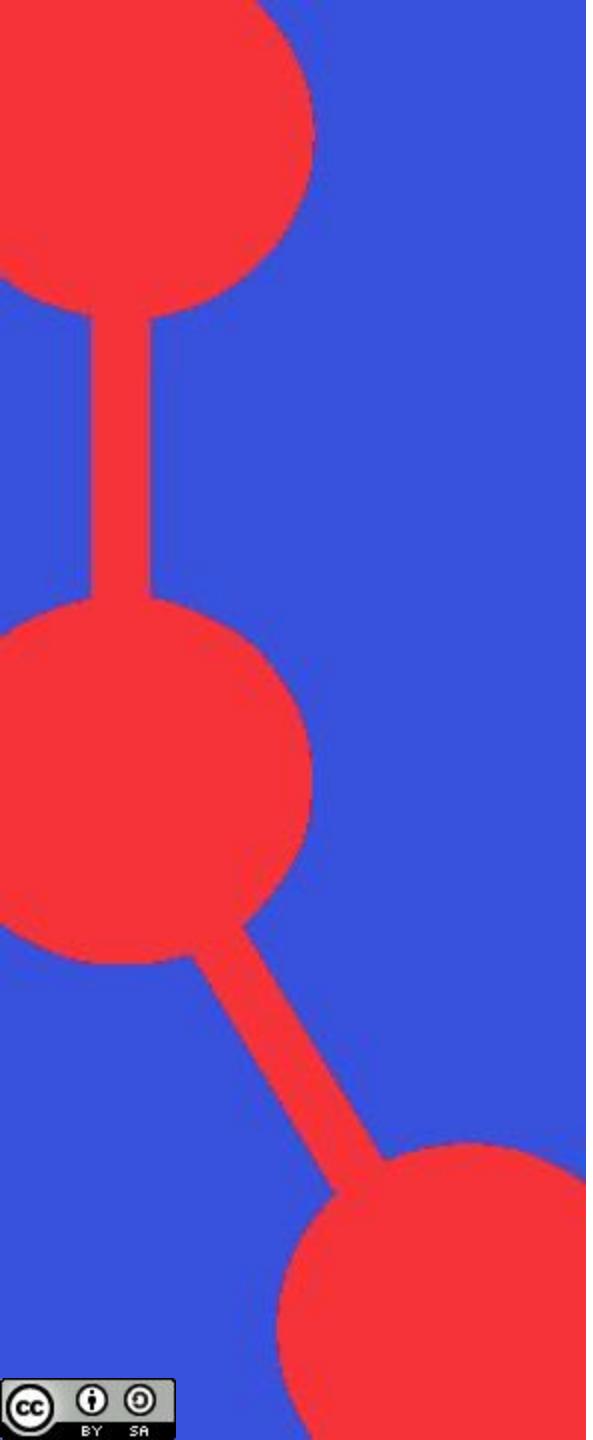


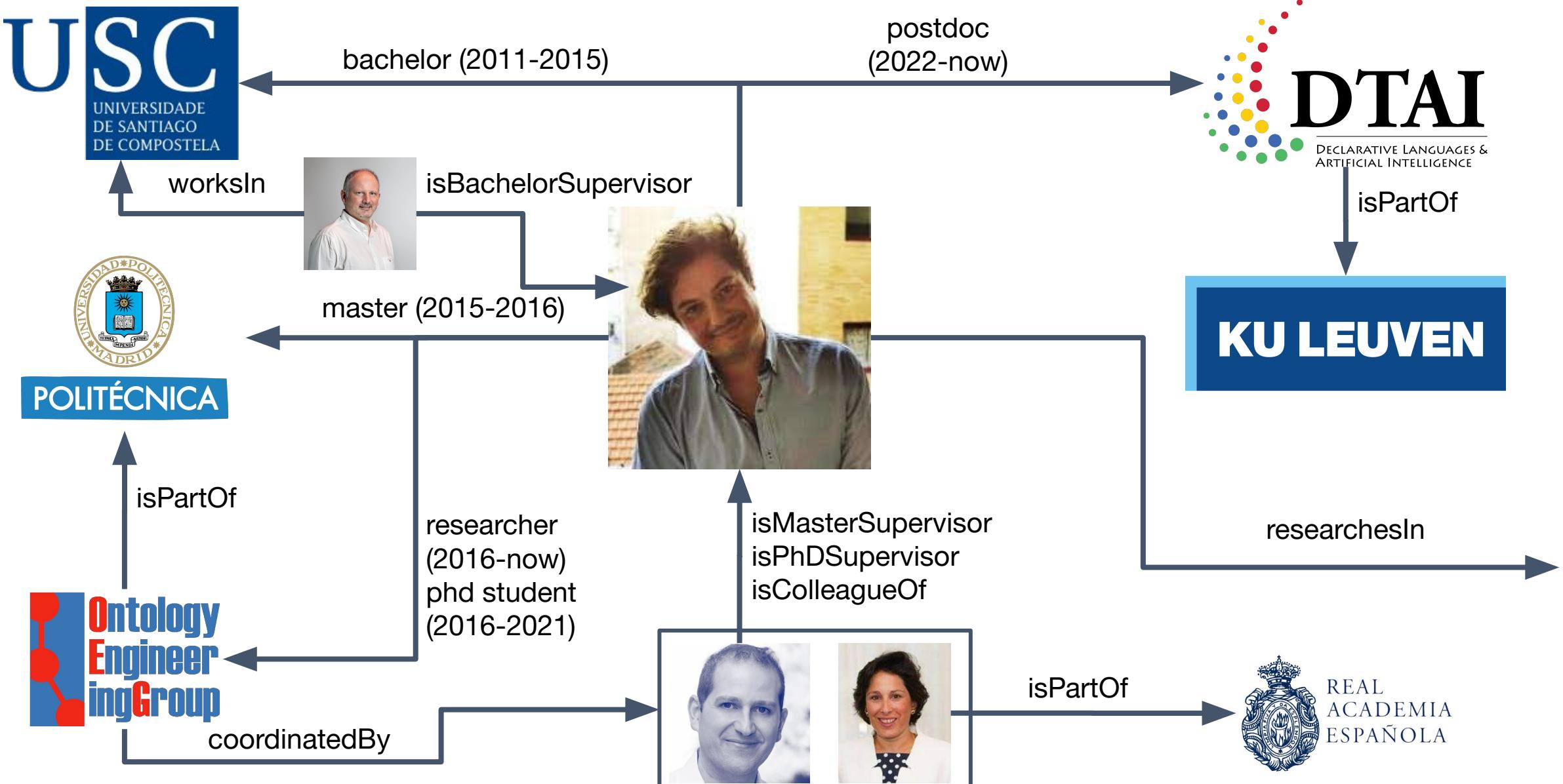


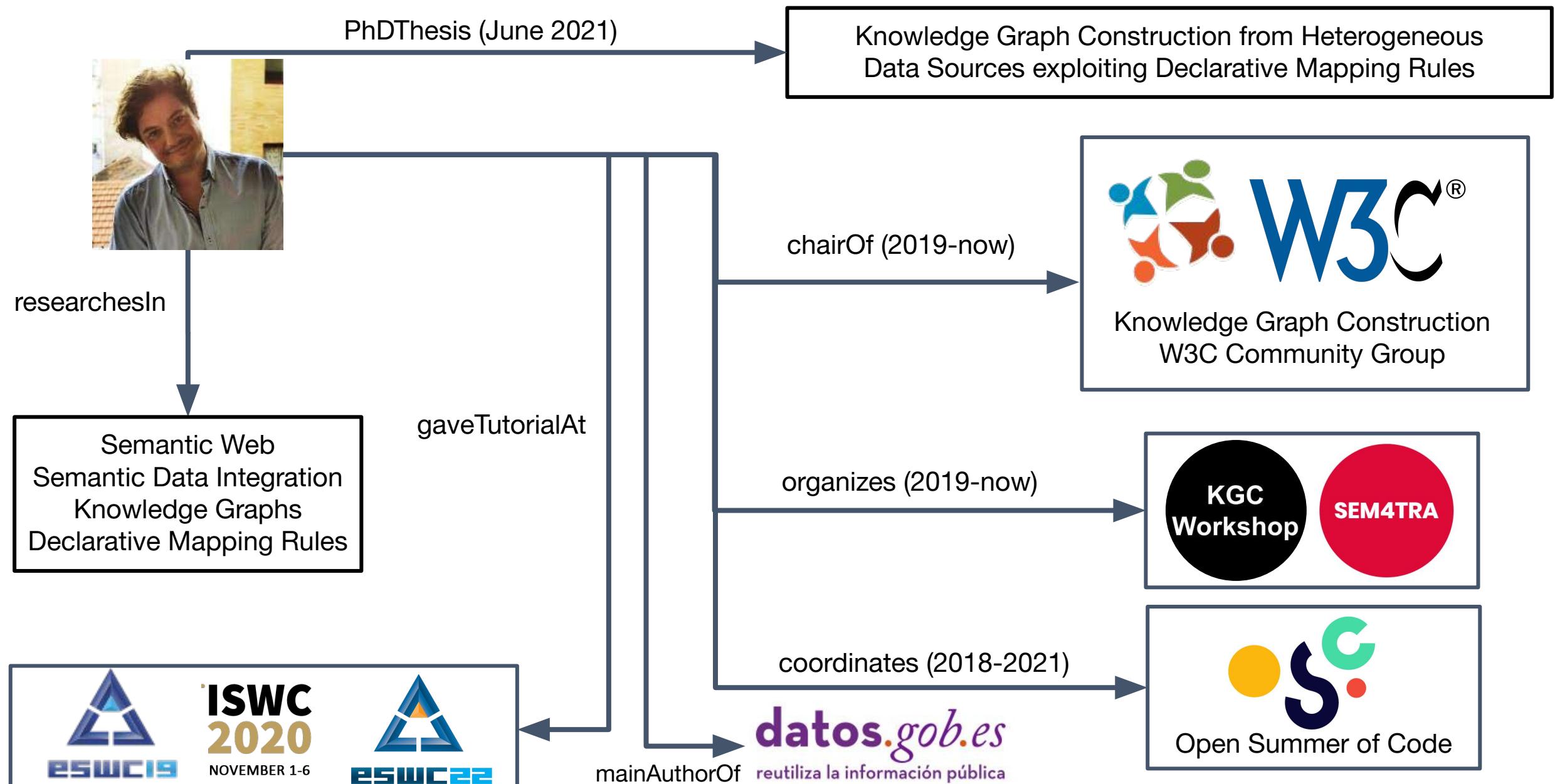
# Knowledge Graphs and Ontologies

## A Practical Introduction to the Semantic Web

David Chaves-Fraga, Ontology Engineering Group  
Universidad Politécnica de Madrid, Spain





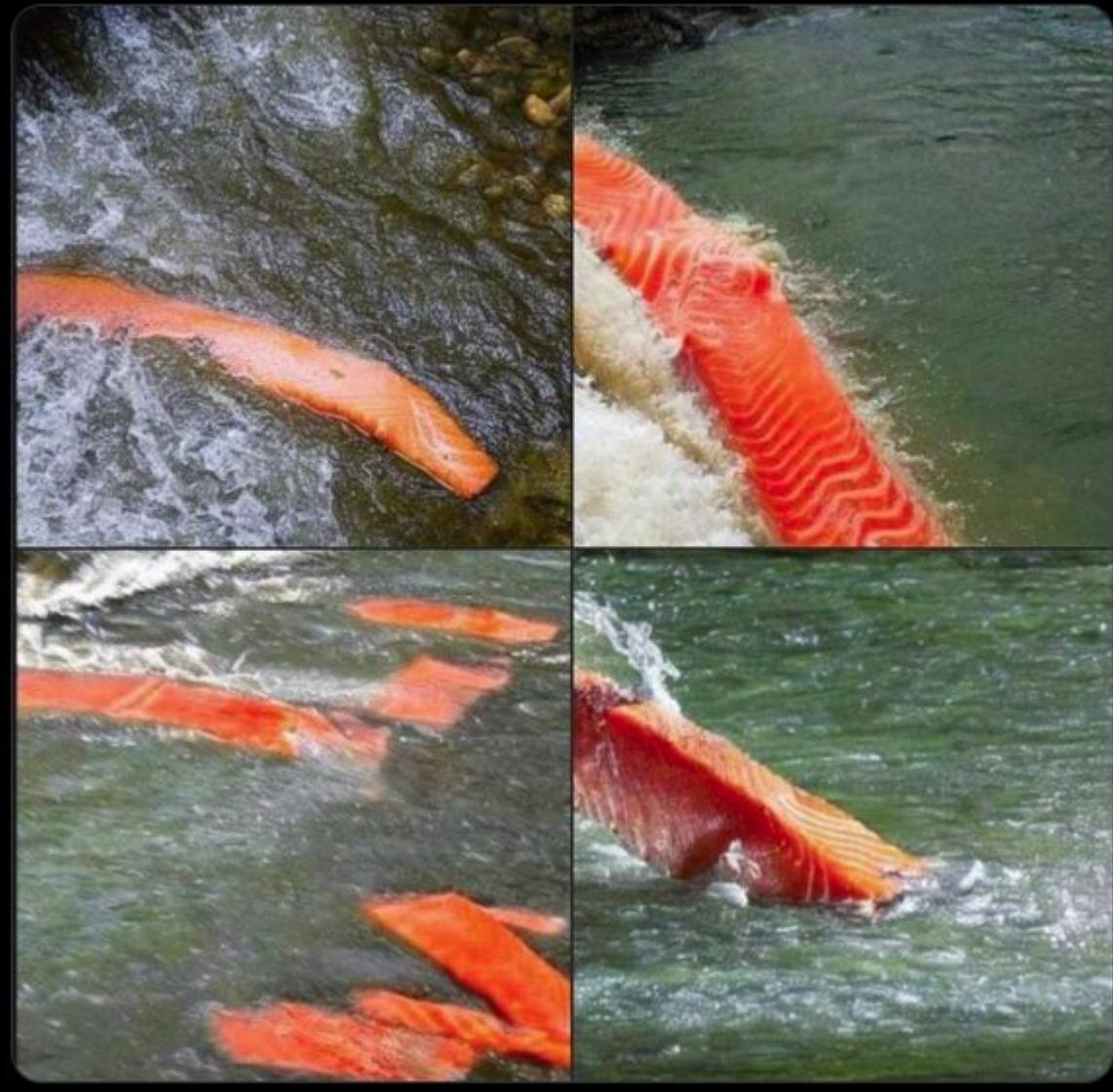




# The problem of the salmon in a river

dying at these AI images of “salmon in a river”

Traducir Tweet



<http://dbpedia.org/resource/Salmon>

[http://dbpedia.org/resource/Salmon as food](http://dbpedia.org/resource/Salmon_as_food)



- This work is licensed under the license  
CC BY-NC-SA 4.0 International
  - <http://purl.org/NET/rdflicense/cc-by-nc-sa4.0>



- You are free:
  - to Share — to copy, distribute and transmit the work
  - to Remix — to adapt the work
- Under the following conditions
  - Non-commercial — You cannot use it for commercial purposes, nor for training inside a commercial company
  - Attribution — You must attribute the work by inserting
    - a credits slide stating: These slides are partially based on “Knowledge Graphs and Ontologies: A Practical Introduction to the SW” by David Chaves-Fraga
  - Share-alike



These slides are partially based on:

- “Semantic Web, Linked Data and Knowledge Graphs” course by Oscar Corcho, Raúl García-Castro, Daniel Garijo, David Chaves-Fraga, Paola Espinoza (Universidad Politécnica de Madrid)
- “Data and Knowledge Graphs” course Anastasia Dimou, David Chaves-Fraga (KULeuven)



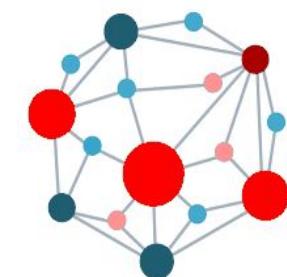
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) Hands-On 1: Querying Wikidata



## Hands-On 2: Constructing and validating your RDF graph (50min)

- 1) Tools for Developing Ontologies and KGs
- 2) RML (from CSV/XML/JSON to RDF)
- 3) SHACL (RDF validation)



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project





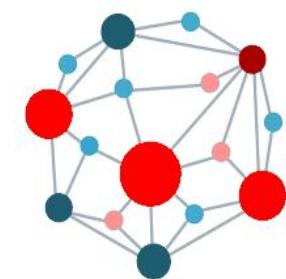
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) Hands-On 1: Querying Wikidata



## Hands-On 2: Constructing and validating your RDF graph (50min)

- 1) Tools for Developing Ontologies and KGs
- 2) RML (from CSV/XML/JSON to RDF)
- 3) SHACL (RDF validation)



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project



- When do we start talking about Knowledge Graphs?
  - Google's announcement of the Google Knowledge Graph in 2012
    - <https://www.blog.google/products/search/introducing-knowledge-graph-things-not/>
    - "Things, not strings"
  - Followed by Airbnb, Amazon, eBay, Facebook, IBM, LinkedIn, Microsoft, Uber, etc.



Marie Curie  
Física

Resumen   Premios   Vídeos



Más imágenes

[https://es.wikipedia.org/wiki/Marie\\_Curie](https://es.wikipedia.org/wiki/Marie_Curie)

**Marie Curie - Wikipedia, la enciclopedia libre**

Maria Salomea Skłodowska-Curie, más conocida como Marie Curie o Madame Curie (Varsovia, 7 de noviembre de 1867-Passy, 4 de julio de 1934), ...

Causa de muerte: [Anemia aplásica](#)      Padres: [Władysław Skłodowski](#); [Bronis...](#)

Nombre de nacimiento: Maria Salomea ...      Conocida por: descubrimiento del radi...

[Pierre Curie](#) · [Ève Curie](#) · [Instituto Curie](#) · [Museo Curie](#)

Otras preguntas de los usuarios

### Información

Maria Salomea Skłodowska-Curie, más conocida como Marie Curie o Madame Curie, fue una física y química polaca nacionalizada francesa. Pionera en el campo de la radioactividad, es la primera y única persona en recibir dos premios Nobel en distintas especialidades científicas: Física y Química.

[Wikipedia](#)

**Nacimiento:** 7 de noviembre de 1867, [Varsovia](#), [Polonia](#)

**Fallecimiento:** 4 de julio de 1934, [Passy](#), [Francia](#)

**Descubrimientos:** [Radio](#), [Polonio](#)

**Premios:** [Premio Nobel de Física](#), [Premio Nobel de Química](#), [MÁS](#)

**Hijas:** [Irène Joliot-Curie](#), [Ève Curie](#)

**Cónyuge:** [Pierre Curie](#) (m. 1895–1906)

**Nietos:** [Hélène Langevin-Joliot](#), [Pierre Joliot](#)

facebook

Google

Linkedin

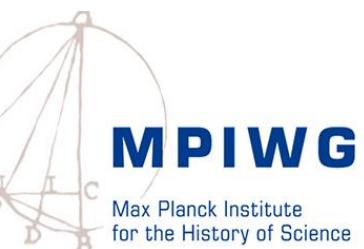
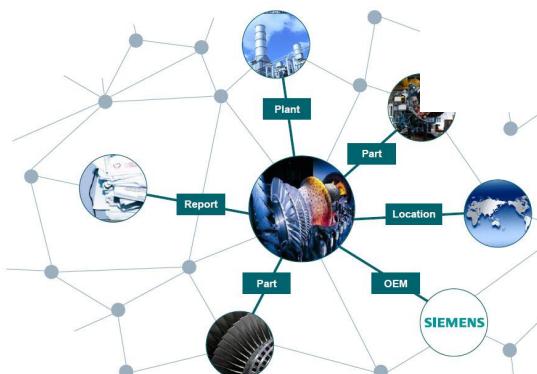
ontotext



#### Why (Knowledge) Graphs?

Benefits of using knowledge graphs for data representation

- The world is entities and relations!
- Intelligent domain model instead of complex (physical) data model
- Schema-on-read instead of complex schema migration for extensions
- Easy integration of multiple data sources (schemas) and types (structured, unstructured, ...)
- Formal semantic representation enables inference and machine processing



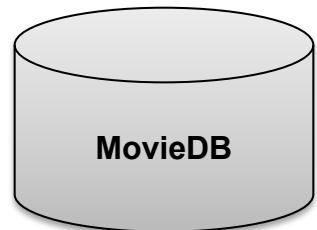
THE  
BRITISH  
MUSEUM



eBay

Uber

Adapted from Mariano Rodriguez Muro from KGB2019 Keynote ESWC2019



The screenshot shows a movie page for 'Even the Rain' (2010) on the Internet Movie Database. The page includes a movie poster, basic information like title, year, and rating (7.3/10), and a detailed plot summary. A large blue callout box overlaid on the page contains the text:

**Data exposed to the Web  
via HTML, pdf, etc.**

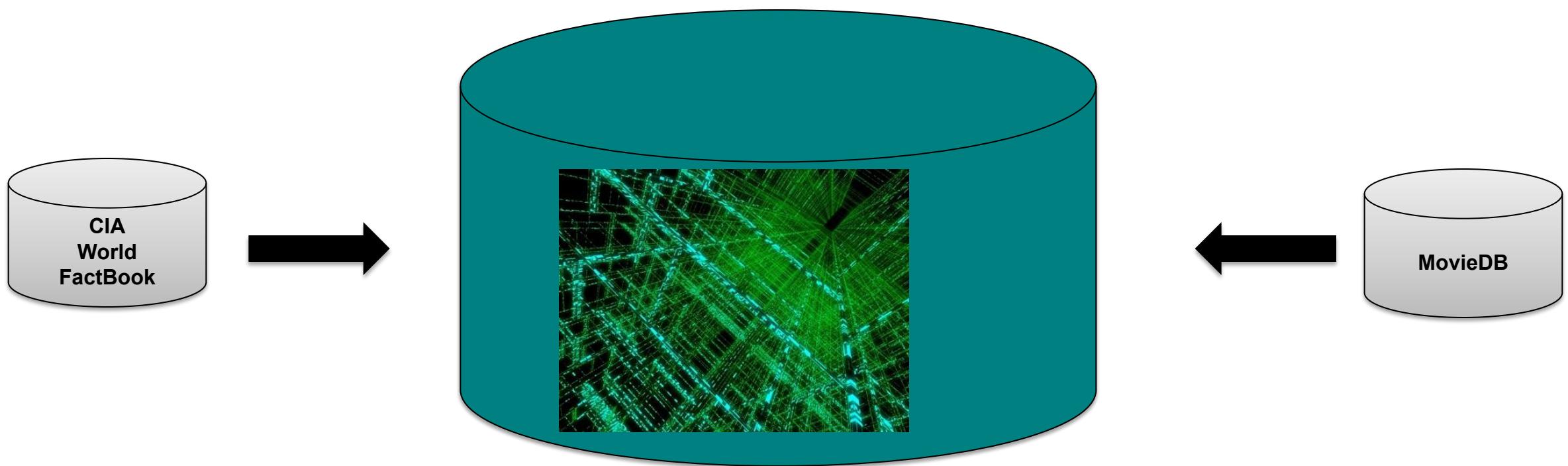


The screenshot shows a page from the CIA's World Factbook about Bolivia. It features a map of Bolivia and surrounding countries, a flag, and some text. A large blue callout box overlaid on the page contains the text:

**Data exposed to the Web  
via HTML, pdf, etc.**



- Use the Web like a single **global database**
  - Move from a Web of documents to a **Web of Data**

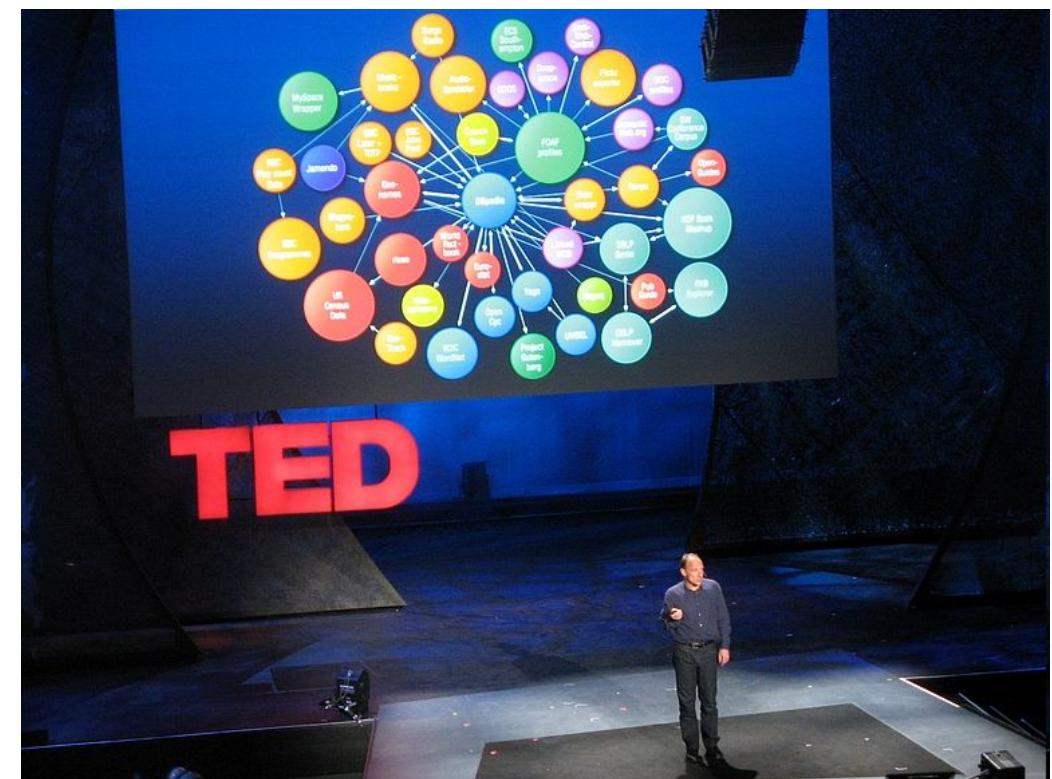




# Linked Data: Four principles (Tim Berners Lee, 2006)

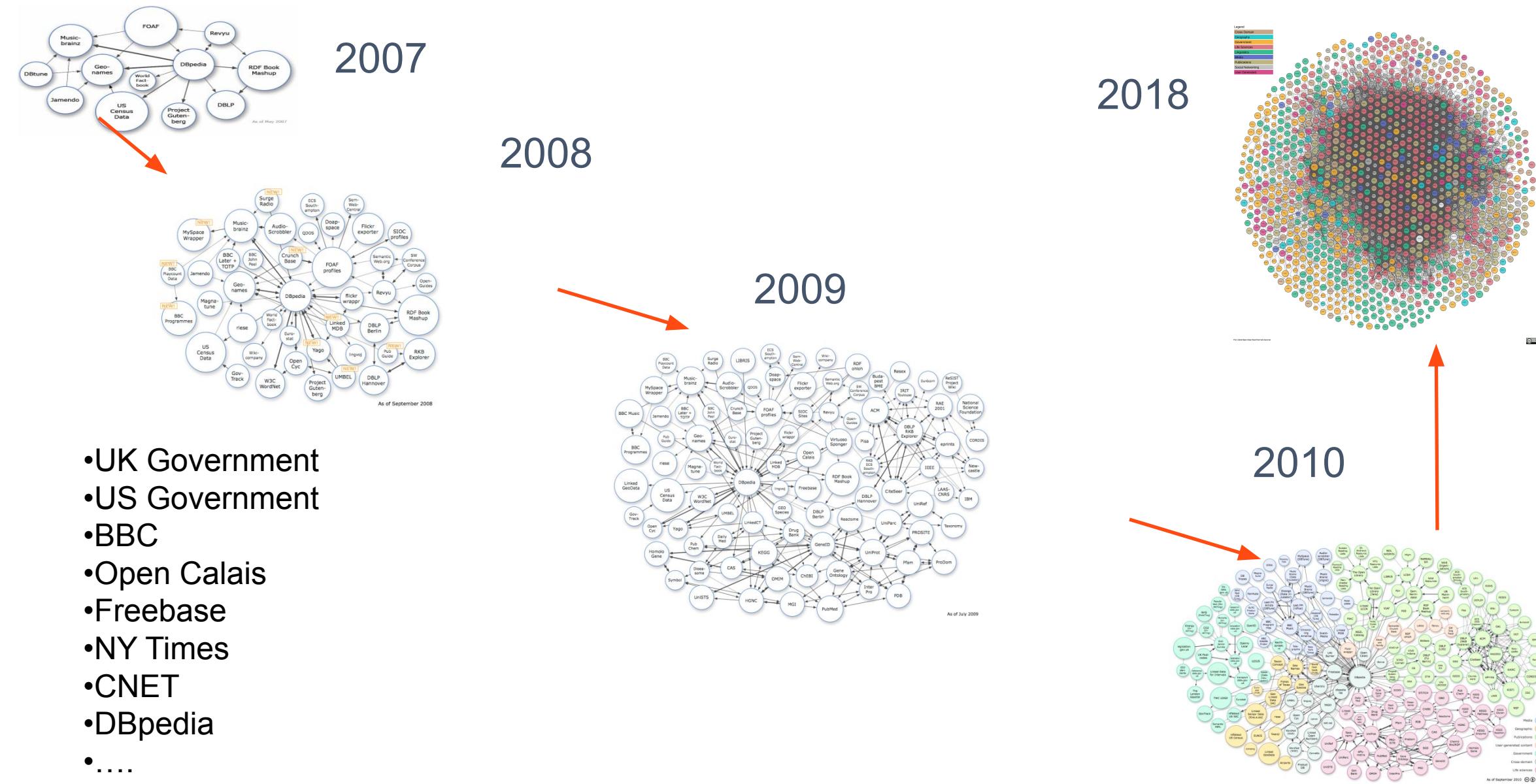
- Use URIs as names for things
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information using standards (RDF\*, SPARQL)
- Include links to other URIs, so that they can discover more things.

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



[https://youtu.be/OM6XIIcm\\_qo](https://youtu.be/OM6XIIcm_qo)

# Linked Open Data evolution



- UK Government
- US Government
- BBC
- Open Calais
- Freebase
- NY Times
- CNET
- DBpedia
- ....

“The Semantic Web is an **extension** of the current Web in which information is given **well-defined meaning**, better enabling computers and people to work in **cooperation**. It is based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications.”

Hendler, J., Berners-Lee, T., and Miller, E.  
Integrating Applications on the Semantic Web, 2002,  
<http://www.w3.org/2002/07/swint.html>

<https://developers.google.com/knowledge-graph/>

<https://www.youtube.com/watch?v=mmQI6VGvX-c>

The screenshot shows the official Google Knowledge Graph Search API documentation. At the top, there's a navigation bar with the Google logo, a search bar, language selection, and product links. Below the header, a blue navigation bar has 'GUÍAS' selected. The main content area features a sidebar with 'Get Started' sections like 'Introduction', 'Prerequisites', and 'How To...' (including 'Install Client Libraries', 'Authorize Requests', 'Use the Knowledge Graph Search Widget', and 'Terms of Service'). The main content area has a title 'Google Knowledge Graph Search API' with a five-star rating. It describes the API as finding entities in the Google Knowledge Graph using schema.org types and JSON-LD specification. A 'Typical use cases' section lists tasks such as getting ranked lists, completing entities, and annotating content. A note states the API is read-only. A 'Sample request' section provides an example of what you can send to the API.

Google Search > Knowledge Graph Search API

SEARCH Buscar LANGUAGE TODOS LOS PRODUCTOS :

GUÍAS REFERENCIA

## Google Knowledge Graph Search API

Get Started

Introduction

Prerequisites

How To...

Install Client Libraries

Authorize Requests

Use the Knowledge Graph Search Widget

Terms of Service

Contenido

Typical use cases

Sample request

Knowledge Graph entities

The Knowledge Graph Search API lets you find entities in the [Google Knowledge Graph](#). The API uses standard [schema.org](#) types and is compliant with the [JSON-LD](#) specification.

### Typical use cases

Some examples of how you can use the Knowledge Graph Search API include:

- Getting a ranked list of the most notable entities that match certain criteria.
- Predictively completing entities in a search box.
- Annotating/organizing content using the Knowledge Graph entities.

**Note:** The Knowledge Graph Search API is a read-only API.

For detailed information about the API methods and parameters, see the [API Reference](#).

### Sample request

The following example shows one kind of request you can send to the API. (But check the [Prerequisites](#) section first. You'll also need to insert your own API key.)



# General-purpose open and domain-specific KGs

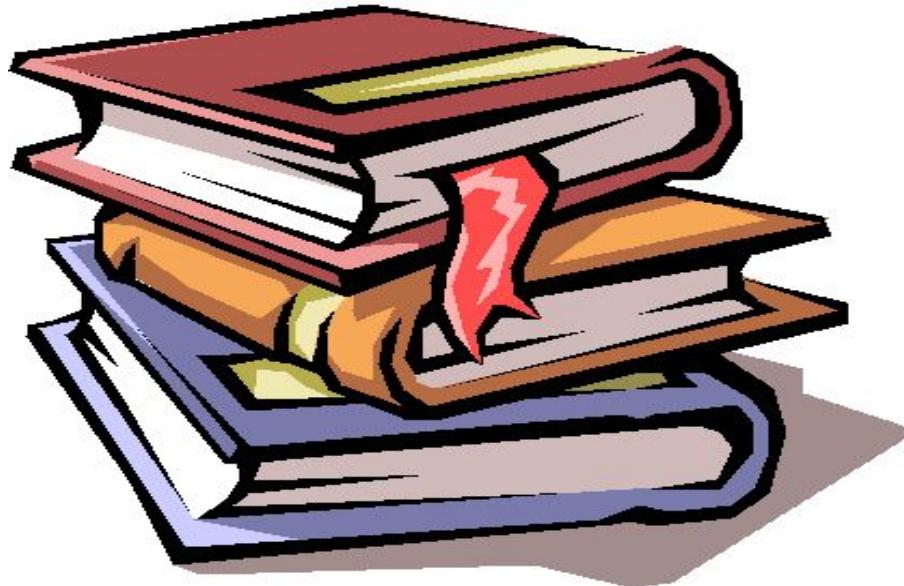
- Some of the well-known ones:
  - DBpedia and YAGO
  - Wikidata (and the earlier Freebase)
  - BabelNet
  - OpenCyc
- And many domain-specific ones
  - Media
  - Government
  - Publications
  - Geography
  - Life Sciences
  - Cultural heritage
  - Law



<https://www.youtube.com/watch?v=TJfrNo3Z-DU>



# Recent references about Knowledge Graphs



- Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutiérrez C, Labra Gayo JE, Kirrane S, Neumaier S, Polleres A, Navigli R, Ngonga Ngomo AC, Rashid SM, Rula A, Schmelzeisen L, Sequeda J, Staab S, Zimmermann A (2020) **Knowledge Graphs**. <https://arxiv.org/abs/2003.02320>
- Ji S, Pan S, Cambria E, Marttinen P, Yu PS (2020) **A Survey on Knowledge Graphs: Representation, Acquisition and Applications** <https://arxiv.org/abs/2002.00388>
- Gutiérrez C, Sequeda J (2021) **Knowledge Graphs**. Communications of the ACM 64(3):96-104. <https://cacm.acm.org/magazines/2021/3/250711-knowledge-graphs/fulltext>
- Guilin Qi, Huajun Chen, Kang Liu, Haofen Wang, Qiu Ji, and Tianxing Wu (2021) **Knowledge Graph**. Springer. *To appear*

(enterprise) **Knowledge Graphs**



Extracted from Mariano Rodriguez Muro from KGB2019 Keynote ESWC2019



Knowledge is more important than data

Symbolic AI still has something to say

See the Web as a big database with structured resources

Semantic Web technologies are not (so) difficult to learn

How to create, validate, manage and query RDF KGs

Successful and real use-cases with real companies

Benefits of structure your knowledge and data as RDF

How KGs help AI

Fork the repository (and the clone it)

<https://github.com/dachafra/kg-tutorial>

Good to have installed in your local machine

Git, Python



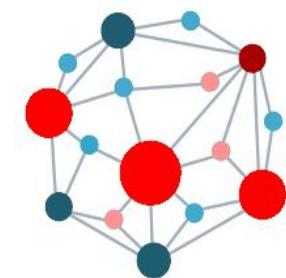
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) **RDF, OWL and SPARQL**
- 3) **RDF APIs and Triplestores**
- 4) Hands-On 1: Querying Wikidata



## Hands-On 2: Constructing and validating your RDF graph (50min)

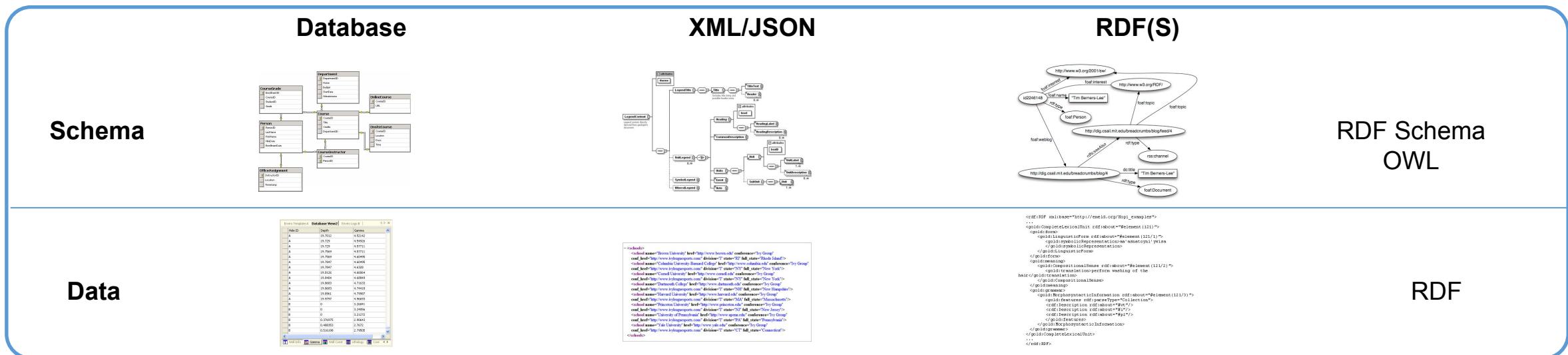
- 1) Tools for Developing Ontologies and KGs
- 2) RML (from CSV/XML/JSON to RDF)
- 3) SHACL (RDF validation)



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project



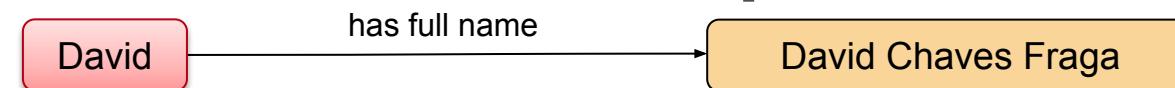


- W3C Recommendation
    - Model
    - Syntax
    - Semantics

- Also known as triples
  - [Subject, Predicate, Object]
- “Raúl is a member of the Ontology Engineering Group”
  - [David, is member of, Ontology Engineering Group]



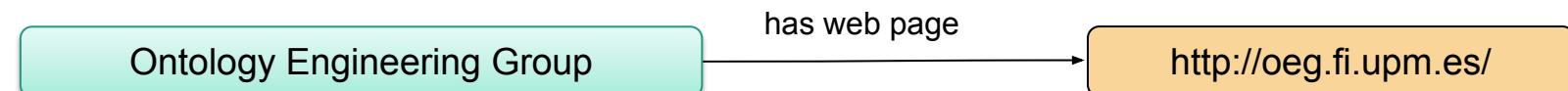
- “Raúl’s full name is Raúl García Castro”
  - [David, has full name, Raúl García Castro]



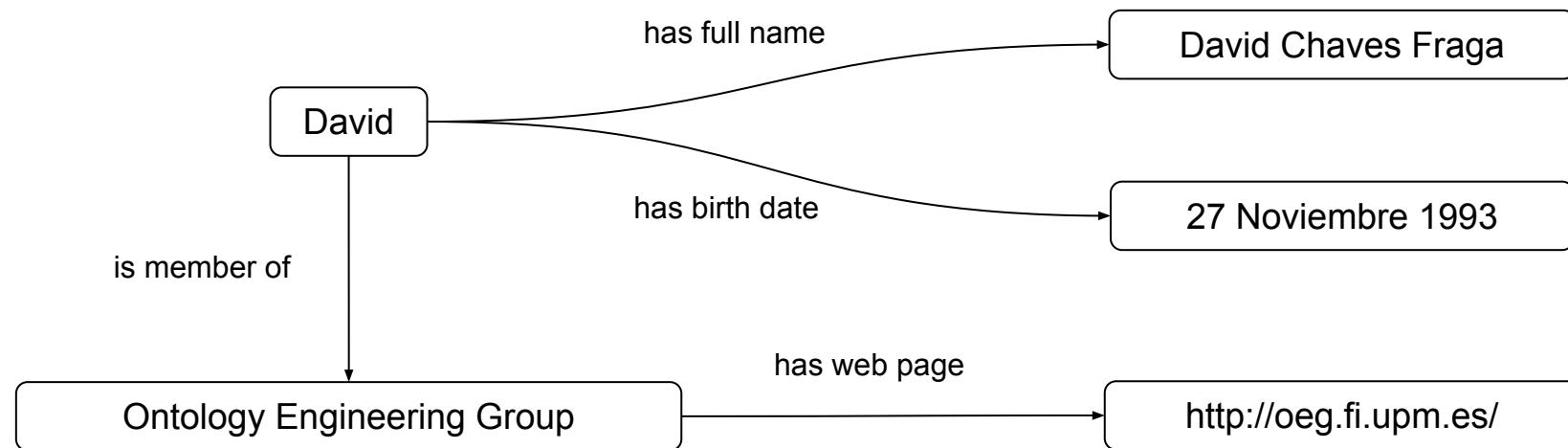
- “Raúl was born on December 26th 1975”
  - [David, was born, 26 December 1975]



- “The homepage of the Ontology Engineering Group is <http://www.oeg-upm.net/>”
  - [Ontology Engineering Group, has web page, <http://oeg.fi.upm.es/>]

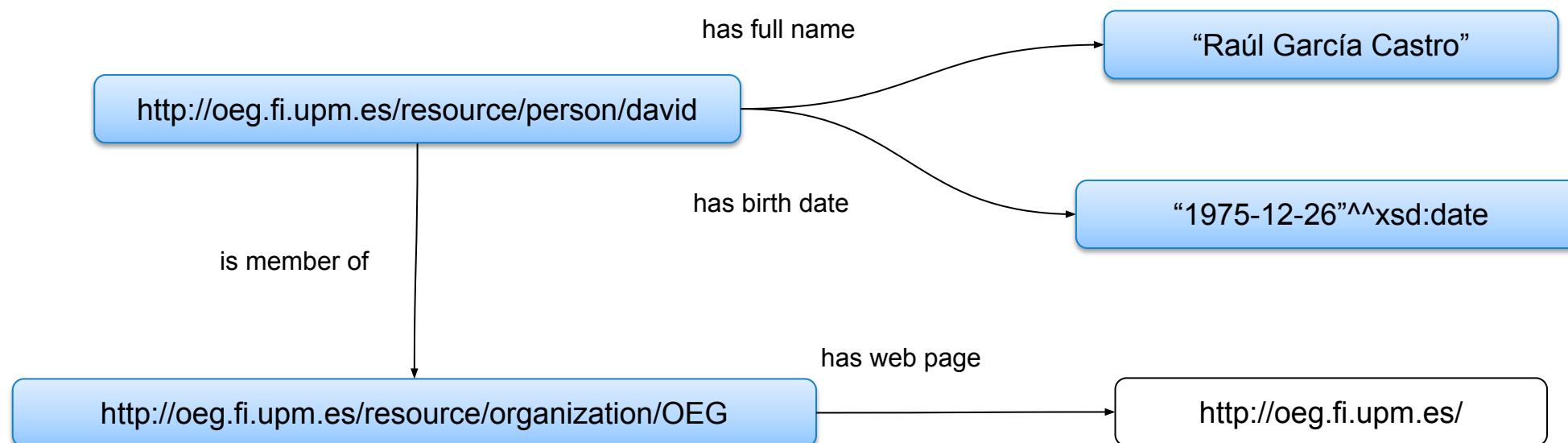


- RDF graphs are sets of triples



- A set of RDF graphs is an RDF Dataset (in RDF1.1)
  - There is a default graph
  - And zero or more named graphs

- Triple objects may be resources or literals
  - **Subjects and predicates are always resources**
- Literals may have an XML Schema datatype associated to them
  - Default: xsd:string
  - RDF1.1 defines a couple of extra ones
    - rdf:langString (e.g., “Spain”@en)
    - rdf:HTML and rdf:XMLLiteral





A list of the RDF-compatible XSD types, with short descriptions"

	Datatype	Value space (informative)
Core types	<a href="#">xsd:string</a>	Character strings (but not all Unicode character strings)
	<a href="#">xsd:boolean</a>	true, false
	<a href="#">xsd:decimal</a>	Arbitrary-precision decimal numbers
	<a href="#">xsd:integer</a>	Arbitrary-size integer numbers
IEEE floating-point numbers	<a href="#">xsd:double</a>	64-bit floating point numbers incl. ±Inf, ±0, NaN
	<a href="#">xsd:float</a>	32-bit floating point numbers incl. ±Inf, ±0, NaN
Time and date	<a href="#">xsd:date</a>	Dates (yyyy-mm-dd) with or without timezone
	<a href="#">xsd:time</a>	Times (hh:mm:ss.sss...) with or without timezone
	<a href="#">xsd:dateTime</a>	Date and time with or without timezone
	<a href="#">xsd:dateTimeStamp</a>	Date and time with required timezone
Recurring and partial dates	<a href="#">xsd:qYear</a>	Gregorian calendar year
	<a href="#">xsd:qMonth</a>	Gregorian calendar month
	<a href="#">xsd:qDay</a>	Gregorian calendar day of the month
	<a href="#">xsd:qYearMonth</a>	Gregorian calendar year and month
	<a href="#">xsd:qMonthDay</a>	Gregorian calendar month and day
	<a href="#">xsd:duration</a>	Duration of time
	<a href="#">xsd:yearMonthDuration</a>	Duration of time (months and years only)
	<a href="#">xsd:dayTimeDuration</a>	Duration of time (days, hours, minutes, seconds only)

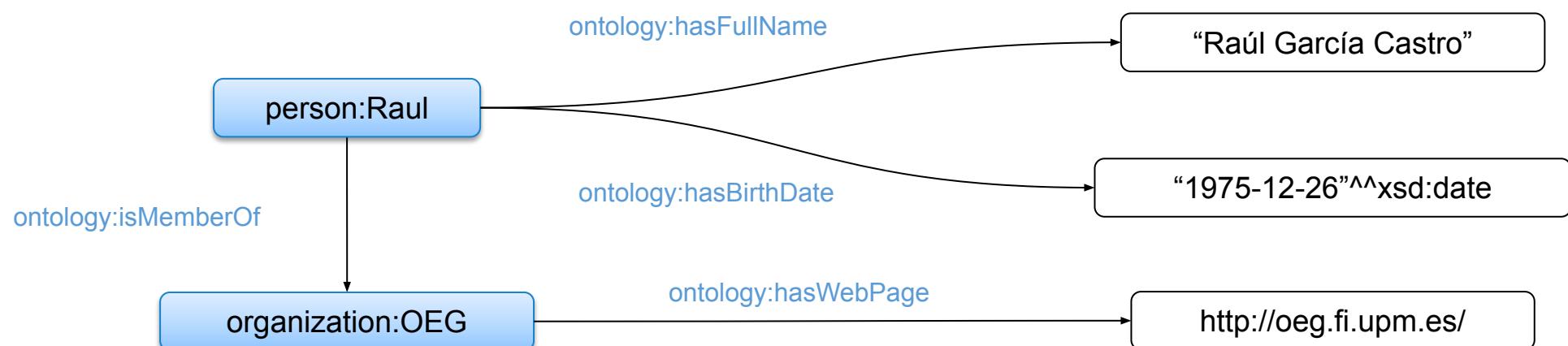


<b>Limited-range integer numbers</b>	<a href="#">xsd:byte</a>	-128...+127 (8 bit)
	<a href="#">xsd:short</a>	-32768...+32767 (16 bit)
	<a href="#">xsd:int</a>	-2147483648...+2147483647 (32 bit)
	<a href="#">xsd:long</a>	-9223372036854775808...+9223372036854775807 (64 bit)
	<a href="#">xsd:unsignedByte</a>	0...255 (8 bit)
	<a href="#">xsd:unsignedShort</a>	0...65535 (16 bit)
	<a href="#">xsd:unsignedInt</a>	0...4294967295 (32 bit)
	<a href="#">xsd:unsignedLong</a>	0...18446744073709551615 (64 bit)
	<a href="#">xsd:positiveInteger</a>	Integer numbers >0
	<a href="#">xsd:nonNegativeInteger</a>	Integer numbers ≥0
	<a href="#">xsd:negativeInteger</a>	Integer numbers <0
	<a href="#">xsd:nonPositiveInteger</a>	Integer numbers ≤0
<b>Encoded binary data</b>	<a href="#">xsd:hexBinary</a>	Hex-encoded binary data
	<a href="#">xsd:base64Binary</a>	Base64-encoded binary data
<b>Miscellaneous XSD types</b>	<a href="#">xsd:anyURI</a>	Absolute or relative URIs and IRIs
	<a href="#">xsd:language</a>	Language tags per [BCP47]
	<a href="#">xsd:normalizedString</a>	Whitespace-normalized strings
	<a href="#">xsd:token</a>	Tokenized strings
	<a href="#">xsd:NMTOKEN</a>	XML NMTOKENs
	<a href="#">xsd:Name</a>	XML Names
	<a href="#">xsd:NCName</a>	XML NCNames

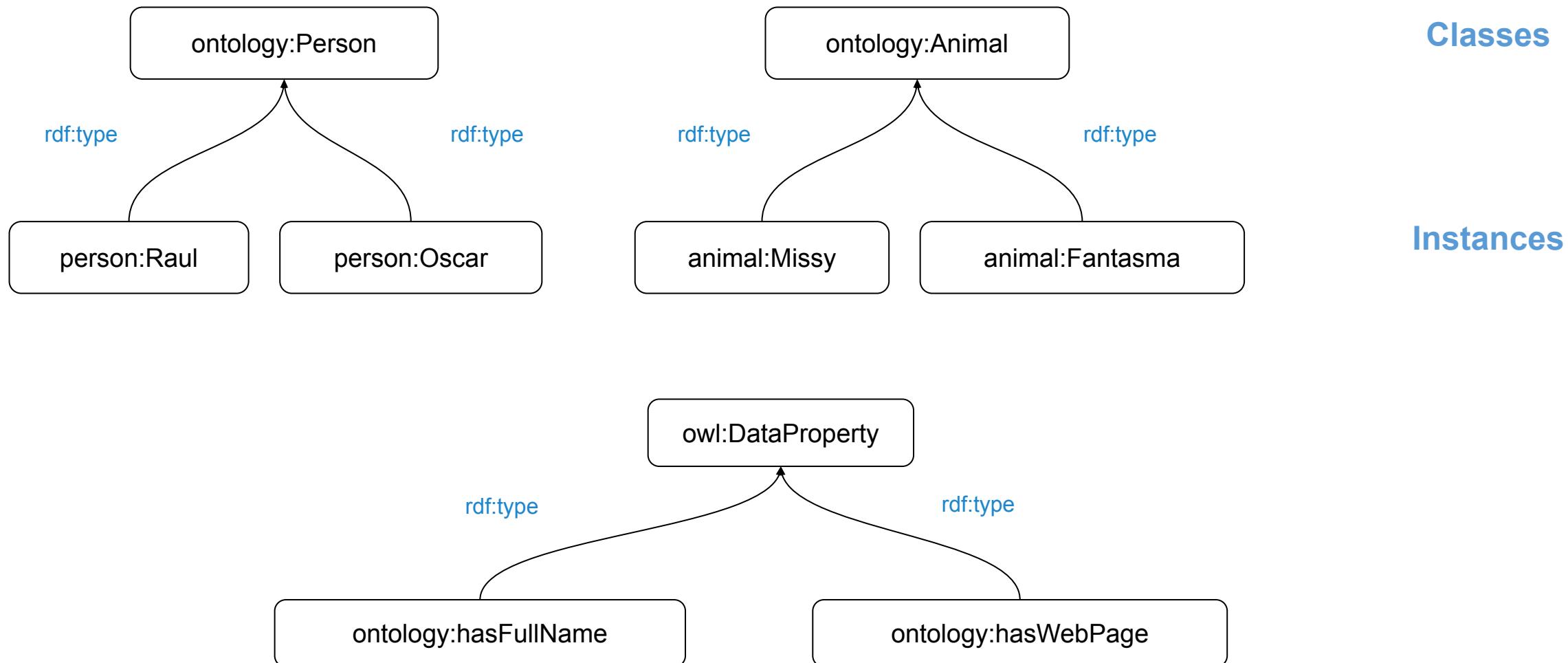


- URIs under a *namespace* are called **vocabularies**

Prefix	URI
person	http://oeg.fi.upm.es/resource/person/
organization	http://oeg.fi.upm.es/resource/organization/
ontology	http://oeg.fi.upm.es/def/people#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#

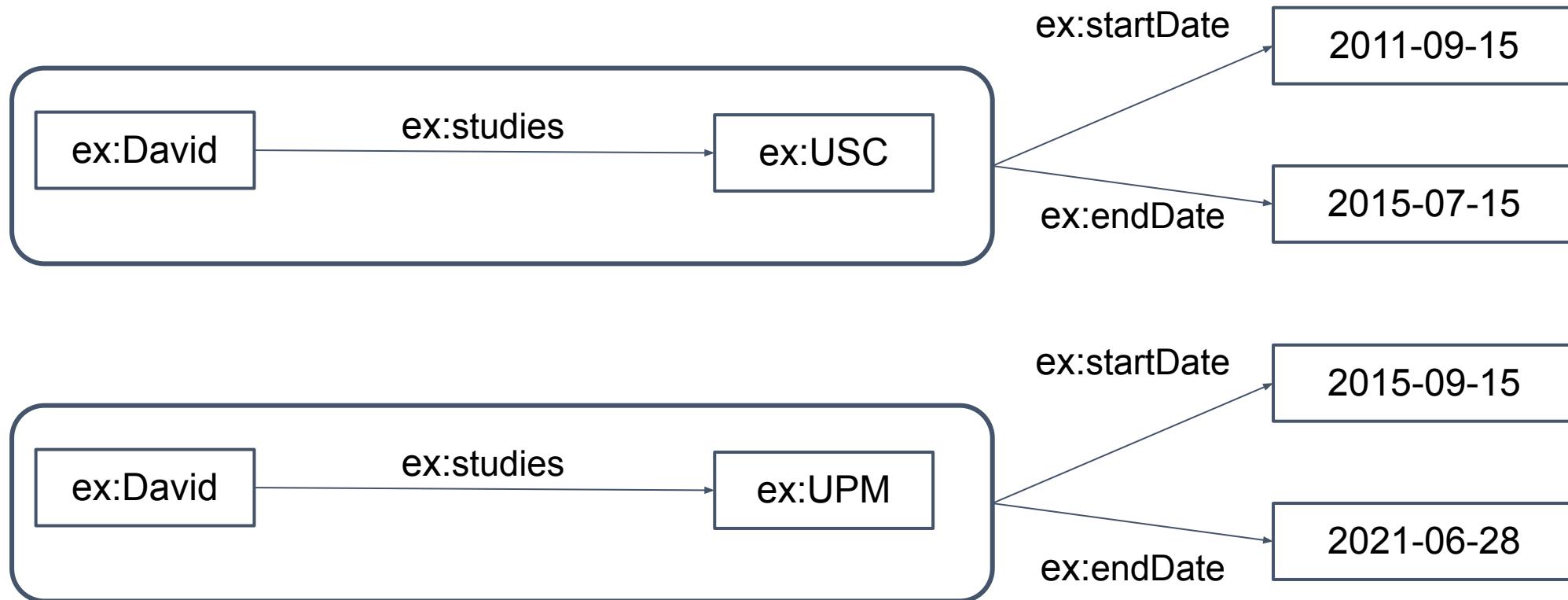


- `rdf:type` is used to associate resources to their corresponding categories/classes



# Statements about Statements (or RDF Reification)

“David studied in USC from 2011-09-15 to 2015-07-15 and in UPM from 2015-09-15 to 2021-06-28”



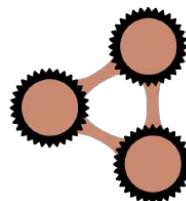
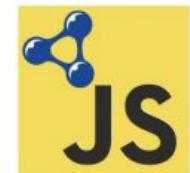
RDF reification: It allows expressing **beliefs (and other modalities), confidence models and metadata about data**

Triples that include a **triple as a subject or an object** are known as RDF-star triples

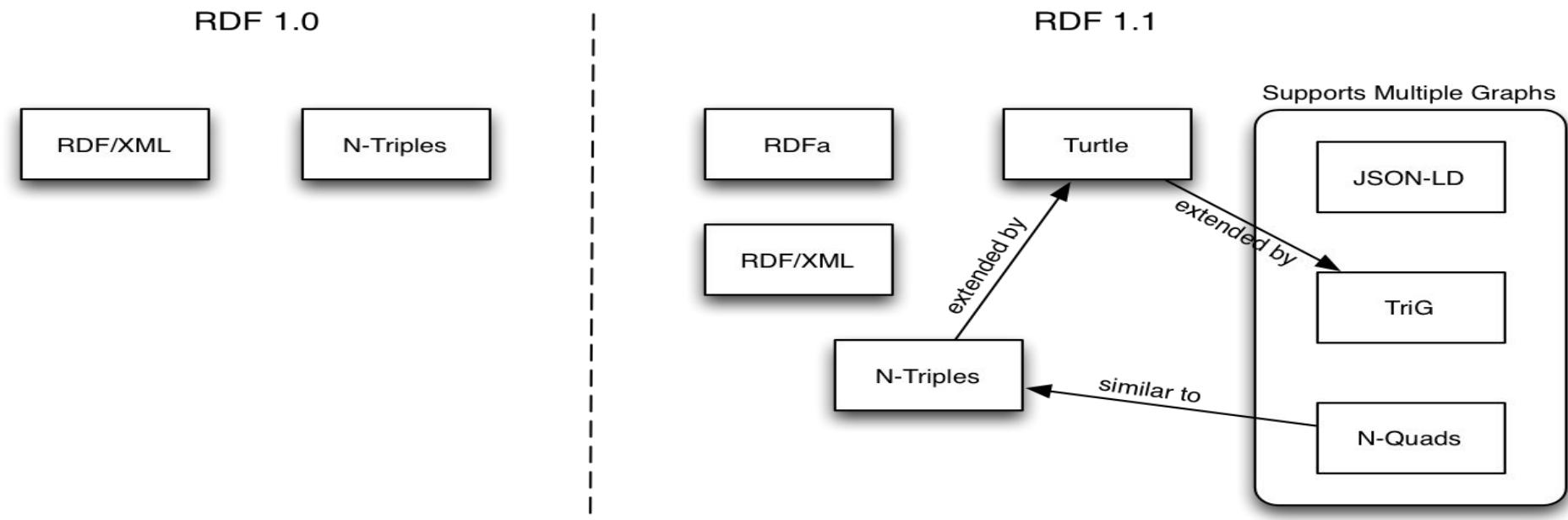
An RDF-star graph is a **set of RDF-star triples**.

**SPARQL-star** extends **SPARQL** to query RDF-star graphs

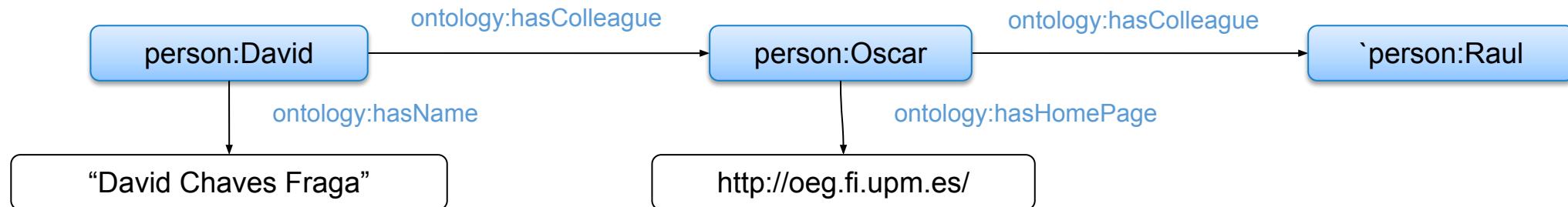
```
<< ex:David ex:studies ex:USC >> ex:startDate "2011-09-15" .  
<< ex:David ex:studies ex:USC >> ex:endDate "2015-07-15" .  
<< ex:David ex:studies ex:UPM >> ex:startDate "2015-09-15" .  
<< ex:David ex:studies ex:UPM >> ex:endDate "2021-06-28" .
```



- Different syntaxes (some of them recently approved – February 2014)
  - RDF/XML ([www.w3.org/TR/rdf-syntax-grammar/](http://www.w3.org/TR/rdf-syntax-grammar/))
  - Turtle (<http://www.w3.org/TR/turtle/>)
  - N-Triples (<http://www.w3.org/TR/n-triples/>)
  - TriG (<http://www.w3.org/TR/trig/>)
  - RDFA (<http://www.w3.org/TR/xhtml-rdfa-primer/>)
  - JSON-LD (<http://www.w3.org/TR/json-ld/>)
  - N-Quads (<http://www.w3.org/TR/n-quads/>)



- Important:** the order of triples inside a document is not relevant

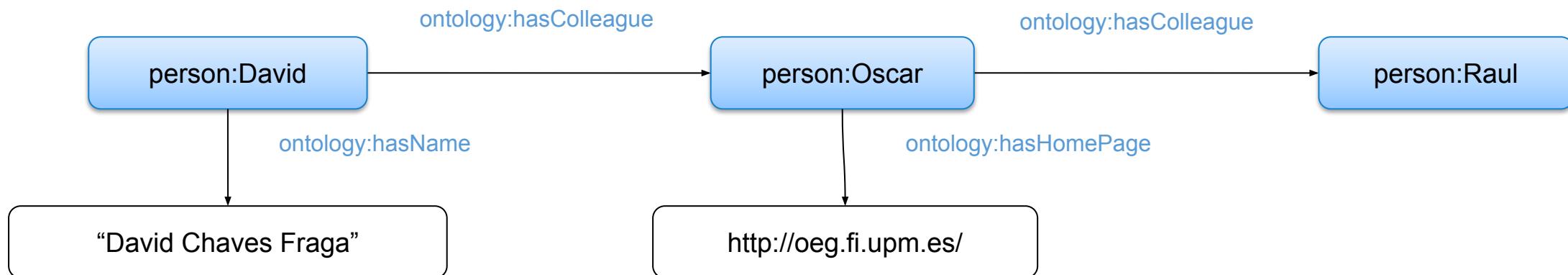


```

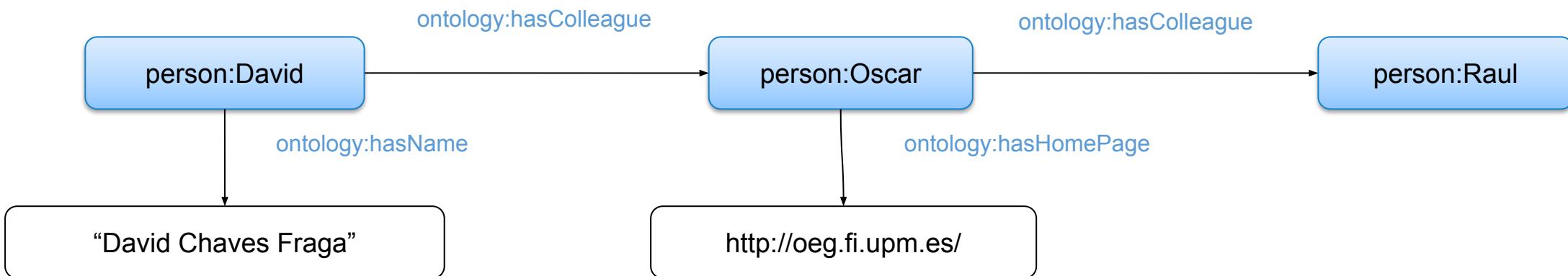
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ontology="http://oeg.fi.upm.es/def/people#"
  xmlns="http://oeg.fi.upm.es/resource/person/"
  xml:base="http://oeg.fi.upm.es/resource/person/" >

  <rdf:Property rdf:about="http://oeg.fi.upm.es/def/people#hasHomePage" />
  <rdf:Property rdf:about="http://oeg.fi.upm.es/def/people#hasColleague" />
  <rdf:Property rdf:about="http://oeg.fi.upm.es/def/people#hasName" />

  <rdf:Description rdf:about="Raul">
    <ontology:hasColleague rdf:resource="Oscar"/>
    <ontology:hasHomePage>http://www.oeg-upm.net/</ontology:hasHomePage>
  </rdf:Description>
  <rdf:Description rdf:about="David">
    <ontology:hasColleague rdf:resource="Oscar"/>
    <ontology:hasName>David Chaves Fraga</ontology:hasName>
  </rdf:Description>
</rdf:RDF>
  
```



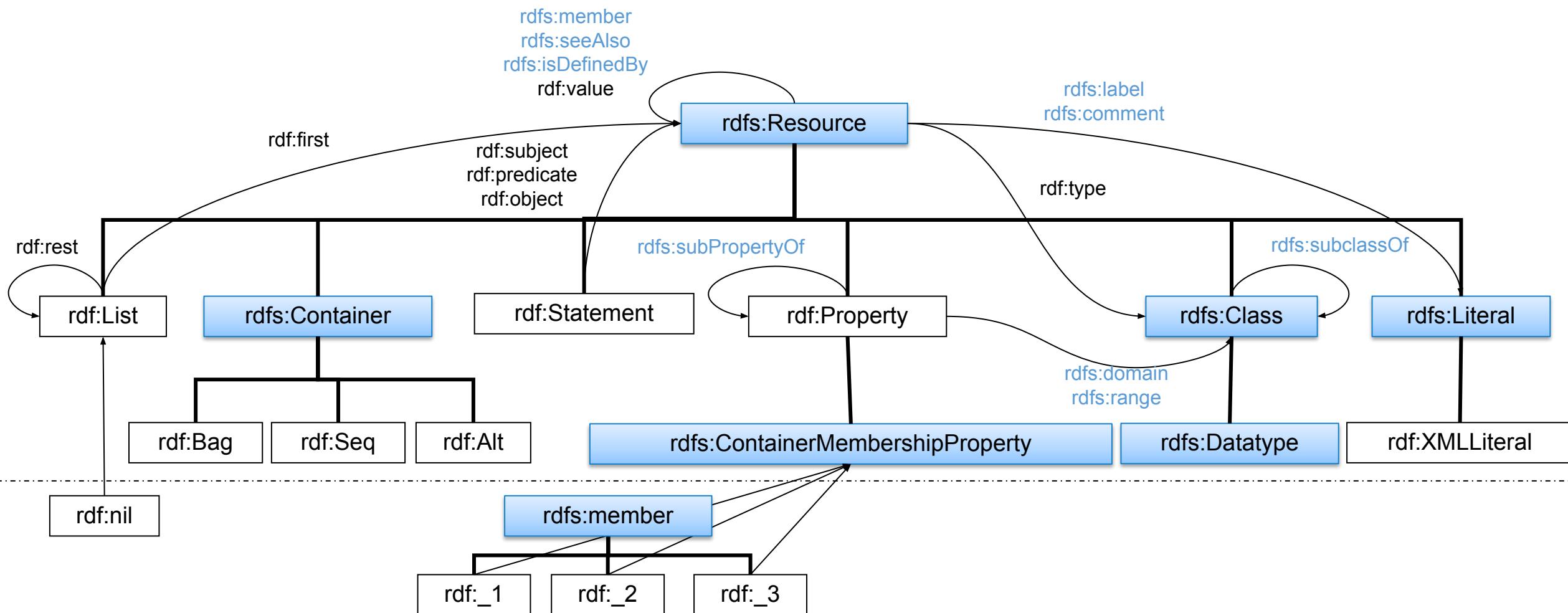
```
@base <http://oeg.fi.upm.es/resource/person/ >
@prefix ontology: <http://oeg.fi.upm.es/def/people#>
:Oscar ontology:hasColleague :Raul .
:Oscar ontology:hasHomePage <http://oeg.fi.upm.es/>.
:David ontology:hasColleague :Oscar .
:David ontology:hasName "David Chaves Fraga".
```



```
@base <http://oeg.fi.upm.es/resource/person/ >
@prefix ontology: <http://oeg.fi.upm.es/def/people#>
:Oscar    ontology:hasColleague :Raul ;
          ontology:hasHomePage <http://oeg.fi.upm.es/>.
:David   ontology:hasColleague :Oscar ;
          ontology:hasName "David Chaves Fraga".
```

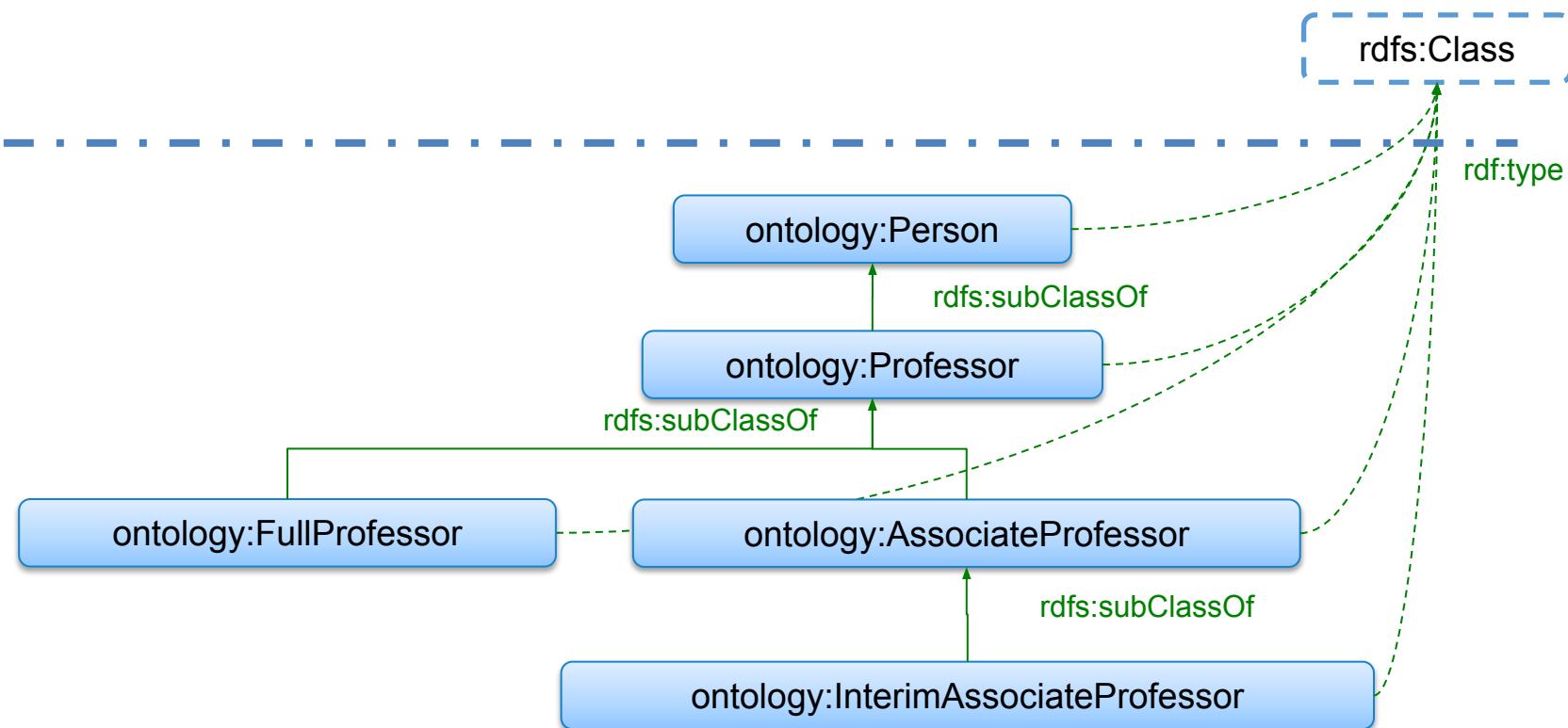
```
1 <http://ejemplo.es/recurso/Victor> <http://xmlns.com/foaf/0.1/age> 25 "years" .  
2 "Maria" <http://xmlns.com/foaf/0.1/age> 25 .  
3 <http://ejemplo.es/recurso/Maria> <http://xmlns.com/foaf/0.1/age> 25 .  
4 <http://ejemplo.es/recurso/Victor> <http://xmlns.com/foaf/0.1/isMarried> .  
5 <http://ejemplo.es/recurso/Victor> http://xmlns.com/foaf/0.1/livesIn "Madrid".  
6 <http://ejemplo.es/recurso/M> <http://xmlns.com/foaf/0.1/fn> "Maria alias "Mary"" .  
7 <http://ejemplo.es/recurso/M> <http://xmlns.com/foaf/0.1/fn> "Maria"^^xsd:string .
```

- Allows describing classes and properties
- Adds restrictions to models

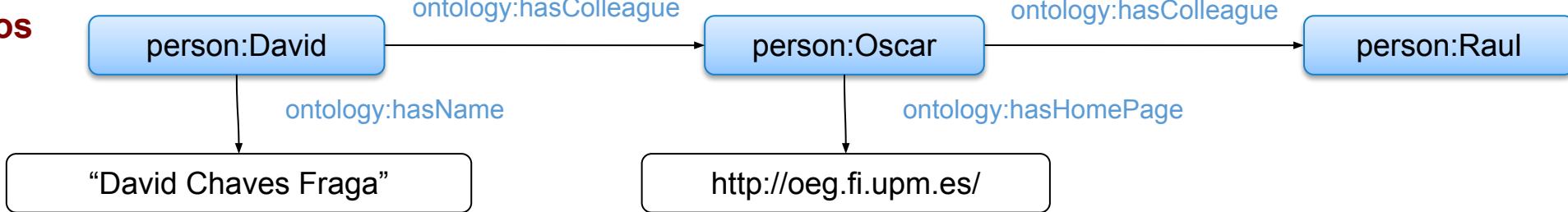


## RDF(S)

## Vocabulario



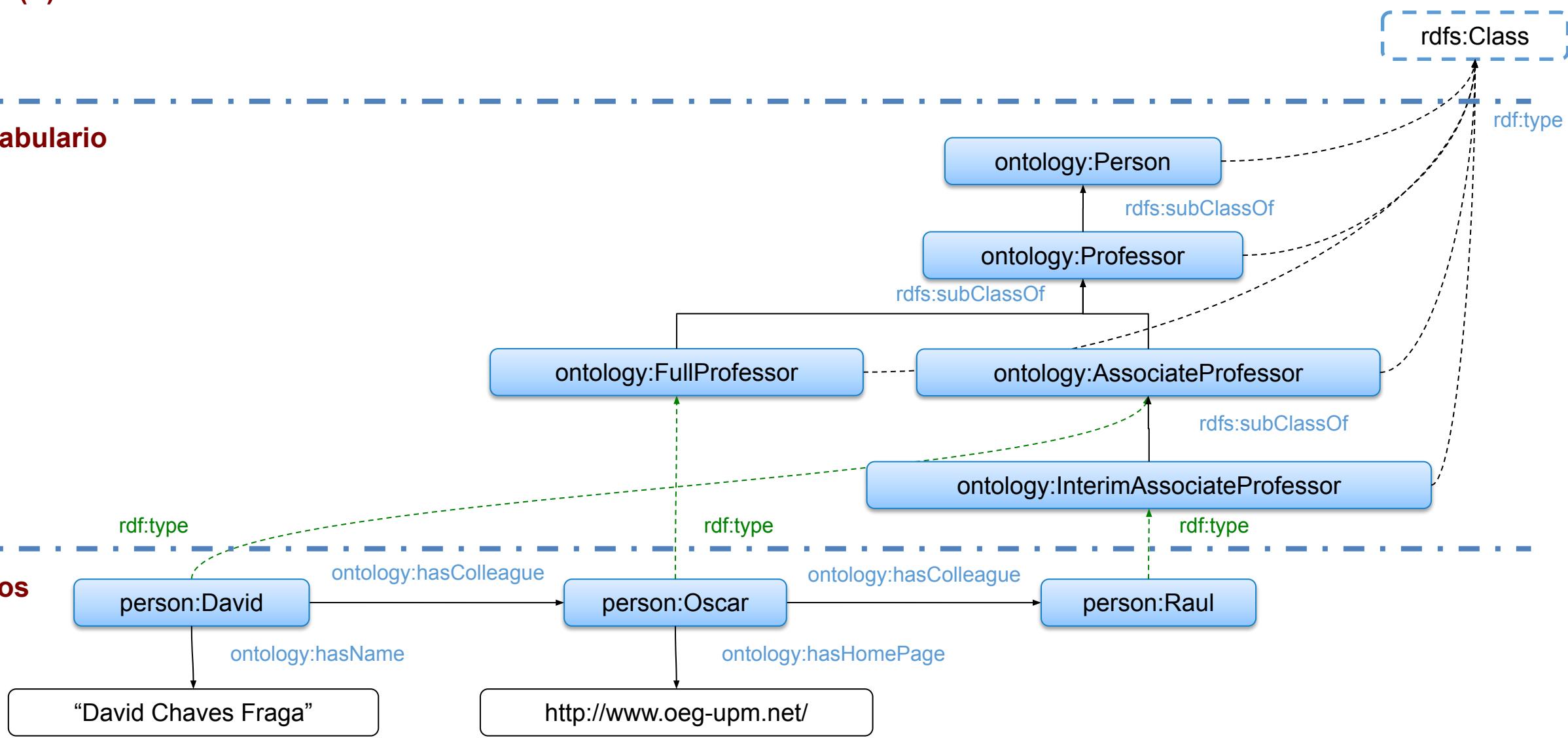
## Datos



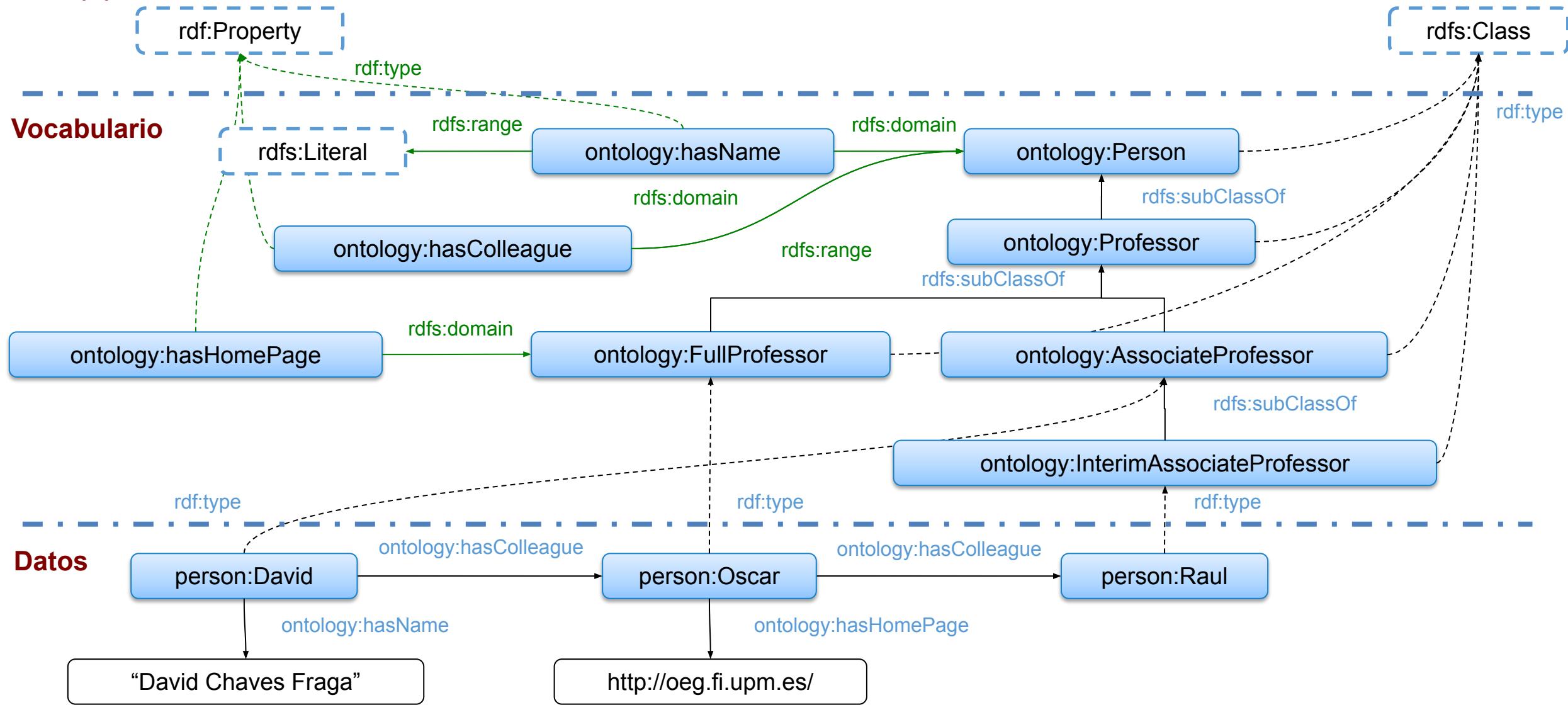
## RDF(S)

## Vocabulario

## Datos



## RDF(S)



- RDFS is too weak to describe resources in sufficient detail
  - No localised range and domain constraints
    - Can't say that the range of hasChild is person when applied to persons and elephant when applied to elephants
  - No existence/cardinality constraints
    - Can't say that all *instances* of person have a mother that is also a person, or that persons have exactly 2 parents
  - No boolean operators
    - Can't say or, not, etc.
  - No transitive, inverse or symmetrical properties
    - Can't say that isPartOf is a transitive property, that hasPart is the inverse of isPartOf or that touches is symmetrical
- Difficult to provide reasoning support
  - No “native” reasoners for non-standard semantics
  - May be possible to reason via FOL axiomatisation



W3C Recommendation (February 2004) / Built on top of RDF(S)

## 3 layers:

- OWL Lite
  - A small subset of primitives
  - Easier for frame-based tools to transition to
- OWL DL
  - Description logic
  - Decidable reasoning
- OWL Full
  - RDF extension, allows metaclasses

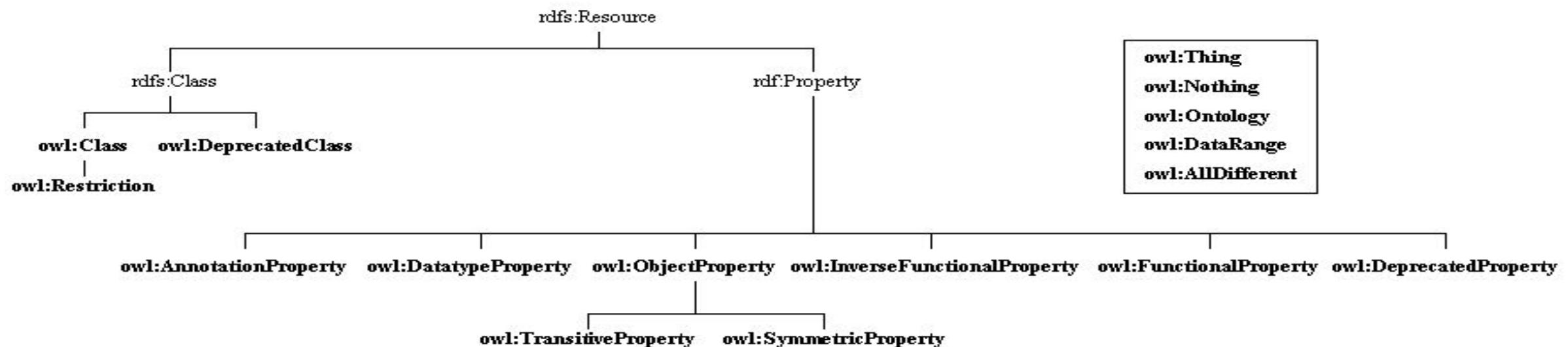


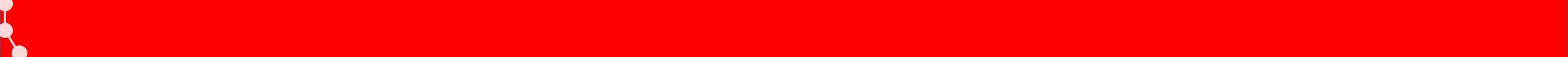
## Several syntaxes:

- Abstract syntax
- Manchester syntax
- RDF/XML



# Class taxonomy of the OWL KR ontology





# **SPARQL Query Language**

**(or how to query RDF data)**

<b>query</b>	<b>result</b>
SELECT queries	XML, JSON, TSV, CSV
ASK queries	true or false in XML or JSON
CONSTRUCT queries	RDF graph constructed with a template
DESCRIBE queries	RDF graph with data about resources

SELECT/CONSTRUCT/DESCRIBE/ASK ...

WHERE {

    <graph-pattern>

}

A SELECT query has two main components:

- a list of selected **variables** and
- a **WHERE** clause with the graph patterns to match

```
SELECT <variables>
WHERE {
    <graph-pattern>
}
```

```
SELECT ?s ?p ?o
WHERE {
    ?s ?p ?o .
}
```



```
SELECT ?p ?o
WHERE {
  ex:Katerina%20Stefanidi ?p ?o.
}
```

<b>p</b>	<b>o</b>
<a href="#"><u>rdf:type</u></a>	<a href="#"><u>foaf:Person</u></a>
<a href="#"><u>ex:score</u></a>	"4.85"^^xsd:decimal
<a href="#"><u>foaf:name</u></a>	"Katerina Stefanidi"@en
<a href="#"><u>ex:country</u></a>	<a href="#"><u>ex:EL</u></a>

```
ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>.
ex:Sandi%20Morris ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>.
ex:Alysha%20Newman ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.
```

```

SELECT ?s ?o

WHERE {
  ?s <http://xmlns.com/foaf/0.1/name> ?o .
}

```

s	o
<a href="#">ex:Anzhelika%20Sidorova</a>	"Anzhelika Sidorova"@en
<a href="#">ex:Sandi%20Morris</a>	"Sandi Morris"@en
<a href="#">ex:Katerina%20Stefanidi</a>	"Katerina Stefanidi"@en
<a href="#">ex:Holly%20Bradshaw</a>	"Holly Bradshaw"@en
<a href="#">ex:Alysha%20Newman</a>	"Alysha Newman"@en
<a href="#">ex:Angelica%20Bengtsson</a>	"Angelica Bengtsson"@en

```

ex:Anzhelika%20Sidorova rdf:type foaf:Person.
ex:Sandi%20Morris rdf:type foaf:Person.
ex:Katerina%20Stefanidi rdf:type foaf:Person.
ex:Holly%20Bradshaw rdf:type foaf:Person.
ex:Alysha%20Newman rdf:type foaf:Person.
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>.
ex:Sandi%20Morris ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw ex:country <http://ex.com/UK>.
ex:Alysha%20Newman ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson ex:country <http://ex.com/SE>.

```

```

SELECT ?s ?p
WHERE {
  ?s ?p "Katerina Stefanidi".
}

```

s	p
---	---

```

SELECT ?s ?p
WHERE {
  ?s ?p "Katerina Stefanidi"@en.
}

```

s	p
<a href="#">ex:Katerina%20Stefanidi</a>	<a href="#">foaf:name</a>

ex:Anzhelika%20Sidorova rdf:type foaf:Person.  
 ex:Sandi%20Morris rdf:type foaf:Person.  
 ex:Katerina%20Stefanidi rdf:type foaf:Person.  
 ex:Holly%20Bradshaw rdf:type foaf:Person.  
 ex:Alysha%20Newman rdf:type foaf:Person.  
 ex:Angelica%20Bengtsson rdf:type foaf:Person.  
  
 ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.  
 ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.  
 ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.  
 ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.  
 ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.  
 ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.  
  
 ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.  
 ex:Sandi%20Morris foaf:name "Sandi Morris"@en.  
 ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.  
 ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.  
 ex:Alysha%20Newman foaf:name "Alysha Newman"@en.  
 ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.  
  
 ex:Anzhelika%20Sidorova ex:country <[http://ex.com/RU](#)>.  
 ex:Sandi%20Morris ex:country <[http://ex.com/US](#)>.  
 ex:Katerina%20Stefanidi ex:country <[http://ex.com/EL](#)>.  
 ex:Holly%20Bradshaw ex:country <[http://ex.com/UK](#)>.  
 ex:Alysha%20Newman ex:country <[http://ex.com/CA](#)>.  
 ex:Angelica%20Bengtsson ex:country <[http://ex.com/SE](#)>.



```
SELECT ?s  
  
WHERE {  
    ?s ?p1 4.80;  
    ?p2 <http://ex.com/UK> .  
}
```

s
<a href="#">ex:Holly%20Bradshaw</a>

ex:[Anzhelika%20Sidorova](#) rdf:type foaf:Person.  
ex:[Sandi%20Morris](#) rdf:type foaf:Person.  
ex:[Katerina%20Stefanidi](#) rdf:type foaf:Person.  
ex:[Holly%20Bradshaw](#) rdf:type foaf:Person.  
ex:[Alysha%20Newman](#) rdf:type foaf:Person.  
ex:[Angelica%20Bengtsson](#) rdf:type foaf:Person.

ex:[Anzhelika%20Sidorova](#) ex:score "4.95"^^xsd:decimal.  
ex:[Sandi%20Morris](#) ex:score "4.90"^^xsd:decimal.  
ex:[Katerina%20Stefanidi](#) ex:score "4.85"^^xsd:decimal.  
ex:[Holly%20Bradshaw](#) ex:score "4.80"^^xsd:decimal.  
ex:[Alysha%20Newman](#) ex:score "4.80"^^xsd:decimal.  
ex:[Angelica%20Bengtsson](#) ex:score "4.80"^^xsd:decimal.

ex:[Anzhelika%20Sidorova](#) foaf:name "Anzhelika Sidorova"@en.  
ex:[Sandi%20Morris](#) foaf:name "Sandi Morris"@en.  
ex:[Katerina%20Stefanidi](#) foaf:name "Katerina Stefanidi"@en.  
ex:[Holly%20Bradshaw](#) foaf:name "Holly Bradshaw"@en.  
ex:[Alysha%20Newman](#) foaf:name "Alysha Newman"@en.  
ex:[Angelica%20Bengtsson](#) foaf:name "Angelica Bengtsson"@en.

ex:[Anzhelika%20Sidorova](#) ex:country <<http://ex.com/RU>>.  
ex:[Sandi%20Morris](#) ex:country <<http://ex.com/US>>.  
ex:[Katerina%20Stefanidi](#) ex:country <<http://ex.com/EL>>.  
ex:[Holly%20Bradshaw](#) ex:country <<http://ex.com/UK>>.  
ex:[Alysha%20Newman](#) ex:country <<http://ex.com/CA>>.  
ex:[Angelica%20Bengtsson](#) ex:country <<http://ex.com/SE>>.

PREFIX ex: <http://ex.com/>

```
SELECT ?s ?o
WHERE {
  ?s ex:score ?o .
  FILTER (?o < 4.90) .
}
```

s	o
ex:Holly%20Bradshaw	"4.80"^^xsd:decimal
ex:Alysha%20Newman	"4.80"^^xsd:decimal
ex:Angelica%20Bengtsson	"4.80"^^xsd:decimal
ex:Katerina%20Stefanidi	"4.85"^^xsd:decimal

ex:Anzhelika%20Sidorova	rdf:type foaf:Person.
ex:Sandi%20Morris	rdf:type foaf:Person.
ex:Katerina%20Stefanidi	rdf:type foaf:Person.
ex:Holly%20Bradshaw	rdf:type foaf:Person.
ex:Alysha%20Newman	rdf:type foaf:Person.
ex:Angelica%20Bengtsson	rdf:type foaf:Person.
ex:Anzhelika%20Sidorova	ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris	ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi	ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw	ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman	ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson	ex:score "4.80"^^xsd:decimal.
ex:Anzhelika%20Sidorova	foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris	foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi	foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw	foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman	foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson	foaf:name "Angelica Bengtsson"@en.
ex:Anzhelika%20Sidorova	ex:country <http://ex.com/RU>.
ex:Sandi%20Morris	ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi	ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw	ex:country <http://ex.com/UK>.
ex:Alysha%20Newman	ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson	ex:country <http://ex.com/SE>.

PREFIX ex: <http://ex.com/>

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

SELECT ?s ?o

WHERE {

?s foaf:name ?o.

FILTER regex(?o, "A").

}

s	o
<a href="#">ex:Anzhelika%20Sidorova</a>	"Anzhelika Sidorova"@en
<a href="#">ex:Alysha%20Newman</a>	"Alysha Newman"@en
<a href="#">ex:Angelica%20Bengtsson</a>	"Angelica Bengtsson"@en

ex:Anzhelika%20Sidorova rdf:type foaf:Person.  
 ex:Sandi%20Morris rdf:type foaf:Person.  
 ex:Katerina%20Stefanidi rdf:type foaf:Person.  
 ex:Holly%20Bradshaw rdf:type foaf:Person.  
 ex:Alysha%20Newman rdf:type foaf:Person.  
 ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.  
 ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.  
 ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.  
 ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.  
 ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.  
 ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.  
 ex:Sandi%20Morris foaf:name "Sandi Morris"@en.  
 ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.  
 ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.  
 ex:Alysha%20Newman foaf:name "Alysha Newman"@en.  
 ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <[http://ex.com/RU](#)>.  
 ex:Sandi%20Morris ex:country <[http://ex.com/US](#)>.  
 ex:Katerina%20Stefanidi ex:country <[http://ex.com/EL](#)>.  
 ex:Holly%20Bradshaw ex:country <[http://ex.com/UK](#)>.  
 ex:Alysha%20Newman ex:country <[http://ex.com/CA](#)>.  
 ex:Angelica%20Bengtsson ex:country <[http://ex.com/SE](#)>.

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?o ?c
WHERE { ?s foaf:name ?o .
      OPTIONAL { ?s ex:country ?c } . }
```

s	o	c
<a href="#">ex:Alysha%20Newman</a>	"Alysha Newman"@en	<a href="#">ex:CA</a>
<a href="#">ex:Angelica%20Bengtsson</a>	"Angelica Bengtsson"@en	<a href="#">ex:SE</a>
<a href="#">ex:Anzhelika%20Sidorova</a>	"Anzhelika Sidorova"@en	<a href="#">ex:RU</a>
<a href="#">ex:Holly%20Bradshaw</a>	"Holly Bradshaw"@en	<a href="#">ex:UK</a>
<a href="#">ex:Katerina%20Stefanidi</a>	"Katerina Stefanidi"@en	<a href="#">ex:EL</a>
<a href="#">ex:Sandi%20Morris</a>	"Sandi Morris"@en	

<a href="#">ex:Anzhelika%20Sidorova</a>	rdf:type foaf:Person.	
<a href="#">ex:Sandi%20Morris</a>	rdf:type foaf:Person.	
<a href="#">ex:Katerina%20Stefanidi</a>	rdf:type foaf:Person.	
<a href="#">ex:Holly%20Bradshaw</a>	rdf:type foaf:Person.	
<a href="#">ex:Alysha%20Newman</a>	rdf:type foaf:Person.	
<a href="#">ex:Angelica%20Bengtsson</a>	rdf:type foaf:Person.	
<a href="#">ex:Anzhelika%20Sidorova</a>	ex:score "4.95"^^xsd:decimal.	
<a href="#">ex:Sandi%20Morris</a>	ex:score "4.90"^^xsd:decimal.	
<a href="#">ex:Katerina%20Stefanidi</a>	ex:score "4.85"^^xsd:decimal.	
<a href="#">ex:Holly%20Bradshaw</a>	ex:score "4.80"^^xsd:decimal.	
<a href="#">ex:Alysha%20Newman</a>	ex:score "4.80"^^xsd:decimal.	
<a href="#">ex:Angelica%20Bengtsson</a>	ex:score "4.80"^^xsd:decimal.	
<a href="#">ex:Anzhelika%20Sidorova</a>	foaf:name "Anzhelika Sidorova"@en.	
<a href="#">ex:Sandi%20Morris</a>	foaf:name "Sandi Morris"@en.	
<a href="#">ex:Katerina%20Stefanidi</a>	foaf:name "Katerina Stefanidi"@en.	
<a href="#">ex:Holly%20Bradshaw</a>	foaf:name "Holly Bradshaw"@en.	
<a href="#">ex:Alysha%20Newman</a>	foaf:name "Alysha Newman"@en.	
<a href="#">ex:Angelica%20Bengtsson</a>	foaf:name "Angelica Bengtsson"@en.	
<a href="#">ex:Anzhelika%20Sidorova</a>	ex:country < <a href="#">http://ex.com/RU</a> >.	
<del><a href="#">ex:Sandi%20Morris</a></del>	<del>ex:country &lt;<a href="#">http://ex.com/US</a>&gt;</del>	
<a href="#">ex:Katerina%20Stefanidi</a>	ex:country < <a href="#">http://ex.com/EL</a> >.	
<a href="#">ex:Holly%20Bradshaw</a>	ex:country < <a href="#">http://ex.com/UK</a> >.	
<a href="#">ex:Alysha%20Newman</a>	ex:country < <a href="#">http://ex.com/CA</a> >.	
<a href="#">ex:Angelica%20Bengtsson</a>	ex:country < <a href="#">http://ex.com/SE</a> >.	



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

SELECT ?s ?o ?c
WHERE {
    ?s foaf:name ?o.
    { ?s ex:country ?c }
    UNION
    { ?s foaf:country ?c }
}
```

```
ex:Anzhelika%20Sidorova ex:country <http://ex.com/RU>.
ex:Sandi%20Morris ex:country <http://ex.com/US>.
ex:Katerina%20Stefanidi ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw foaf:country <http://ex.com/UK>.
ex:Alysha%20Newman foaf:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson foaf:country <http://ex.com/SE>.
```

s	o	c
<a href="#">ex:Anzhelika%20Sidorova</a>	"Anzhelika Sidorova"@en	<a href="#">ex:RU</a>
<a href="#">ex:Katerina%20Stefanidi</a>	"Katerina Stefanidi"@en	<a href="#">ex:EL</a>
<a href="#">ex:Holly%20Bradshaw</a>	"Holly Bradshaw"@en	<a href="#">ex:UK</a>
<a href="#">ex:Alysha%20Newman</a>	"Alysha Newman"@en	<a href="#">ex:CA</a>
<a href="#">ex:Angelica%20Bengtsson</a>	"Angelica Bengtsson"@en	<a href="#">ex:SE</a>
<a href="#">ex:Sandi%20Morris</a>	"Sandi Morris"@en	<a href="#">ex:US</a>

PREFIX ex: <<http://ex.com/>>

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

SELECT ?s ?n ?o

WHERE {

?s a foaf:Person .

?s foaf:name ?n .

FILTER NOT EXISTS { ?s ex:country ?o }

}



ex:Anzhelika%20Sidorova rdf:type foaf:Person.  
 ex:Sandi%20Morris rdf:type foaf:Person.  
 ex:Katerina%20Stefanidi rdf:type foaf:Person.  
 ex:Holly%20Bradshaw rdf:type foaf:Person.  
 ex:Alysha%20Newman rdf:type foaf:Person.  
 ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.  
 ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.  
 ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.  
 ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.  
 ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.  
 ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.  
 ex:Sandi%20Morris foaf:name "Sandi Morris"@en.  
 ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.  
 ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.  
 ex:Alysha%20Newman foaf:name "Alysha Newman"@en.  
 ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

s	o	c
<a href="#">ex:Holly%20Bradshaw</a>	"Holly Bradshaw"@en	<a href="#">ex:UK</a>
<a href="#">ex:Sandi%20Morris</a>	"Sandi Morris"@en	<a href="#">ex:US</a>

ex:Anzhelika%20Sidorova ex:country <<http://ex.com/RU>>.  
 ex:Sandi%20Morris foaf:country <<http://ex.com/US>>.  
 ex:Katerina%20Stefanidi ex:country <<http://ex.com/EL>>.  
 ex:Holly%20Bradshaw foaf:country <<http://ex.com/UK>>.  
 ex:Alysha%20Newman ex:country <<http://ex.com/CA>>.  
 ex:Angelica%20Bengtsson ex:country <<http://ex.com/SE>>.

PREFIX ex: <<http://ex.com/>>

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

SELECT ?o (COUNT(?s) AS ?total)

WHERE {

?s a ?o.

}

GROUP BY ?o

<b>o</b>	<b>total</b>
<a href="#">foaf:Person</a>	"6" <sup>^^xsd:integer</sup>

ex:[Anzhelika%20Sidorova](#) rdf:type foaf:Person.  
 ex:[Sandi%20Morris](#) rdf:type foaf:Person.  
 ex:[Katerina%20Stefanidi](#) rdf:type foaf:Person.  
 ex:[Holly%20Bradshaw](#) rdf:type foaf:Person.  
 ex:[Alysha%20Newman](#) rdf:type foaf:Person.  
 ex:[Angelica%20Bengtsson](#) rdf:type foaf:Person.

ex:[Anzhelika%20Sidorova](#) ex:score "4.95"<sup>^^xsd:decimal</sup>.  
 ex:[Sandi%20Morris](#) ex:score "4.90"<sup>^^xsd:decimal</sup>.  
 ex:[Katerina%20Stefanidi](#) ex:score "4.85"<sup>^^xsd:decimal</sup>.  
 ex:[Holly%20Bradshaw](#) ex:score "4.80"<sup>^^xsd:decimal</sup>.  
 ex:[Alysha%20Newman](#) ex:score "4.80"<sup>^^xsd:decimal</sup>.  
 ex:[Angelica%20Bengtsson](#) ex:score "4.80"<sup>^^xsd:decimal</sup>.

ex:[Anzhelika%20Sidorova](#) foaf:name "Anzhelika Sidorova"@en.  
 ex:[Sandi%20Morris](#) foaf:name "Sandi Morris"@en.  
 ex:[Katerina%20Stefanidi](#) foaf:name "Katerina Stefanidi"@en.  
 ex:[Holly%20Bradshaw](#) foaf:name "Holly Bradshaw"@en.  
 ex:[Alysha%20Newman](#) foaf:name "Alysha Newman"@en.  
 ex:[Angelica%20Bengtsson](#) foaf:name "Angelica Bengtsson"@en.

ex:[Anzhelika%20Sidorova](#) ex:country <<http://ex.com/RU>>.  
 ex:[Sandi%20Morris](#) foaf:country <<http://ex.com/US>>.  
 ex:[Katerina%20Stefanidi](#) ex:country <<http://ex.com/EL>>.  
 ex:[Holly%20Bradshaw](#) foaf:country <<http://ex.com/UK>>.  
 ex:[Alysha%20Newman](#) ex:country <<http://ex.com/CA>>.  
 ex:[Angelica%20Bengtsson](#) ex:country <<http://ex.com/SE>>.

PREFIX ex: <<http://ex.com/>>

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

```
SELECT ?o (COUNT(?s) AS ?total)
WHERE {
  ?s foaf:name ?n ; ex:score ?o .
}
GROUP BY ?o
HAVING (?o < 4.85)
```

<b>o</b>	<b>total</b>
"4.80"^^xsd:decimal	"3"^^xsd:integer

ex:Anzhelika%20Sidorova rdf:type foaf:Person.  
 ex:Sandi%20Morris rdf:type foaf:Person.  
 ex:Katerina%20Stefanidi rdf:type foaf:Person.  
 ex:Holly%20Bradshaw rdf:type foaf:Person.  
 ex:Alysha%20Newman rdf:type foaf:Person.  
 ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.  
 ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.  
 ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.  
 ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.  
 ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.  
 ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.  
 ex:Sandi%20Morris foaf:name "Sandi Morris"@en.  
 ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.  
 ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.  
 ex:Alysha%20Newman foaf:name "Alysha Newman"@en.  
 ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <<http://ex.com/RU>>.  
 ex:Sandi%20Morris foaf:country <<http://ex.com/US>>.  
 ex:Katerina%20Stefanidi ex:country <<http://ex.com/EL>>.  
 ex:Holly%20Bradshaw foaf:country <<http://ex.com/UK>>.  
 ex:Alysha%20Newman ex:country <<http://ex.com/CA>>.  
 ex:Angelica%20Bengtsson ex:country <<http://ex.com/SE>>.

PREFIX ex: <<http://ex.com/>>

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

CONSTRUCT { ?s ex:name ?o }

WHERE {

?s a ?c.

?s foaf:name ?o.

}

ex:Anzhelika%20Sidorova rdf:type foaf:Person.  
ex:Sandi%20Morris rdf:type foaf:Person.  
ex:Katerina%20Stefanidi rdf:type foaf:Person.  
ex:Holly%20Bradshaw rdf:type foaf:Person.  
ex:Alysha%20Newman rdf:type foaf:Person.  
ex:Angelica%20Bengtsson rdf:type foaf:Person.

ex:Anzhelika%20Sidorova ex:score "4.95"^^xsd:decimal.  
ex:Sandi%20Morris ex:score "4.90"^^xsd:decimal.  
ex:Katerina%20Stefanidi ex:score "4.85"^^xsd:decimal.  
ex:Holly%20Bradshaw ex:score "4.80"^^xsd:decimal.  
ex:Alysha%20Newman ex:score "4.80"^^xsd:decimal.  
ex:Angelica%20Bengtsson ex:score "4.80"^^xsd:decimal.

ex:Anzhelika%20Sidorova foaf:name "Anzhelika Sidorova"@en.  
ex:Sandi%20Morris foaf:name "Sandi Morris"@en.  
ex:Katerina%20Stefanidi foaf:name "Katerina Stefanidi"@en.  
ex:Holly%20Bradshaw foaf:name "Holly Bradshaw"@en.  
ex:Alysha%20Newman foaf:name "Alysha Newman"@en.  
ex:Angelica%20Bengtsson foaf:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova ex:country <<http://ex.com/RU>>.  
ex:Sandi%20Morris foaf:country <<http://ex.com/US>>.  
ex:Katerina%20Stefanidi ex:country <<http://ex.com/EL>>.  
ex:Holly%20Bradshaw foaf:country <<http://ex.com/UK>>.  
ex:Alysha%20Newman ex:country <<http://ex.com/CA>>.  
ex:Angelica%20Bengtsson ex:country <<http://ex.com/SE>>.

```
PREFIX ex: <http://ex.com/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT { ?s ex:name ?o }
WHERE {
  ?s a ?c.
  ?s foaf:name ?o.
}
```

ex:Anzhelika%20Sidorova	ex:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris	ex:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi	ex:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw	ex:name "Holly Bradshaw"@en.
ex:Alysha%20Newman	ex:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson	ex:name "Angelica Bengtsson"@en.

ex:Anzhelika%20Sidorova	rdf:type foaf:Person.
ex:Sandi%20Morris	rdf:type foaf:Person.
ex:Katerina%20Stefanidi	rdf:type foaf:Person.
ex:Holly%20Bradshaw	rdf:type foaf:Person.
ex:Alysha%20Newman	rdf:type foaf:Person.
ex:Angelica%20Bengtsson	rdf:type foaf:Person.
ex:Anzhelika%20Sidorova	ex:score "4.95"^^xsd:decimal.
ex:Sandi%20Morris	ex:score "4.90"^^xsd:decimal.
ex:Katerina%20Stefanidi	ex:score "4.85"^^xsd:decimal.
ex:Holly%20Bradshaw	ex:score "4.80"^^xsd:decimal.
ex:Alysha%20Newman	ex:score "4.80"^^xsd:decimal.
ex:Angelica%20Bengtsson	ex:score "4.80"^^xsd:decimal.
ex:Anzhelika%20Sidorova	foaf:name "Anzhelika Sidorova"@en.
ex:Sandi%20Morris	foaf:name "Sandi Morris"@en.
ex:Katerina%20Stefanidi	foaf:name "Katerina Stefanidi"@en.
ex:Holly%20Bradshaw	foaf:name "Holly Bradshaw"@en.
ex:Alysha%20Newman	foaf:name "Alysha Newman"@en.
ex:Angelica%20Bengtsson	foaf:name "Angelica Bengtsson"@en.
ex:Anzhelika%20Sidorova	ex:country <http://ex.com/RU>.
ex:Sandi%20Morris	foaf:country <http://ex.com/US>.
ex:Katerina%20Stefanidi	ex:country <http://ex.com/EL>.
ex:Holly%20Bradshaw	foaf:country <http://ex.com/UK>.
ex:Alysha%20Newman	ex:country <http://ex.com/CA>.
ex:Angelica%20Bengtsson	ex:country <http://ex.com/SE>.

Amazon Neptune <https://aws.amazon.com/neptune/>

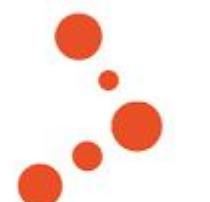
BlazeGraph <https://blazegraph.com/>

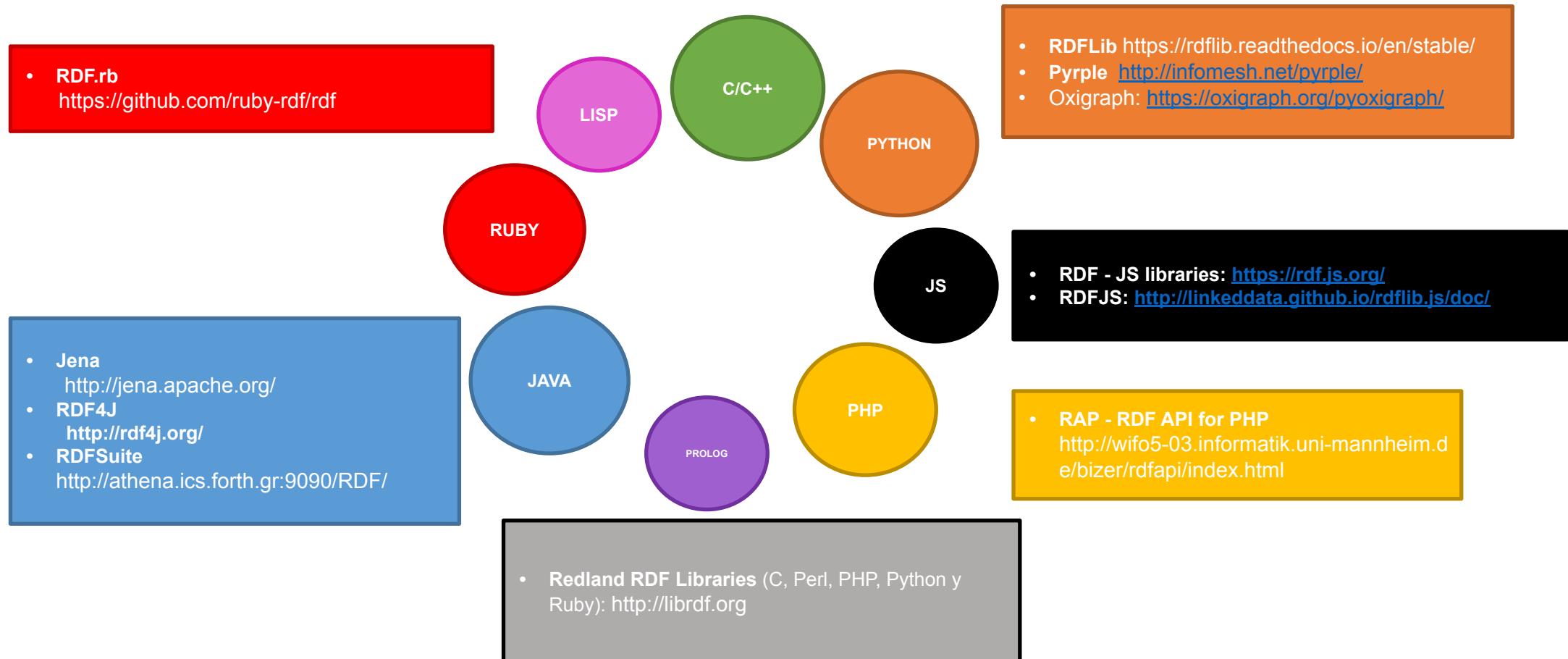
GraphDB <https://graphdb.ontotext.com/>

OpenLink Virtuoso <https://lod.openlinksw.com/sparql>

Oracle Graph Database <https://www.oracle.com/be/database/graph/>

Stardog <https://www.stardog.com/platform/>





Full list available at  
<https://www.w3.org/2001/sw/wiki/Tools>



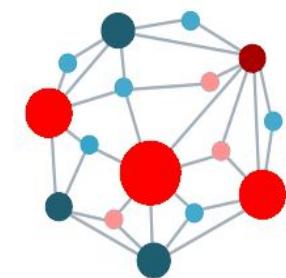
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) **Hands-On 1: Querying Wikidata**



## Hands-On 2: Constructing and validating your RDF graph (50min)

- 1) Tools for Developing Ontologies and KGs
- 2) RML (from CSV/XML/JSON to RDF)
- 3) SHACL (RDF validation)



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project



## Wikidata-Queries Examples



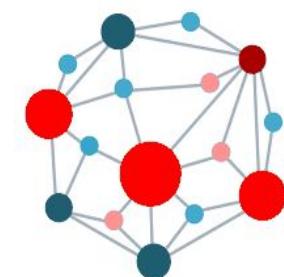
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) Hands-On 1: Querying Wikidata



## Constructing and validating your RDF graph (50min)

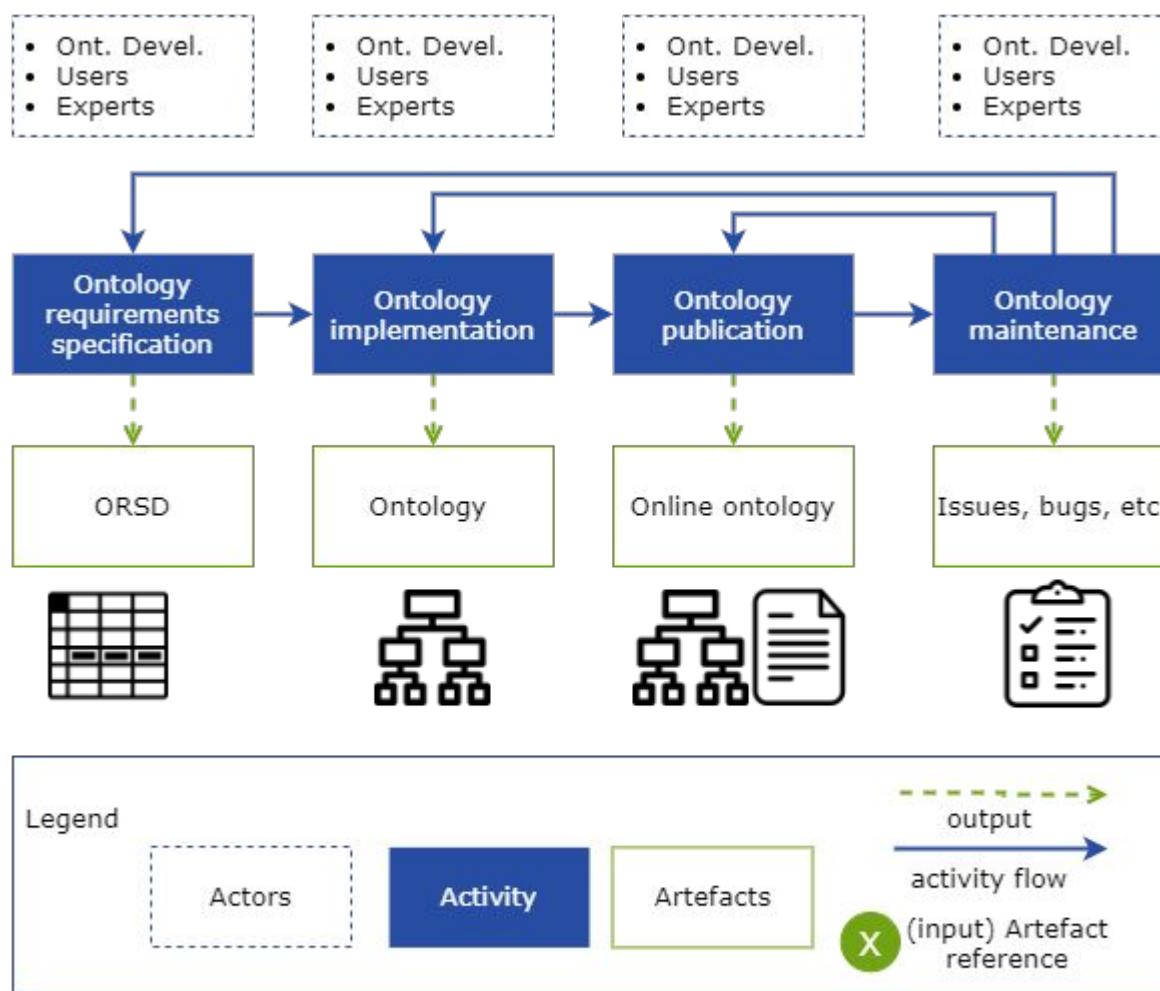
- 1) Tools for Developing Ontologies and KGs**
- 2) RML (from CSV/XML/JSON to RDF) and SHACL (RDF validation)
- 3) Hands-On 2: Construct a KG as a knowledge engineer



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project





The Linked Open Terms Methodology:

<https://lot.linkeddata.es/>



LOT Methodology <http://lot.linkeddata.es>  
@Ontology Engineering Group

The screenshot shows the LOV homepage. At the top, there's a navigation bar with icons for VOCABS, TERMS, AGENTS, and SPARQL/DUMP. Below the header is a large teal banner with the text "Linked Open Vocabularies (LOV)". Underneath the banner are several interactive buttons: "+ Suggest", "Documentation", "Follow", a search bar, and a user icon. To the left, a teal box displays the text "792 Vocabularies in LOV" above a circular network graph where nodes represent vocabularies like dul, gn, frbr, gleif-b, doap, voaf, sioc, obo, and dctype. To the right, a section titled "Latest insertion" lists three entries: "puv - Parameter Usage Vocabulary ontology" (2022-10-27), "dtx\_srti - LOD SRTI DATEX II" (2022-10-13), and "sealit - SeaLiT Ontology" (2022-09-21).

- Reuse vocabularies is a must!
- Resource: <https://lov.linkeddata.es/>



# Chowlk Converter

Chowlk takes as input an ontology conceptualization made with diagrams.net and generates its implementation in OWL.

## Authors:

María Poveda-Villalón (Ontology Engineering Group, Universidad Politécnica de Madrid)

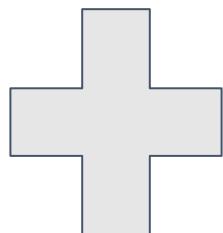
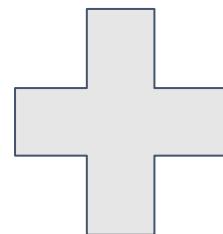
Serge Chávez-Feria (Ontology Engineering Group, Universidad Politécnica de Madrid)

- Use diagrams.net instead of protégé (de-facto standard)
- Conceptualization of the ontology through human friendly user interface
- Translation from XML to OWL
- All info: <https://chowlk.linkeddata.es/>



Ontoology automate parts of the **collaborative ontology development process**. Given a repository with an owl file, OnToology will survey it and produce **diagrams, a complete documentation and validation based on common pitfalls**. OnToology also creates an issue in Github with an evaluation summary.

Let's try: <http://ontoology.linkeddata.es/>





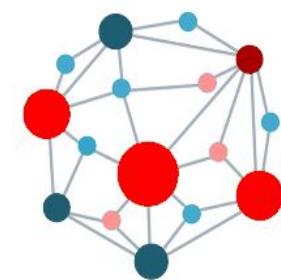
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) Hands-On 1: Querying Wikidata



## Hands-On 2: Constructing and validating your RDF graph (50min)

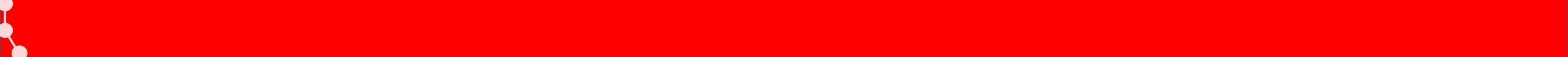
- 1) Tools for Developing Ontologies and KGs
- 2) **RML (from CSV/XML/JSON to RDF)**
- 3) **SHACL (RDF validation)**



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project



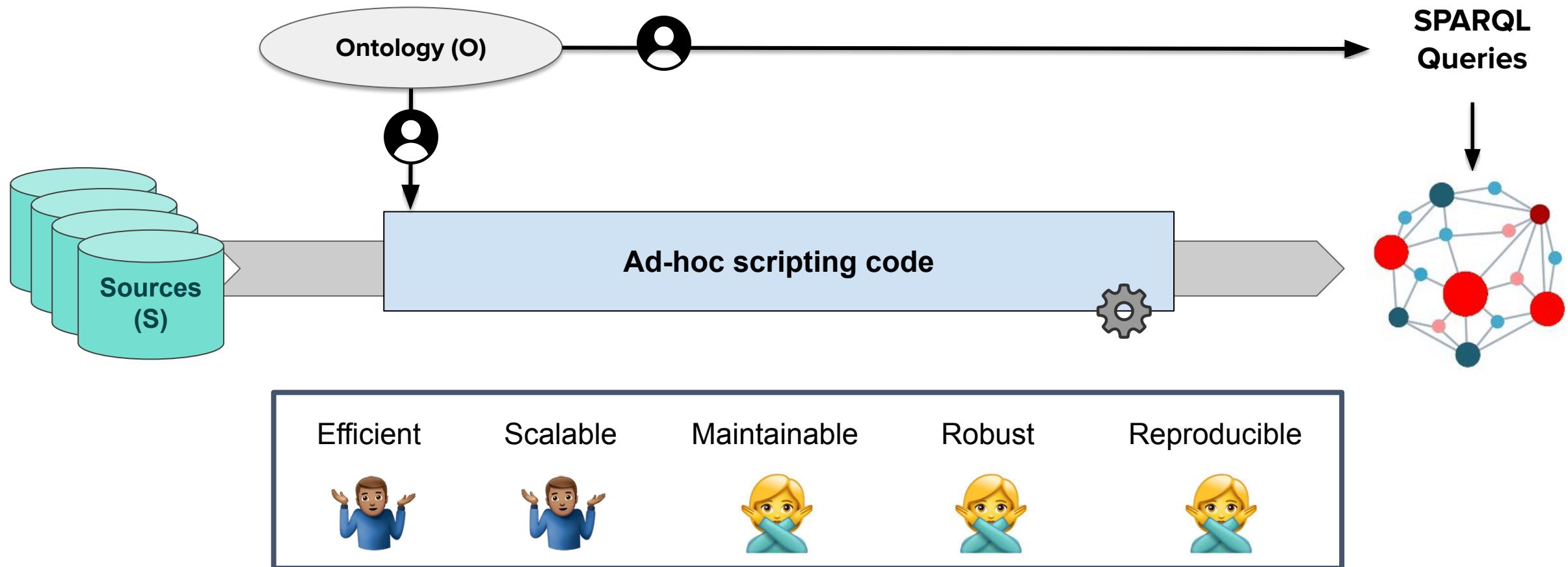


# **RML**

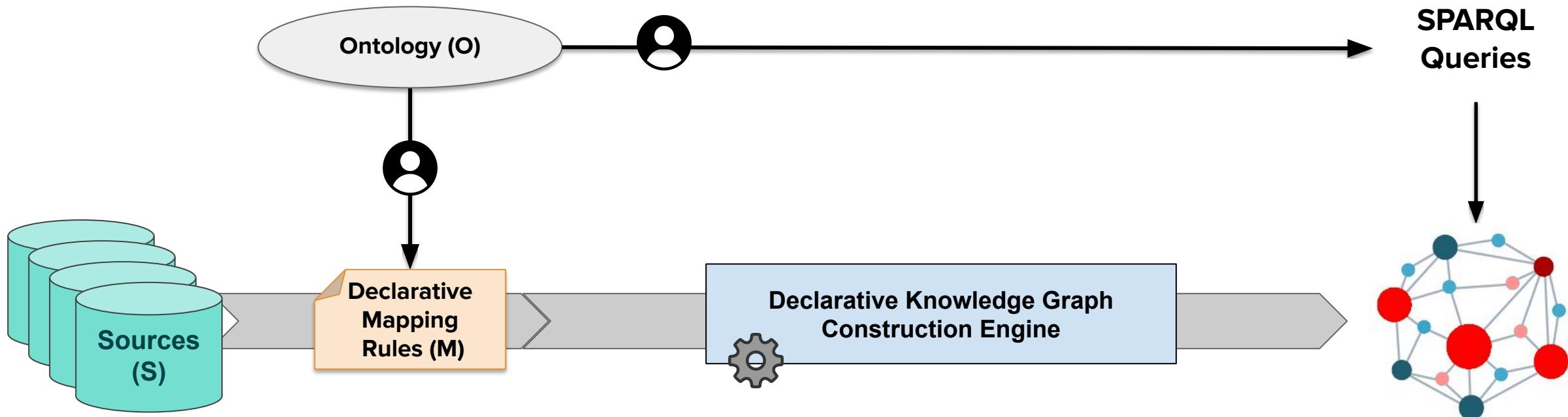
## **The RDF Mapping Language**

### **(or how to transform raw data into RDF)**

# Knowledge Graph Construction: Scripting-based



# KG Construction with Mapping Rules



Efficient



Scalable



Maintainable



Robust



Reproducible



## Example Input CSV files

**EMP.csv**

EMPNO	ENAME	JOB	DEPTNO
7369	SMITH	CLERK	10

**DEPT.csv**

DEPTNO	DNAME	LOC
10	APP SERVER	NEW YORK

## Desired RDF Output

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.  
<http://data.example.com/employee/7369> ex:name "SMITH".  
<http://data.example.com/employee/7369> ex:department <http://data.example.com/department/10>.  
  
<http://data.example.com/department/10> rdf:type ex:Department.  
<http://data.example.com/department/10> ex:name "APP SERVER".  
<http://data.example.com/department/10> ex:location "NEW YORK".
```

```
prefixes:  
rr: <http://www.w3.org/ns/r2rml#>.  
ex: <http://example.com/ns#>.  
rml: <http://semweb.mmlab.be/ns/rml#>.
```



```
mappings:
```

```
  TripplesMap1:  
    sources:  
    s:  
    po:
```

```
TripplesMap2:  
  sources:
```

```
  s:  
  po:
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.  
@prefix ex: <http://example.com/ns#>.  
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
```

```
<#TriplesMap1>
```

```
  rml:logicalSource [ ... ];  
  rr:subjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  ...  
  .
```

```
<#TriplesMap2>
```

```
  rml:logicalSource [ ... ];  
  rr:subjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  ...  
  .
```



```
prefixes:  
rr: <http://www.w3.org/ns/r2rml#>.  
ex: <http://example.com/ns#>.  
rml: <http://semweb.mmlab.be/ns/rml#>.
```



```
mappings:  
TriplesMap1:  
  sources:  
    - [EMP.csv~csv]  
  s:  
  po:
```

```
TriplesMap2:  
  sources:  
    - [DEPT.csv~csv]  
  s:  
  po:
```

```
<#TriplesMap1>  
  rml:logicalSource [  
    rml:source "EMP.csv";  
    rml:referenceFormulation ql:CSV ];
```

```
  rr:subjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  ...  
  .
```

```
<#TriplesMap2>  
  rml:logicalSource [  
    rml:source "DEPT.csv";  
    rml:referenceFormulation ql:CSV; ];
```

```
  rr:subjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  rr:predicateObjectMap [ ... ];  
  ...  
  .
```



```
prefixes:  
rr: <http://www.w3.org/ns/r2rml#>.  
ex: <http://example.com/ns#>.  
rml: <http://semweb.mmlab.be/ns/rml#>.
```



## mappings:

```
TriplesMap1:  
  sources:  
    - [EMP.csv~csv]  
  s:  
  po:  
    - [a, ex:Employee]
```

## TriplesMap2:

```
sources:  
  - [DEPT.csv~csv]  
s:  
po:  
  - [rdf:type, ex:Department]
```

```
<#TriplesMap1>  
  rml:logicalSource [  
    rml:source "EMP.csv";  
    rml:referenceFormulation ql:CSV ];  
  rr:subjectMap [  
    rr:class ex:Employee ];
```

```
    rr:predicateObjectMap [ ... ];  
    rr:predicateObjectMap [ ... ];  
    ...  
  .
```

```
<#TriplesMap2>  
  rml:logicalSource [  
    rml:source "DEPT.csv";  
    rml:referenceFormulation ql:CSV; ];  
  rr:subjectMap [  
    rr:class ex:Department ];
```

```
    rr:predicateObjectMap [ ... ];  
    rr:predicateObjectMap [ ... ];  
    ...  
  .
```





```
prefixes:  
rr: <http://www.w3.org/ns/r2rml#>.  
ex: <http://example.com/ns#>.  
rml: <http://semweb.mmlab.be/ns/rml#>.
```



```
mappings:  
TriplesMap1:  
sources:  
- [EMP.csv~csv]  
s: http://data.example.com/employee/ $(EMPNO)  
po:  
- [a, ex:Employee]
```

```
TriplesMap2:  
sources:  
- [DEPT.csv~csv]  
s: http://data.example.com/department/ $(DEPTNO)  
po:  
- [rdf:type, ex:Department]
```

```
<#TriplesMap1>  
rml:logicalSource [ .... ];  
rr:subjectMap [  
rr:template "http://data.example.com/employee/{EMPNO}";  
rr:termType rr:IRI; rr:class ex:Employee  
];  
rr:predicateObjectMap [  
rr:predicateMap [ ... ] rr:objectMap [ ... ] ];
```



```
<#TriplesMap2>  
rml:logicalSource [ .... ];  
rr:subjectMap [  
rr:template "http://data.example.com/department/{DEPTNO}";  
rr:termType rr:IRI; rr:class ex:Department  
];  
rr:predicateObjectMap [  
rr:predicateMap [ ... ] rr:objectMap [ ... ] ];
```





```
prefixes:
rr: <http://www.w3.org/ns/r2rml#>.
ex: <http://example.com/ns#>.
rml: <http://semweb.mmlab.be/ns/rml#>.
```

```
mappings:
TriplesMap1:
sources:
- [EMP.csv~csv]
s: http://data.example.com/employee/${EMPNO}
po:
- [a, ex:Employee]
```

```
TriplesMap2:
sources:
- [DEPT.csv~csv]
s: http://data.example.com/department/${DEPTNO}
po:
- [rdf:type, ex:Department]
- [ex:name, ${DNAME}]
- [ex:location, http://locations.org/\${LOC}]
```



```
<#TriplesMap2>
rr:logicalTable [ rr:tableName "DEPT" ];
rr:subjectMap [
  rr:template "http://data.example.com/employee/{DEPTNO}";
  rr:termType rr:IRI; rr:class ex:Department;
];
rr:predicateObjectMap [
  rr:predicateMap [ rr:constant ex:name;
    rr:termType rr:IRI ]
  rr:objectMap [ rr:reference "DNAME";
    rr:termType rr:Literal ]
];
rr:predicateObjectMap [
  rr:predicateMap [ rr:constant ex:location;
    rr:termType rr:IRI
  ];
  rr:objectMap [
    rr:template "http://locations.org/{LOC}";
    rr:termType rr:Literal ]
];
.
```





```
prefixes:
rr: <http://www.w3.org/ns/r2rml#>.
ex: <http://example.com/ns#>.
rml: <http://semweb.mmlab.be/ns/rml#>.
```



```
mappings:
TriplesMap1:
  sources:
    - [EMP.csv~csv]
    s: http://data.example.com/employee/${EMPNO}
  po:
    - [a, ex:Employee]
    - p: ex:department
      o:
```

**- mapping: TriplesMap2**

condition:

function: equal

parameters:

- [str1, \$(DEPTID)]
- [str2, \$(DEPTNO)]

TriplesMap2:

sources:

- [DEPT.csv~csv]

s: http://data.example.com/department/\${DEPTNO}

po:

- [rdf:type, ex:Department]

```
<#TriplesMap1>
rr:logicalSource [...];
rr:subjectMap [
  rr:template "http://data.example.com/employee/{EMPNO}";
  rr:termType rr:IRI; rr:class ex:Employee
];
rr:predicateObjectMap [
  rr:predicateMap [
    rr:constant ex:department; rr:termType rr:IRI
  ]
  rr:objectMap [
    rr:parentTriplesMap <#TriplesMap2>;
    rr:joinCondition [
      rr:child "DEPTID"; rr:parent "DEPTNO";
    ];
  ]
];

```



<#TriplesMap2>

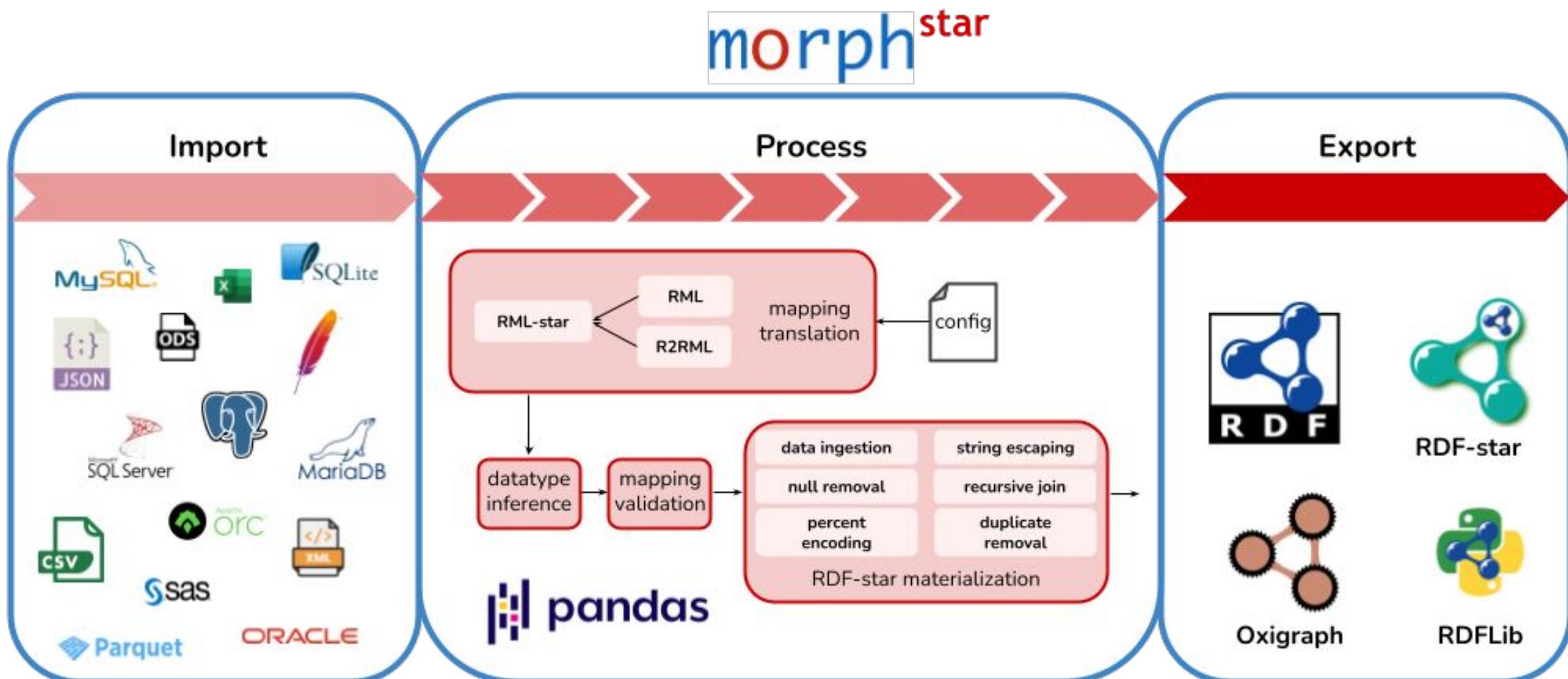
```
rr:logicalSource [ ... ];
rr:subjectMap [
  rr:template "http://data.example.com/department/{DEPTNO}";
  rr:termType rr:IRI; rr:class ex:Department
];
rr:predicateObjectMap [
  rr:predicateMap [ ... ] rr:objectMap [ ... ];
]
```



```
prefixes:  
rr: <http://www.w3.org/ns/r2rml#>.  
ex: <http://example.com/ns#>.  
rml: <http://semweb.mmlab.be/ns/rml#>.  
  
mappings:  
TriplesMap1:  
    sources:  
        - [EMP.csv~csv]  
    s: http://data.example.com/employee/${EMPNO}  
    po:  
        - [a, ex:Employee]  
        - p: ex:department  
        o:  
            - mapping: TriplesMap2  
    condition:  
        function: equal  
    parameters:  
        - [str1, ${DEPTID}]  
        - [str2, ${DEPTNO}]
```

```
TriplesMap2:  
    sources:  
        - [DEPT.csv~csv]  
    s: http://data.example.com/department/${DEPTNO}  
    po:  
        - [rdf:type, ex:Department]  
        - [ex:name, ${DNAME}]  
        - [ex:location, http://locations.org/${LOC}]
```





# morph



<https://github.com/oeg-upm/morph-kgc/>



<https://pypi.org/project/morph-kgc/>



<https://morph-kgc.readthedocs.io/>



<https://short.upm.es/umdvm>



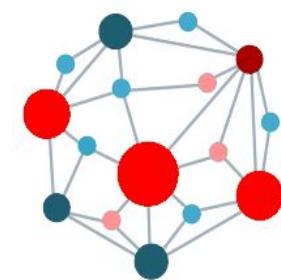
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) Hands-On 1: Querying Wikidata



## Hands-On 2: Constructing and validating your RDF graph (50min)

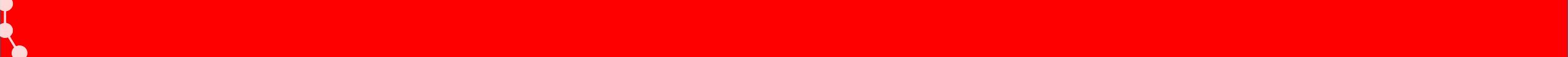
- 1) Tools for Developing Ontologies and KGs
- 2) RML (from CSV/XML/JSON to RDF)
- 3) **SHACL (RDF validation)**



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project





# **SHACL**

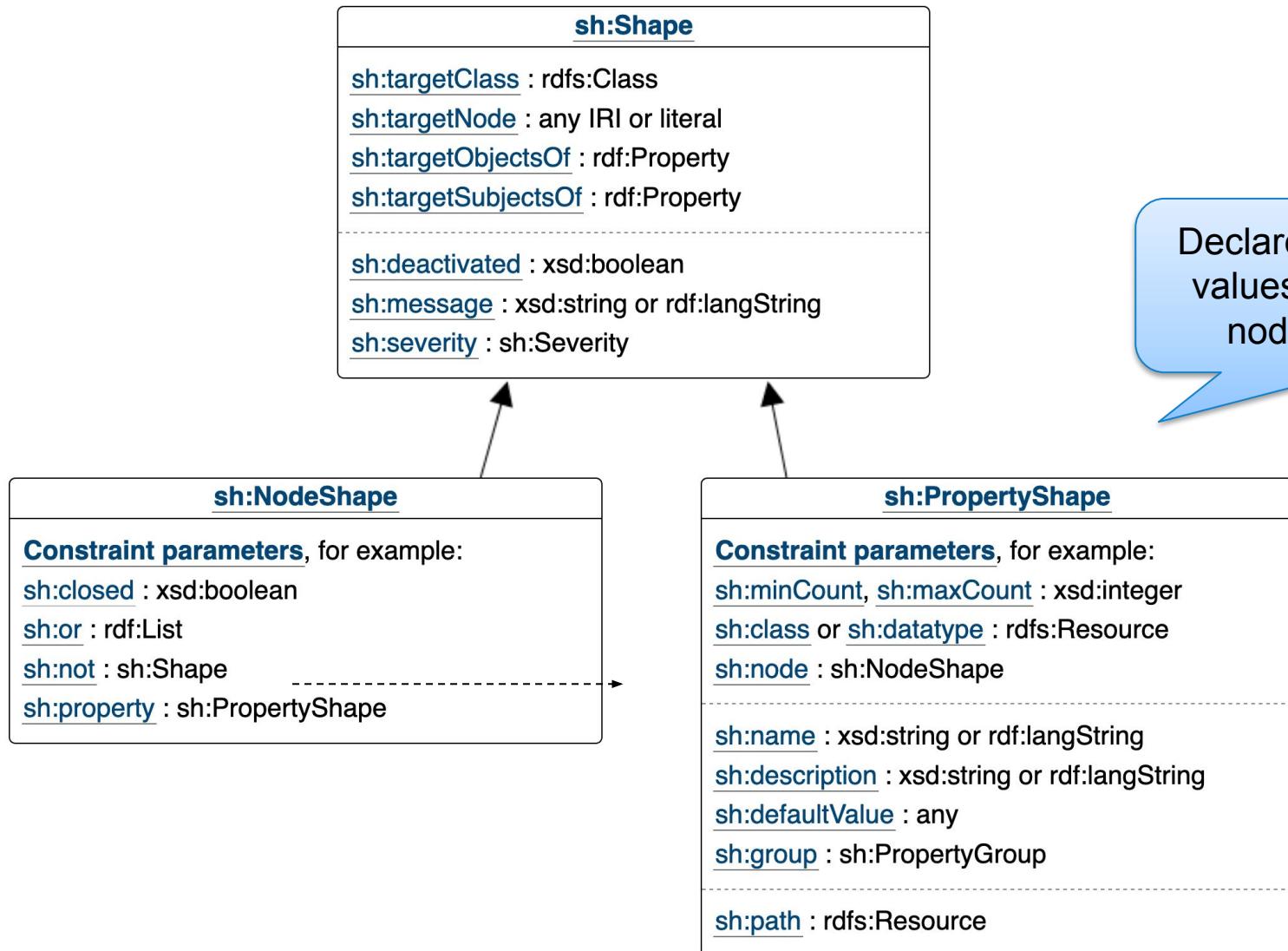
## **Shapes Constraint Language**

**(or how to validate your RDF data against constraints)**

Playground: <https://shacl-playground.zazuko.com/>

sh: <http://www.w3.org/ns/shacl#>

Declare constraints directly on a node



Declare constraints on the values associated with a node through a path

- Specify constraints that need to be met with respect to focus nodes
- *sh:property* associates a shape with a property shape

```
ex:PersonShape
  a sh:NodeShape ;
  sh:targetClass ex:Person ;
  sh:property ex:PersonShape-name .
```

## Data graph

```
1867 schema:CivicStructure  
9 schema:Museum  
0 schema:CollegeOrUniversity
```

## Shapes graph

```
ex:CivicStructureShape  
a sh:NodeShape ;  
sh:targetClass schema:CivicStructure .
```

```
ex:MuseumShape  
a sh:NodeShape ;  
sh:targetClass schema:Museum .
```

```
ex:UniversityShape  
a sh:NodeShape ;  
sh:targetClass schema:CollegeOrUniversity .
```

## Validation report

```
[  
a sh:ValidationReport ;  
sh:conforms true  
].
```





- Specify constraints that need to be met with respect to nodes that can be reached from the focus node
- Reachable nodes are specified using *sh:path* either with a property IRI or a SHACL property path

```
ex:PersonShape-name
  a sh:PropertyShape ;
    sh:path ex:name ;
    sh:minCount 1.
```

```
ex:CivicStructureShape
  a sh:NodeShape ;
  sh:targetClass schema:CivicStructure ;
  sh:property ex:UPRNShape .

ex:MuseumShape
  a sh:NodeShape ;
  sh:targetClass schema:Museum ;
  sh:property ex:UPRNShape .

ex:UniversityShape
  a sh:NodeShape ;
  sh:targetClass schema:CollegeOrUniversity ;
  sh:property ex:UPRNShape .

ex:UPRNShape
  a sh:PropertyShape ;
  sh:path ec:uprn .
```

- Associated with (node or property) shapes to declare constraints
- A shape may have several constraint components
- A constraint component may have multiple values (all constraints apply)
- Constraint components:
  - Value type
  - Cardinality
  - Value range
  - String-based
  - Language-based
  - Property pair
  - Logical
  - Shape-based
  - Closed shapes

- Specify the set of values that a node can have

Property	Description
sh:datatype	Specifies that values must be literals with some datatype
sh:class	Specifies that values must be SHACL instances of some class
sh:nodeKind	Possible values: sh:BlankNode, sh:IRI, sh:Literal, sh:BlankNodeOrIRI, sh:BlankNodeOrLiteral, sh:IRIOrLiteral
sh:in	Enumerates the value nodes that a property is allowed to have
sh:hasValue	A node must have a given value



- Specify restrictions on the minimum and maximum number of distinct value nodes
- The default cardinality for property shapes is {0,unbounded}

Property	Description
sh:minCount	Restricts minimum number of value nodes. If not defined, there is no restriction (no minimum)
sh:maxCount	Restricts maximum number of value nodes. If not defined, there is no restriction (unbounded)

- Specify conditions on the string representation of value nodes

Property	Description
sh:minLength	The minimum string length; if 0, there is no length restriction but fails with blank nodes
sh:maxLength	The maximum string length
sh:pattern	Regular expression that must be matched (SPARQL REGEX)
sh:flags	Optional string of regular expression flags
sh:languageIn	Allowed language tags
sh:uniqueLang	No pair of value nodes may use the same language tag



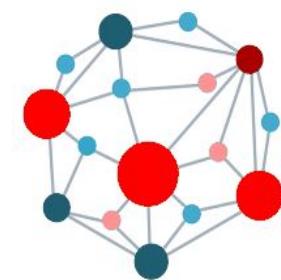
## An introduction to the Semantic Web (1,5h)

- 1) What is a KG? Why KGs now?
- 2) RDF, OWL and SPARQL
- 3) RDF APIs and Triplestores
- 4) Hands-On 1: Querying Wikidata



## Hands-On 2: Constructing and validating your RDF graph (50min)

- 1) Tools for Developing Ontologies and KGs
- 2) RML (from CSV/XML/JSON to RDF)
- 3) SHACL (RDF validation)



## Hands-On 3: Examples of Exploiting KGs for AI (30 min)

- 1) Entity recognition with KGs
- 2) RDF generation from text with OpenAI
- 3) Semantic Table Interpretation
- 4) The Drugs4Covid Project



## Semantic Answer Type, Entity, and Relation Linking Task (SMART2022) 3rd

Organizer: Nandana Mihindukulasooriya, Mohnish Dubey, Alfio Gliozzo, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Ricardo Usbeck, Gaetano Rossiello and Debayan Banerjee

The challenge, SMART 2022, consists of three tasks Semantic Answer Type prediction task, Entity and Relation Linking tasks. Given a question in natural language, the three tasks in this challenge are to predict the answer type for a given question, identify entities and relations in the questions using a target ontology (e.g., DBpedia or Wikidata). This is a continuation of SMART2020, and SMART2021 challenges.

Website: <https://smart-task.github.io/>



<https://www.dbpedia-spotlight.org/>



<https://labs.tib.eu/falcon/falcon2/>



## Knowledge Base Construction from Pre-trained Language Models (LM-KBC) NEW

Organizer: Sneha Singhania, Tuan-Phong Nguyen, Simon Razniewski

Pre-trained language models (LMs) have advanced a range of semantic tasks and have also shown promise for knowledge extraction from the models itself. Although several works have explored this ability in a setting called probing or prompting, the viability of knowledge base construction from LMs has not yet been explored. In this challenge, participants are asked to build actual knowledge bases from LMs, for given subjects and relations. In crucial difference to existing probing benchmarks like LAMA (Petroni et al., 2019), we make no simplifying assumptions on relation cardinalities, i.e., a subject-entity can stand in relation with zero, one, or many object-entities. Furthermore, submissions need to go beyond just ranking the predictions, and materialize outputs, which are evaluated by established KB metrics of precision and recall. The challenge comes with two tracks: (i) a BERT-type LM track with low computational requirements, and (ii) an open track, where participants can use any LM of their choice.

Website: <http://lm-kbc.github.io>



## Knowledge Graph Generation from Text

## Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2022) • 4th

Organizer: Nora Abdelmageed, Jiaoyan Chen, Vincenzo Cutrona, Vasilis Efthymiou, Oktie Hassanzadeh, Madelon Hulsebos, Ernesto Jimenez-Ruiz, Juan F. Sequeda, Kavitha Srinivas

This challenge aims at benchmarking systems dealing with the tabular data to KG matching problem, so as to facilitate their comparison on the same basis and the reproducibility of the results.

Website: <https://www.cs.ox.ac.uk/isg/challenges/sem-tab/>



<https://dbpedia.mtab.app/mtab>

We created a knowledge graph with evidence from scientific publications on coronaviruses. Drugs, diseases, genes and proteins are identified and normalised to enable complex searches on the articles.



Drugs4Covid extracts and structures evidence from scientific papers in five steps: Harvest, Parse, Extract, Integrate, and Publish. Try out our public tools.

<https://drugs4covid.oeg.fi.upm.es/>

<https://drugs4covid.oeg.fi.upm.es/services/bio-qa>



## RDF2vec.org

The hitchhiker's guide to RDF2vec.

### About RDF2vec

RDF2vec is a tool for creating vector representations of RDF graphs. In essence, RDF2vec creates a numeric vector for each node in an RDF graph.

<http://rdf2vec.org/>



# Knowledge Graphs and Ontologies

## A Practical Introduction to the Semantic Web

David Chaves-Fraga, Ontology Engineering Group  
Universidad Politécnica de Madrid, Spain

