

d-wise

Empowering innovation for a healthier world



STRATEGY



DATA



CLOUD



PRIVACY

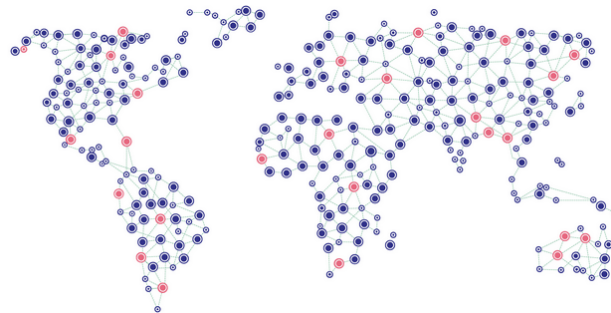


ANALYTICS

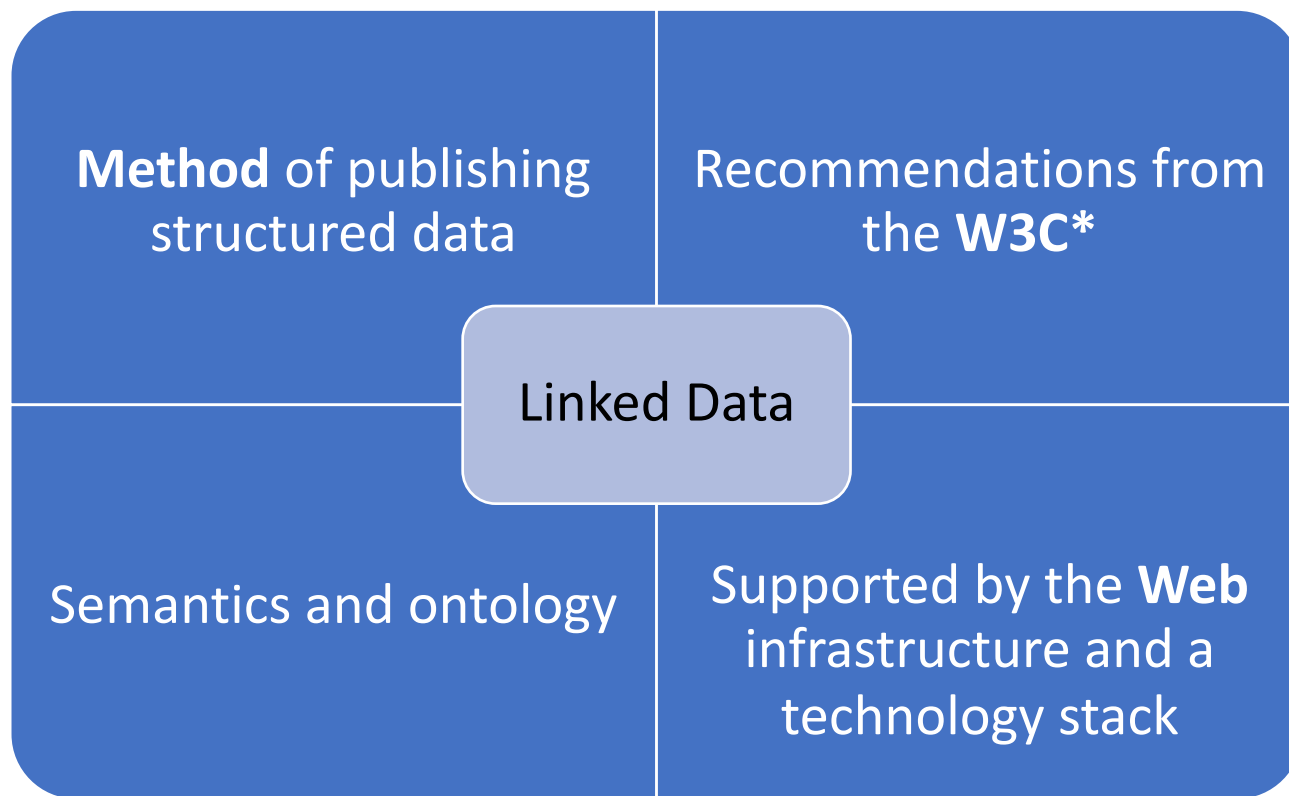
Linked Data

Nicolas Dupuis, d-wise

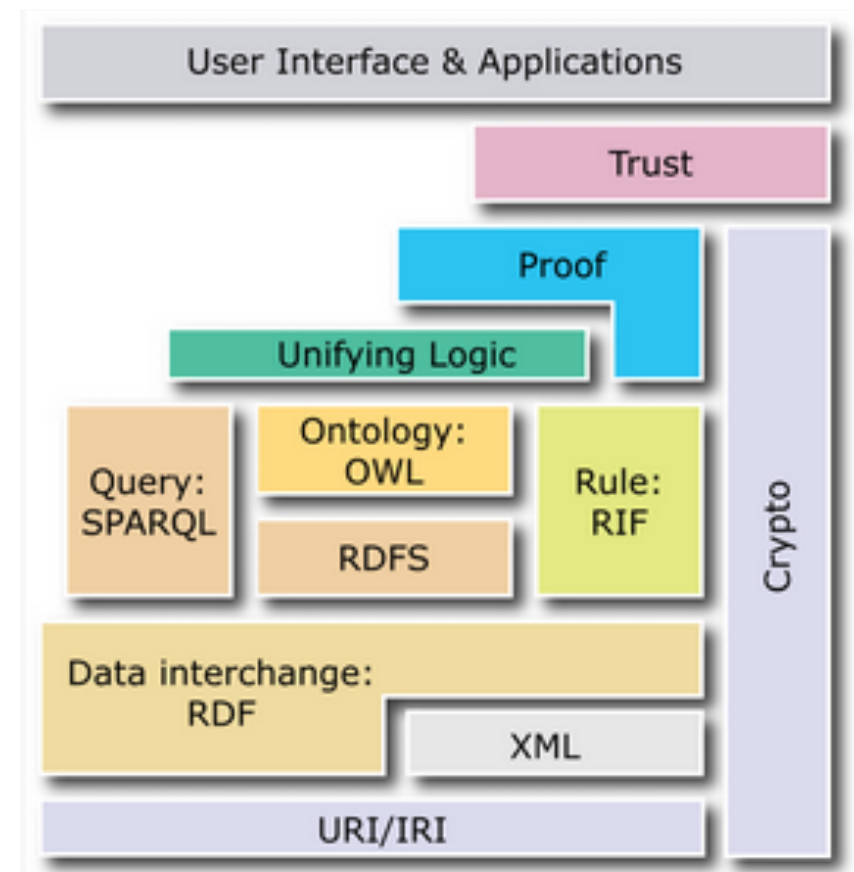
November 26, 2019



Sir Berners-Lee



* Consortium World Wide Web



Rectangular data: the shortcomings

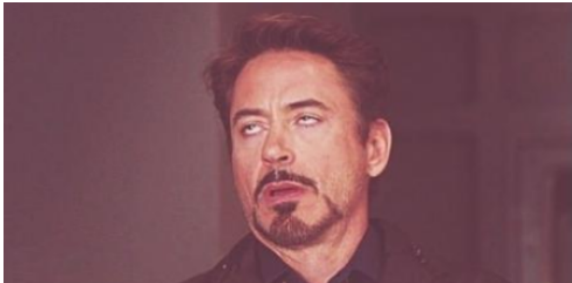
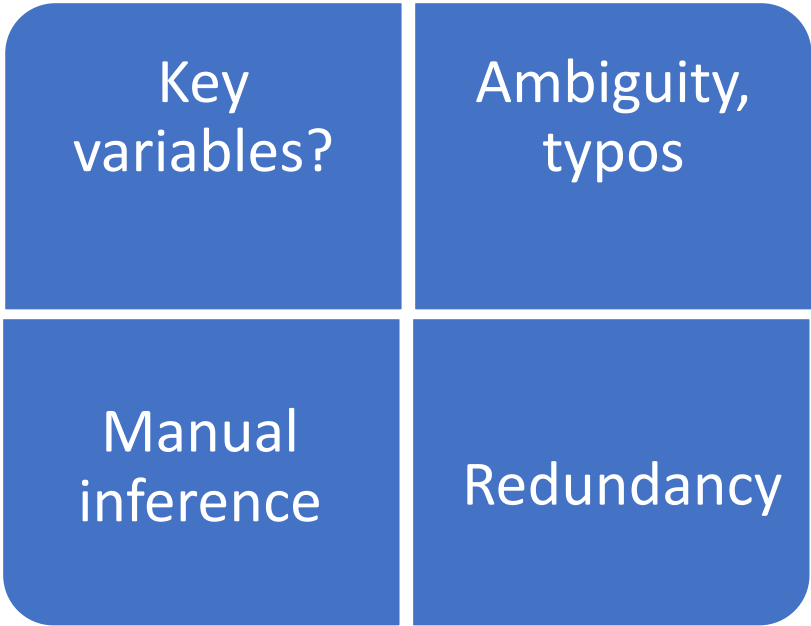
Table A

Name	Spouse	Secrete_Identity
Clark Kent	Lois	Superman
Peter Parker	Mary-Jane	Spyderman



Table B

Name	Activity	DOB
L. Lane	Journalist	1937
MJ. Watson	Model	1965
MJ. Watson	Actress	1965
C. Kent	Journalist	1938
P. Parker	Photographer	1963

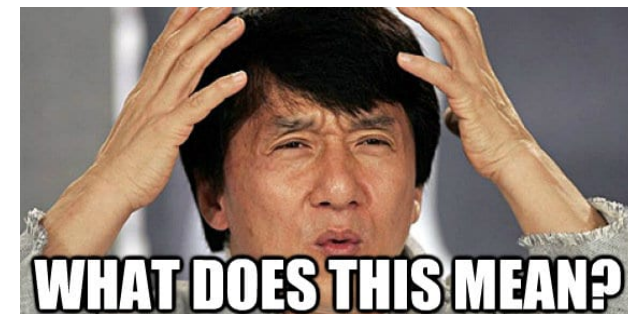


My goal:



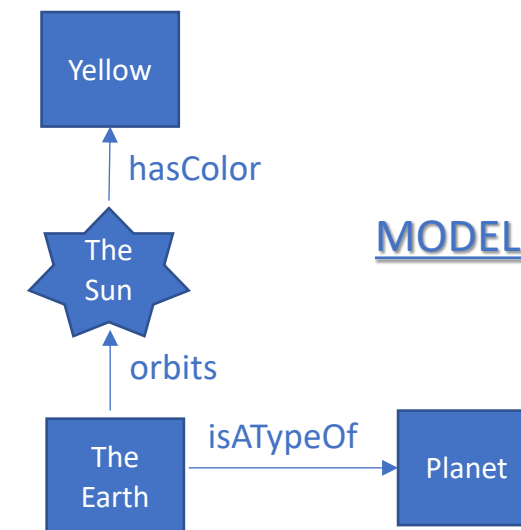
Also, use Internet [memes](#) 😊

Semantics?



- [Semantics](#) is the linguistic study of meaning, i.e. the relationship between a word and what it stands for
- [RDF](#) (Resource Data Framework) is the W3C standard **data model** to make statements about things, to model knowledge
- These statements are known as **triples**:

Subject	Predicate (property name)	Object (property value)
The Sun	hasColor	Yellow
The Earth	isATypeOf	Planet
The Earth	orbits	The Sun



RDF serialization

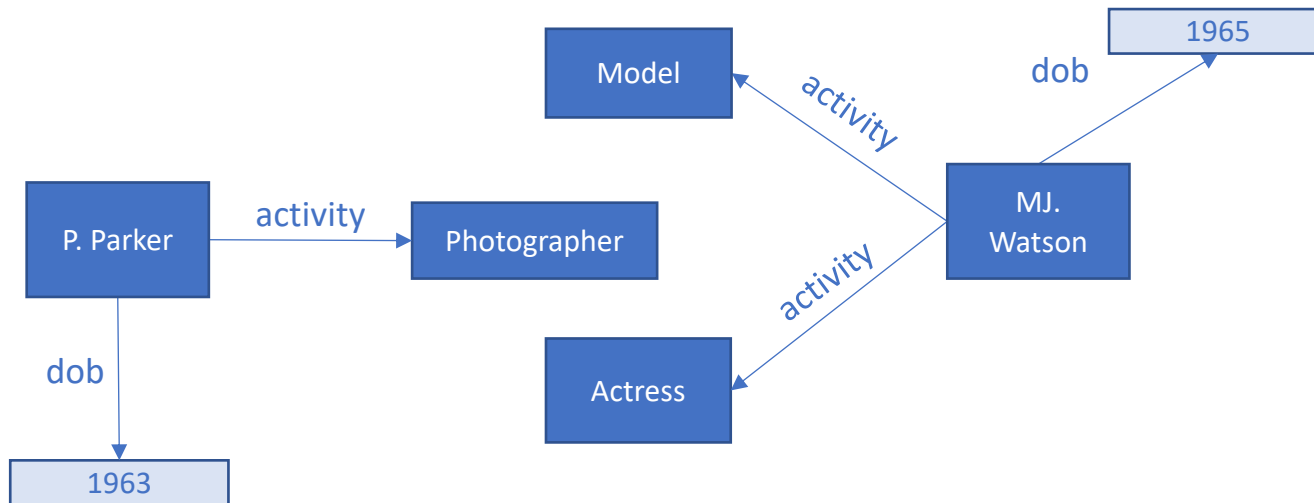
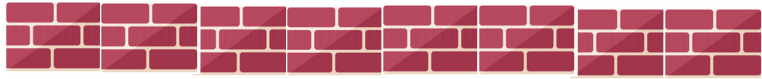
- RDF is an abstract model, the information itself can be stored in a text file using a **serialization format**.
- [Turtle](#) (Terse RDF Triple language) is [published](#) by the W3C

	in Turtle format:
A statement	<code>green-goblin enemyOf spiderman .</code>
A list of predicates	<code>green-goblin enemyOf spiderman ; type Person ; name "Green Goblin" .</code>
A list of objects	<code>spiderman name "Spiderman"@en , "L'homme araignée"@fr .</code>



Linking tables A and B – Attempt #1

```
"Peter Parker" Spouse "MJ" ;  
secrete_ID "Spiderman" .
```



```
"P. Parker" activity "Photographer" ;  
dob "1963" .  
"MJ Watson" activity "Model" ,  
"Actress" ;
```

FAIL

No auto-merge

FAIL

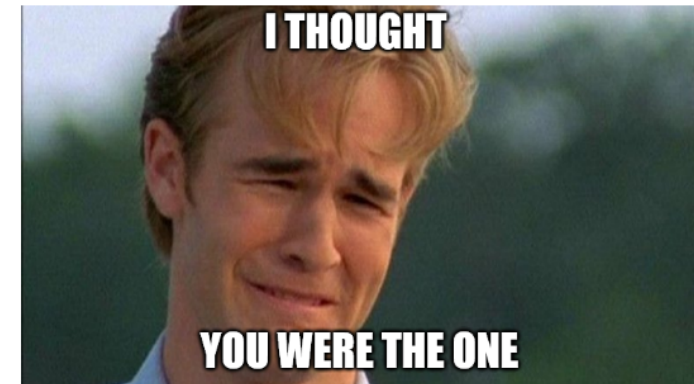
Still ambiguous

FAIL

No inference



RDF is just a data model

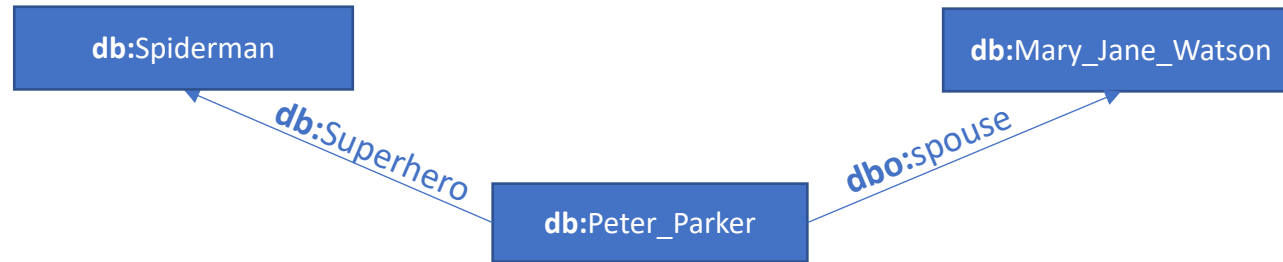


Uniform Resource Identifier

- A [URI](#) is a unique string of characters that unambiguously identifies a particular resource.
- The most common form of URI is the Uniform Resource Locator ([URL](#)). All URL are URI.
- Linked Data recommendations:
 - define things with a URI,
 - the URI should be a URL,
 - the URL should have browsable content.



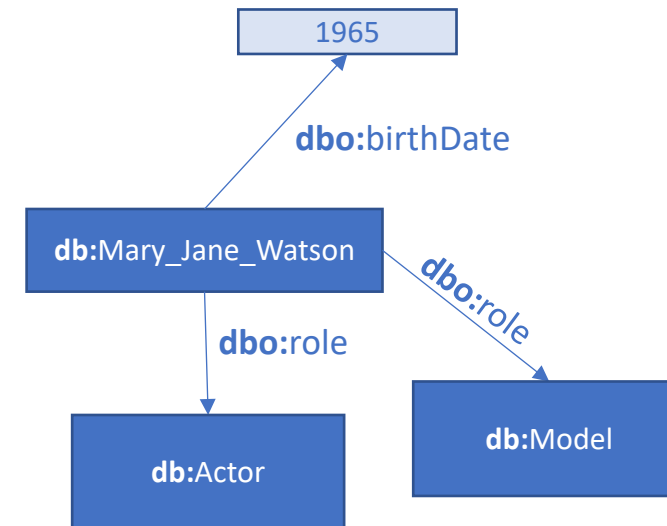
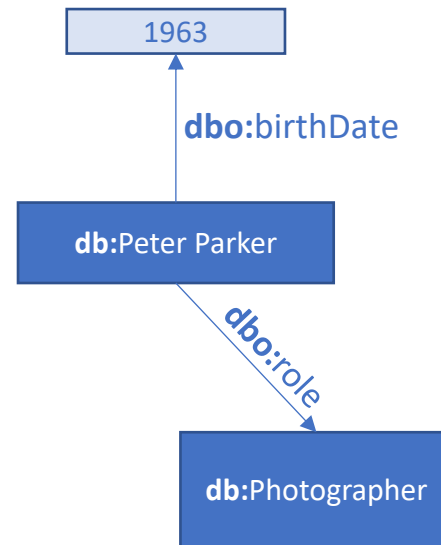
Linking tables A and B – Attempt #2



Effortless merge



Unambiguous



qname	namespace
db	http://dbpedia.org/page/
dbo	http://dbpedia.org/ontology/



Graph database

SPARQL is the RDF query language published by the W3C



SPARQL query

Result

```
PREFIX dbo: <http://dbpedia.org/page/>
SELECT ?subject ?job
WHERE {?subject dbo:role ?job .}
```

subject	job
Clark_Kent	Journalist
Mary_Jane_Watson	Model
etc...	

People and their jobs

```
SELECT ?subject
WHERE {?subject dbo:role db:Actor
      ?subject dbo:role db:Model .}
```

subject
Mary_Jane_Watson

*People who are Model
and Actor*

```
SELECT ?subject ?spouse
WHERE {?subject dbo:role db:Journalist .
      OPTIONAL {?subject dbo:spouse ?spouse .}}
```

subject	spouse
Clark_Kent	Lois_Lane
Lois_Lane	

*Journalists and their
marital status (if any)*

```
SELECT ?Journalists ?dob
WHERE {?Journalists dbo:role db:Journalist .
      ?Journalists dbo:birthDate ?dob .
      FILTER (?dob > "1937") }
```

Journalists	dob
Clark_Kent	1938

*Journalists born after
1937*



SPARQL query

Result

```
CONSTRUCT {?object dbo:spouse ?subject}
WHERE {?subject dbo:spouse ?object .}
```

Subject	Predicate	Object
Lois_Lane	dbo:spouse	Clark_Kent
Mary_Jane_Watson	dbo:spouse	Peter_Parker

Spouse's spouses.

```
SELECT ?s
WHERE {?s dbo:birthDate ?dob.}
ORDER BY ?dob
LIMIT 1
```

s

Lois_Lane

Oldest

```
SELECT ?s
WHERE {?s dbo:role db:Journalist .
      FILTER NOT EXISTS {?s dbo:spouse ?o } }
```

s

Lois_Lane

Journalist who are single

```
SELECT (COUNT (?subject) as ?howMany)
WHERE {?subject dbo:role db:Journalist . }
```

howMany

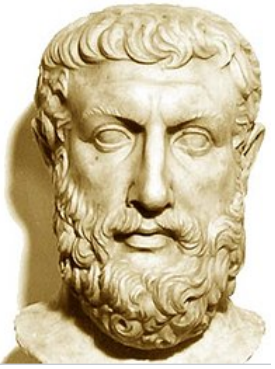
2

How many journalists

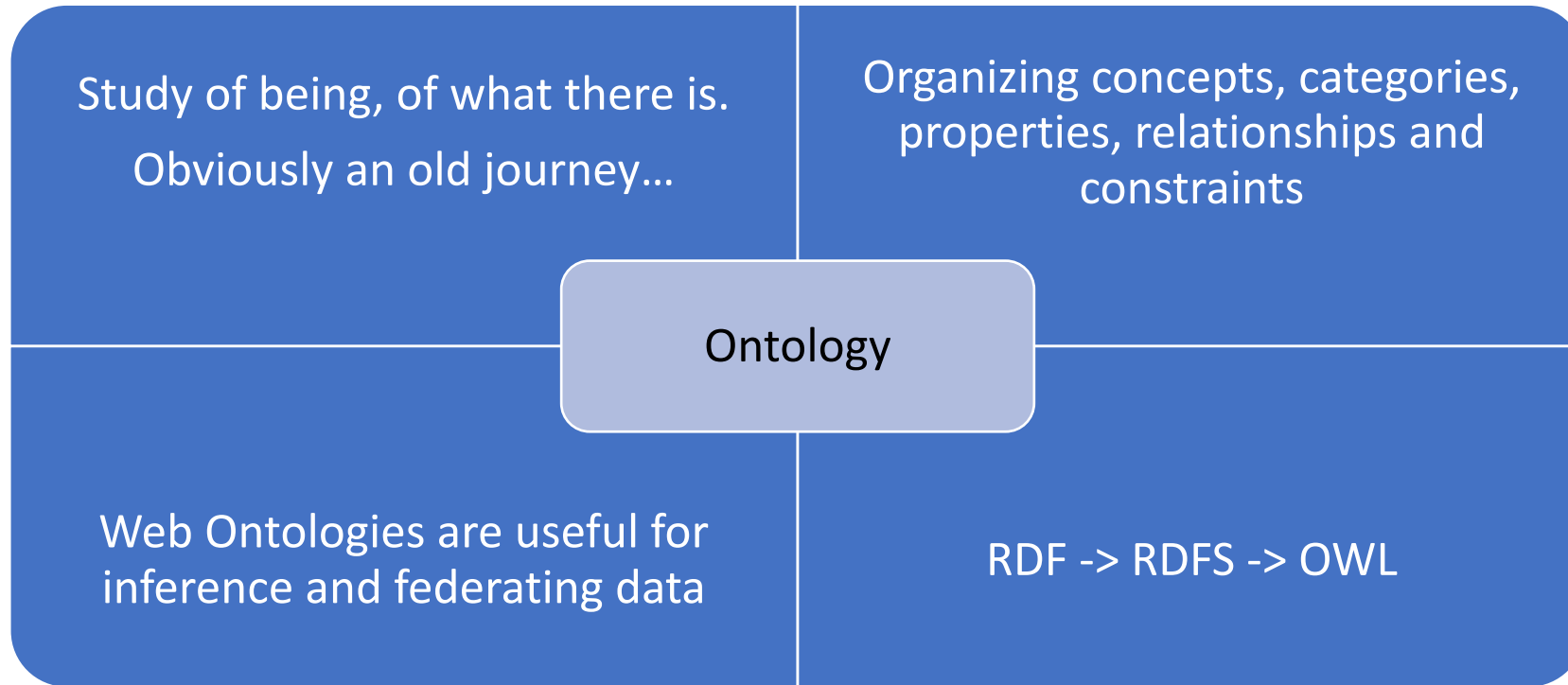
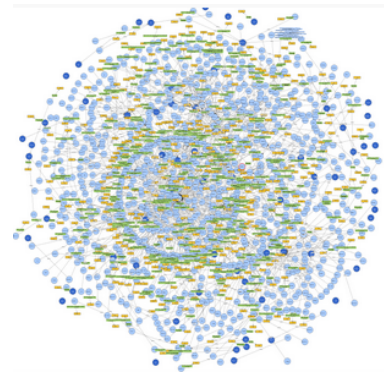
```
SELECT ?s (COUNT (?job) as ?jobs)
WHERE {?s dbo:role ?job . }
GROUP BY ?s
HAVING (?jobs > 1)
```

s	jobs
Mary_Jane_Watson	2
Lois_Lane	1

How many jobs per person



Ontologies



Web philosophy: “Anyone can say Anything about Anything” (AAA)

Types of Headaches

Migraine



Hypertension



Stress



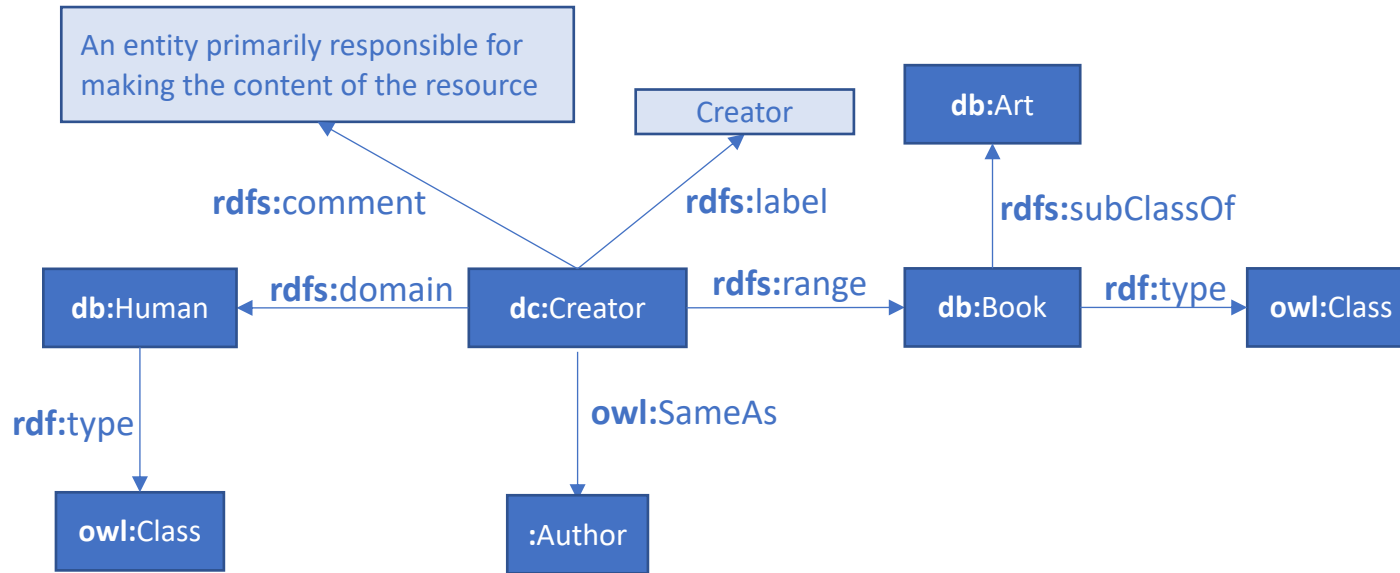
OWL



RDFS and OWL

- **RDFS** and **OWL** provide modeling tools (= constructs) for knowledge description & discovery, to author ontologies
- OWL (from W3C) builds on RDFS and comes with more subtle constructs and finer-grained modeling.
- Constructs have **formal semantics** and are best used for inference and federation (AAA !)

ONTOLOGY



FORMAL SEMANTICS

`rdfs:subClassOf`

```
CONSTRUCT {?s rdfs:type ?c2}
SELECT {?c1 rdfs:subClassOf ?c2 .
       ?s rdfs:type ?c1 }
```

`rdfs:domain`

```
CONSTRUCT {?s rdfs:type ?domain}
WHERE {?prop rdfs:domain ?domain .
       ?s ?prop ?o .}
```

`rdfs:range`

```
CONSTRUCT {?o rdfs:type ?range}
WHERE {?prop rdfs:range ?range .
       ?s ?prop ?o .}
```

`owl:SameAs`

```
CONSTRUCT {?s2 ?p ?o}
SELECT {?s owl:sameAs ?s2 .
       ?s ?p ?o .}
```

(and same for p and o)

ASSERTED INDIVIDUALS (aka data)

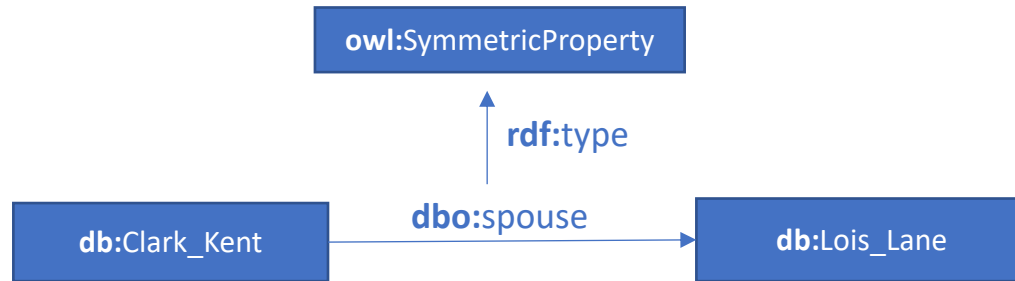
```
db:Stan_Lee dc:creator ISBN:978-1524763138
ISBN:978-2809480665 dc:title "Excelsior!"
```

INFERRED DATA

```
db:Stan_Lee rdfs:type db:Human .
ISBN:978-2809480665 rdfs:type db:Book .
ISBN:978-2809480665 rdfs:type db:Art .
```



Challenge #1 - Simple inference



```
SELECT ?s
WHERE {?s dbo:spouse ?o .}
```



Semantic
Reasoner

spouse
Clark_Kent

spouse
Clark_Kent
Lois_Lane

Asserted data

Inferred data

```
CONSTRUCT {?o ?prop ?s}
WHERE {?prop rdf:type owl:SymmetricProperty .
      ?s ?prop ?o .}
```

Challenge #2: Data federation



<http://www.dailyplanet.com/Lois/sparql>

```
:Clark :email "clarkkent@dailyplanet.com"  
:Clark :email "Ibelieve@IcanFly.com"  
:Clark foaf:name "Clark Kent"  
:Lois :likes :Clark
```



<http://www.dailyplanet.com/Jimmy/sparql>

```
:Superman :isLocated "Metropolis"  
:Superman :emailAddress "Ibelieve@IcanFly.com"
```



<http://www.dailyplanet.com/Perry/sparql>

```
:emailAdress owl:sameAs :email  
  
:email rdf:type owl:InverseFunctionalProperty
```



```
SELECT ?p ?o  
WHERE { :Superman ?p ?o. }
```

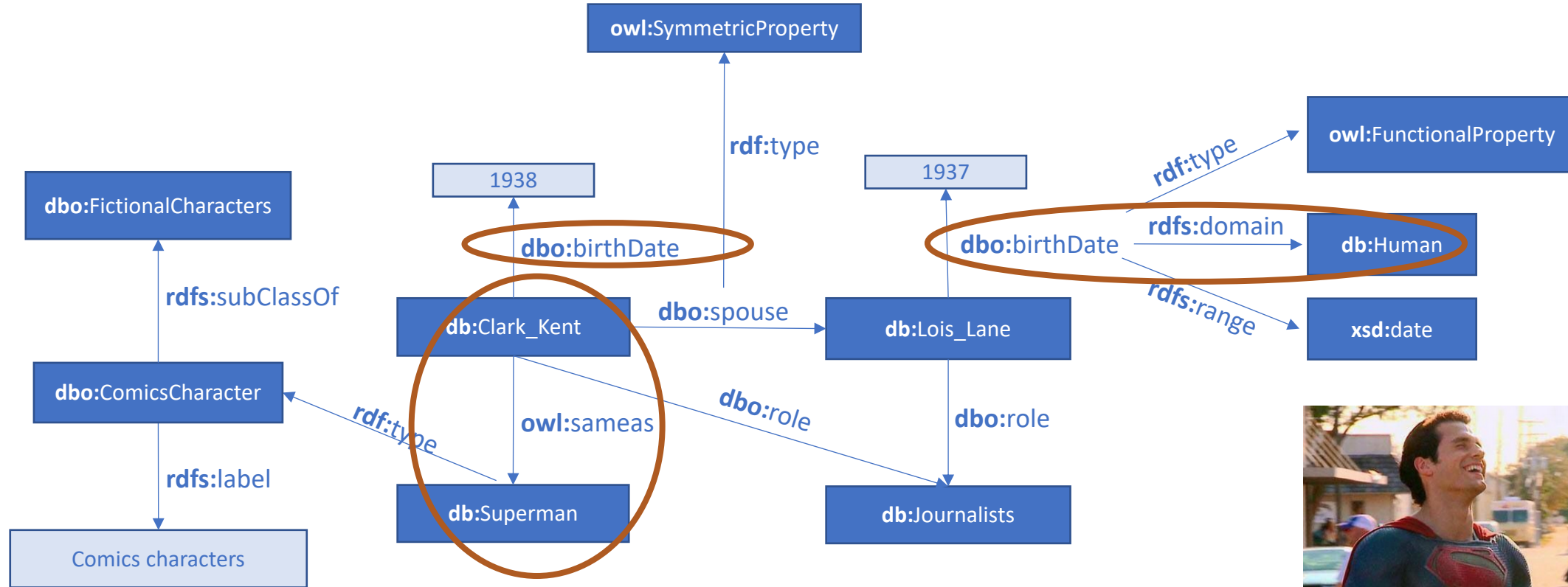


: Superman

p	o
:islocated	Metropolis
:emailAddress	Ibelieve@IcanFly.com
:email	Ibelieve@IcanFly.com
owl:sameAs	:Clark
foaf:name	Clark Kent
:email	clarkkent@dailyplanet.com

```
CONSTRUCT {?subject owl:sameAs ?subject2}  
WHERE {?prop rdf:type owl:InverseFunctionalProperty .  
       ?subject ?prop ?o .  
       ?subject2 ?prop ?o .}
```

Challenge #3: pushing it too far



`db:Superman` `rdf:type` `db:Human`





