

July 2, 2019



# KR Querying & Reasoning for Large Knowledge Graphs

## Part I

Sebastian Rudolph, TU Dresden

July 2, 2019



# KR Querying & Reasoning for Large Knowledge Graphs Part II

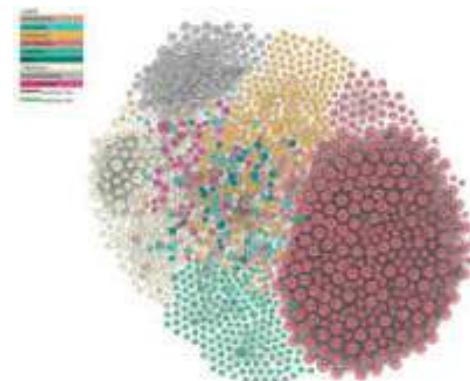
Axel Polleres, WU Vienna

# Are OWL and RDFS entailment enough?

- Determining Satisfiability and Consistency and Entailments in KGs is one thing...
- **But:**
  - Mostly you actually want to **retrieve** information from a KG
  - You also need to deal with contextualized information
  - Existing KGs aren't consistent 😞



Vs.



# Mostly you actually want to retrieve information from a KG

- E.g. from 

**London**

This article discusses the historical, Royal city of London, now Greater London. For the modern city and ceremonial county, see [Greater London](#). For the county, see [London \(metropolitan county\)](#).

**London** is a city and metropolis in England, the capital of England and the United Kingdom. It is a major port and the second-most populous urban area in the European Union after Paris, with an urban population of 8.2 million. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million. London is one of the world's leading global cities. It is a major cultural, political, and financial centre, and the largest city in the European Union by most measures.

<https://en.wikipedia.org/wiki/London>

**Automatic Extractors**



- One of the central datasets of the Linked Open Data-Cloud
- RDF extracted from Wikipedia-Infoboxes
- SPARQL endpoint, e.g.:
  - „Cities in the UK with more than 1M population“:

**About: London**

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

Founded by L�ostrum, who named it Londinium, London is a city and county. It is the capital of London, largely reflected in its 172 square-mile (442 km<sup>2</sup>) metropolitan boundaries. As of 2011 this fifth-century foundation has also referred to the local council and, since 1965, its largely self-governing Metropolis, known as Greater London, which includes 32 districts. London is the most populous city in the European Union, with an urban population of 8.2 million, and the second-most populous city in the European Union by most measures, with a metropolitan area of over 13 million. London is one of the world's leading global cities, a major cultural, political and financial centre, and the largest city in the European Union by most measures.

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

Besides OWL, RDF, RDFS, we need query languages!

## Structured queries (SPARQL):

<http://yasgui.org/short/UV0yHx8ft>

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

```
SELECT DISTINCT ?city ?pop WHERE {
  ?city a yago:City108524735 .
  ?city dbo:country :United_Kingdom.
  ?city dbo:populationTotal ?pop
  FILTER ( ?pop > 1000000 ) }
```

# You also need to deal with contextualized information

- E.g. from 

## Rome

From Wikipedia, the free encyclopedia  
For other uses, see [Rome \(disambiguation\)](#).

**Rome** (Latin: *Roma*; Italian: *Roma*) is the capital city and a special comune of Italy (named "Comune di Roma Capitale"). Rome also serves as the capital of the Lazio region, with 2,870,000 residents in 1286 km<sup>2</sup> (800 ft<sup>2</sup>/mi<sup>2</sup>). It is also the country's most populous commune. It is the fourth most populous city in the European Union by population within city limits. It is the centre of the Metropolitan City of Rome, which has a population of 4,356,226 residents, thus making it the most populous metropolitan city in Italy. Rome is located in the central-western portion of the Italian Peninsula, within Lazio (Latium), along the shores of the Tiber. The Vatican City, the smallest country in the world,<sup>7</sup> is an

<https://en.wikipedia.org/wiki/Rome>

example of a country within a city; the reason Rome has been often defined as capital of new states.<sup>108</sup>

Rome's history spans 28 centuries. While Roman mythology states the founding of Rome at around 753 BC, the city has been inhabited for much longer, making it one of the oldest continuously inhabited sites in Europe.<sup>109</sup> The city's early

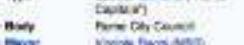
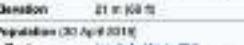
population originated from a mix of Latins, Etruscans, and Greeks. Eventually, the city successively became the capital of the Roman Kingdom, the Roman

Republic, and the Roman Empire, and is regarded by some as the first ever

metropolis.<sup>110</sup> It was first called the

Great City (Lat. *Urbs Aeterna; Italica Cittasempera*) by the Romans and

the City of the 1000 Hills (Lat. *Civitas*

Detailed info and images	
	
	
	
	
	

Automatic Extractors

„Cities in the **Italy** with more than 1M population“:

About: <b>Rome</b>	
<a href="http://dbpedia.org/resource/Rome">http://dbpedia.org/resource/Rome</a>	
<b>dbo:populationAsOf</b>	■ 2014 (xsd:integer)
<b>dbo:populationBlank</b>	■ 2869461 (xsd:integer) ■ 4321244 (xsd:integer)

Doesn't work!

Structured queries (SPARQL):

<http://yasgui.org/short/UVQyHx8ft>

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

SELECT DISTINCT ?city ?pop WHERE {
  ?city a yago:City108524735 .
  ?city dbo:country :Italy.
  ?city dbo:populationTotal ?pop
  FILTER ( ?pop > 1000000 )
}
```

# Existing KGs aren't consistent 😞 [1]

• E.g. 

**About: European Union**

An Entity of Type : populated place, from harvard Graph : http://dbpedia.org/within Data Space : dbpedia.org

The European Union (EU) is a political-economic union of 28 member states that are located primarily in Europe. It has an area of 4,324,782 km<sup>2</sup> (1,688,828 sq mi). As of 2017, the EU has a population of 510 million.

<b>rdf:type</b>	<ul style="list-style-type: none"> <li>■ <a href="#">owl:Thing</a></li> <li>■ <a href="#">dbo:Place</a></li> <li>■ <a href="#">dbo:Location</a></li> <li>■ <a href="#">wikidata:Q6256</a></li> <li>■ <a href="#">dbo:Country</a></li> <li>■ <a href="#">dbo:Organisation</a></li> <li>■ <a href="#">dbo:PopulatedPlace</a></li> <li>■ <a href="#">geo:SpatialThing</a></li> </ul>
-----------------	---

Dbpedia Ontology:

`dbo:Agent owl:disjointWith dbo:Place.`

 `dbo:Country rdfs:subClassOf dbo:Place.`  
`dbo:Organisation rdfs:subClassOf dbo:Agent.`

1. Stefan Bischof, Markus Krötzsch, Axel Polleres, and Sebastian Rudolph. Schema-agnostic query rewriting in SPARQL 1.1. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, Lecture Notes in Computer Science (LNCS). Springer, October 2014. [[.pdf](#)]

# Querying and Reasoning for KGs?

- Querying KGs with SPARQL (10min)
- Reasoning by Querying (Rewriting vs. Materialisation) (10min)
- Querying and Reasoning over Contextualised graph data (10min)
- Other issues: (15min)
  - Federation, Path Queries over Linked Data
  - SPARQL for KG Construction (scalability issues)
  - Updates

# Querying KGs with SPARQL

*Find some more detailed material to introduce SPARQL on my Webpage:  
<http://polleres.net/presentations/>*

# SPARQL

- “Just like SQL, only for matching patterns on (directed labelled) Graphs ...”

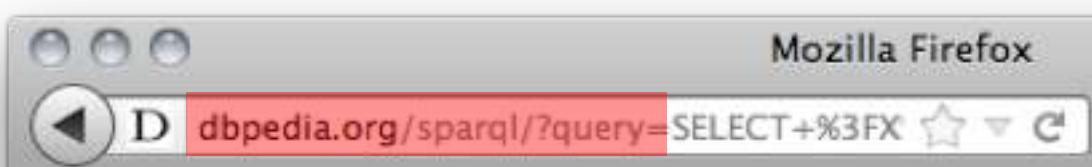
```

SELECT ?X
WHERE
{
  ?X <http://dbpedia.org/ontology/birthPlace> <http://dbpedia.org/resource/Bologna> .
}

```



- Standard protocol to access RDF data over the Web (SPARQL Protocol)



# Conjunction (.) , disjunction (UNION), optional (OPTIONAL) patterns and filters (FILTER)

*Names of scientists from places in Italy?*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
```

```
SELECT ?N
WHERE
{
  ?X a dbo:Scientist; foaf:name ?N ;
      dbo:birthPlace [ dbo:country dbr:Italy]
}
ORDER BY ?N
LIMIT 10
```

- Shortcuts for namespace prefixes and to group several triple patterns
- Slicing and dicing (ORDER BY, LIMIT/OFFSET ...)

N
1 "({Senator for life)}"@en
2 "({Senator for life)}"@en
3 "({Senator for life)}"@en
4 "({Senator for life)}"@en
5 "Abramo Bartolommeo Massalongo"@en
6 "Adolfo Panfili"@en
7 "Adolpho Ducke"@en
8 "Adriano Buzzati-Traverso"@en
9 "Agnes Pockels"@en
10 "Agostino Bassi"@en

# Conjunction (.) , disjunction (UNION), optional (OPTIONAL) patterns and filters (FILTER)

*Names of scientists or writers born in Bologna?*

```
SELECT ?N
WHERE
{
  { ?X a dbo:Scientist }
  UNION
  { ?X a dbo:Writer }

  ?X dbo:birthPlace dbr:Bologna; foaf:name ?N
} ORDER BY ?N
```

N
1 "Alberto Carpinteri"@en
2 "Alberto Carpintieri"@en
3 "Amico Bignami"@en
4 "Augusto Righi"@en
5 "Carlo Alberto Nucci"@en
6 "Clotilde Tambroni"@en

# Conjunction (.) , disjunction (UNION), optional (OPTIONAL) patterns and filters (FILTER)

*Names of scientists born in Bologna **and optionally their deathPlace?***

```
SELECT ?N ?D
WHERE
{
  ?X a dbo:Scientist ; dbo:birthPlace dbr:Bologna; foaf:name ?N
  OPTIONAL {?X dbo:deathPlace ?D }
} ORDER BY ?N
```

Note: variables can be  
unbound in a result!

N	D
1 "Alberto Carpinteri"	<a href="http://dbpedia.org/resource/Rome">http://dbpedia.org/resource/Rome</a>
2 "Alberto Carpinteri"	<a href="http://dbpedia.org/resource/Bologna">http://dbpedia.org/resource/Bologna</a>
3 "Amico Bignami"	
4 "Augusto Righi"	
5 "Carlo Alberto Nucci"	
6 "Costanzo Varolio"	<a href="http://dbpedia.org/resource/Rome">http://dbpedia.org/resource/Rome</a>

# Conjunction (.) , disjunction (UNION), optional (OPTIONAL) patterns and filters (FILTER), ...

*People born in Bologna called "Valentina"*

```
SELECT ?X ?N
WHERE
{
  ?X dbo:birthPlace dbr:Bologna ;
    foaf:name ?N .
  FILTER( CONTAINS( ?N, "Valentina" ) )
}
```

X	N	Δ
1 <a href="http://dbpedia.org/resource/Valentina_Fago">http://dbpedia.org/resource/Valentina_Fago</a>	"Valentina Fago"@en	▼

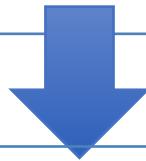
*SPARQL has lots of FILTER functions to filter text with regular expressions (REGEX), filter numerics (<,>,=,+,-...), dates, etc.)*

# CONSTRUCT Queries to create new triples (or to transform one Knowledge Graph to another)

- *Bologna scientists, their birth and death places:*

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

```
CONSTRUCT { ?X a :BolognaScientist . ?Y a :BolognaAuthor . }
WHERE { { ?X a dbo:Scientist }
        UNION
        { ?Y a dbo:Writer } }
```



```
dbr:Francesco_Maria_Grimaldi :bornIn :Bologna ; :diedIn dbr:Bologna ;
    :type ns2:BolognaScientist .
dbr:Lamberto_Cesari dbo:born :Bologna ; :diedIn dbr:Ann_Arbor ;
    :type :BolognaScientist .
dbr:Amico_Bignami :born :Bologna ; :diedIn dbr:Rome ;
    :type ns2:BolognaScientist .
dbr:Giovanni_Aldini :born :Bologna ; :diedIn dbr:Milan ;
    :type :BolognaScientist .
dbr:Guglielmo_Marconi :born :Bologna ; dbo:diedIn dbr:Rome ;
    :type :BolognaScientist .

...
dbr:Mario_Finzi :type :BolognaAuthor .
```

# Reasoning by Querying

- ***Materialisation*** (*can be done by rules/queries*) [2]
- ***Rewriting*** [1]

2. Axel Polleres, Aidan Hogan, Renaud Delbru, and Jürgen Umbrich. RDFS & OWL reasoning for linked data. In *Reasoning Web 2013*, volume 8067 of *LNCS*, pages 91--149. Springer, Mannheim, Germany, July 2013. [[.pdf](#)]

# Reasoning by Querying – Materialisation:

```
SELECT ?X WHERE
{
  ?X a dbo:Scientist ; dbo:birthPlace dbr:Bologna .
}
```

No answer ☹



instance data:

```
dbr:Marta_Grandi a dbo:Entomologist ;
dbo:birthPlace dbr:Bologna .
```

```
dbr:Costanzo_Varolio a dbo:Medician;
dbo:birthPlace dbr:Bologna .
```



Ontology (schema data):

```
dbo:Entomologist rdfs:subClassOf dbo:Scientist.
dbo:Medician rdfs:subClassOf dbo:Scientist.
dbo:Scientist rdfs:subClassOf dbo:Person.
dbo:Person rdfs:subClassOf dbo:Agent.
dbo:Organisation rdfs:subClassOf dbo:Agent.
dbo:birthPlace rdfs:domain dbo:Person .

...
```

# RDFS deduction rules:

cf. <https://www.w3.org/TR/rdf11-mt/>

$\frac{u \ a \ x}{u \ rdfs:subPropertyOf \ v} \quad v \ rdfs:subPropertyOf \ w \quad w \ rdfs:subPropertyOf \ x \quad \text{rdfs4}$	$\frac{u \ rdfs:subPropertyOf \ v \quad v \ rdfs:subPropertyOf \ w \quad w \ rdfs:subPropertyOf \ x}{u \ rdfs:subPropertyOf \ x} \quad \text{rdfs5}$	$\frac{u \ rdfs:subPropertyOf \ v \quad v \ rdfs:subPropertyOf \ w \quad w \ rdfs:subPropertyOf \ x}{u \ rdfs:subPropertyOf \ rdfs:member} \quad \text{rdfs12}$
$\frac{u \ a \ _w \ . \quad g \ l}{u \ a \ l \ .} \quad \text{rdfs13}$	$\frac{u \ rdfs:subPropertyOf \ u \ .}{u \ rdf:type \ rdfs:Property} \quad \text{rdfs6}$	$\frac{u \ rdfs:subPropertyOf \ u \ .}{u \ rdf:type \ rdfs:Datatype} \quad \text{rdfs14}$
$\frac{u \ a \ l \ .}{u \ a \ l \ .} \quad \text{rdfs15}$	$\frac{u \ rdfs:subPropertyOf \ b \ . \quad u \ a \ y \ .}{u \ a \ y \ .} \quad \text{rdfs7}$	$\frac{u \ rdfs:subClassOf \ rdfs:Resource \ .}{u \ rdf:type \ rdfs:Class} \quad \text{rdfs8}$
$\frac{u \ rdfs:domain \ u \ . \quad u \ a \ y \ .}{u \ rdf:type \ u \ .} \quad \text{rdfs2}$	$\frac{u \ rdfs:subClassOf \ x \ . \quad u \ rdf:type \ u \ .}{v \ rdf:type \ x \ .} \quad \text{rdfs9}$	$\frac{u \ rdfs:subClassOf \ u \ .}{u \ rdf:type \ rdfs:Class} \quad \text{rdfs10}$
$\frac{u \ rdfs:range \ u \ . \quad u \ a \ v \ .}{v \ rdf:type \ u \ .} \quad \text{rdfs3}$	$\frac{u \ rdfs:subClassOf \ v \ . \quad u \ rdfs:subClassOf \ x \ .}{u \ rdfs:subClassOf \ x \ .} \quad \text{rdfs11}$	
$\frac{u \ a \ x \ .}{u \ rdf:type \ rdfs:Resource \ .} \quad \text{rdfs4}$		
$\frac{u \ a \ v \ .}{v \ rdf:type \ rdfs:Resource \ .} \quad \text{rdfs4b}$		

Could be read as Datalog deduction rules, e.g.:

```
triple(U,rdfs:subClassOf,S)  :- triple(U,rdfs:subClassOf,V) , triple(V,rdfs:subClassOf,S) .
triple(V,rdfs:type,S)      :- triple(U,rdfs:subClassOf,S) , triple(V,rdf:type,U) .
```

# RDFS deduction rules:

cf. <https://www.w3.org/TR/rdf11-mt/>

$$\frac{u \text{ rdfs:subClassOf } x . \quad v \text{ rdf:type } u .}{v \text{ rdf:type } x .} \text{ rdfs9}$$

$$\frac{u \text{ rdfs:subClassOf } v . \quad v \text{ rdfs:subClassOf } x .}{u \text{ rdfs:subClassOf } x .} \text{ rdfs11}$$

Could be read as Datalog deduction rules, e.g.:

```
triple(U,rdfs:subClassOf,S)  :- triple(U,rdfs:subClassOf,V) , triple(V,rdfs:subClassOf,S) .
triple(V,rdfs:type,S)      :- triple(U,rdfs:subClassOf,S) , triple(V,rdf:type,U) .
```

# RDFS deduction rules:

cf. <https://www.w3.org/TR/rdf11-mt/>

$$\frac{u \text{ rdfs:subClassOf } x . \quad v \text{ rdf:type } u .}{v \text{ rdf:type } x .} \text{ rdfs9}$$

$$\frac{u \text{ rdfs:subClassOf } v . \quad v \text{ rdfs:subClassOf } x .}{u \text{ rdfs:subClassOf } x .} \text{ rdfs11}$$

... and Datalog deduction rules could be read/written as **SPARQL Construct** statements:

```
CONSTRUCT { ?U rdfs:subClassOf ?S } WHERE { ?U rdfs:subClassOf ?V . ?V rdfs:subClassOf ?S }
CONSTRUCT { ?V rdf:type ?S } WHERE { ?U rdfs:subClassOf ?S . ?V rdf:type ?U }
```

# Reasoning by Querying – Materialisation:

```
SELECT ?X WHERE
{
  ?X a dbo:Scientist ; dbo:birthPlace dbr:Bologna .
}
```



instance data:

dbr:Marta\_Grandi a **dbo:Entomologist** ;  
**dbo:birthPlace** dbr:Bologna .

dbr:Costanzo\_Varolio a **dbo:Medician**;  
**dbo:birthPlace** dbr:Bologna .

Applying the rules of the previous slides exhaustively (until a fixpoint), will yield additional implicit KG edges (i.e., RDF triples):

dbr:Marta\_Grandi a **dbo:Scientist**,  
**dbo:Person**, **dbo:Agent**.

dbr:Costanzo\_Varolio a  
**dbo:Scientist**, **dbo:Person**, **dbo:Agent**.



Ontology (schema data):

**dbo:Entomologist rdfs:subClassOf dbo:Scientist.**  
**dbo:Medician rdfs:subClassOf dbo:Scientist.**  
**dbo:Scientist rdfs:subClassOf dbo:Person.**  
**dbo:Person rdfs:subClassOf dbo:Agent.**  
**dbo:Organisation rdfs:subClassOf dbo:Agent.**  
**dbo:birthPlace rdfs:domain dbo:Person .**

**dbo:Entomologist rdfs:subClassOf**  
**dbo:Person, dbo:Agent.**  
**dbo:Medician rdfs:subClassOf**  
**dbo:Person, dbo:Agent.**  
**dbo:Sienticst rdfs:subClassOf**  
**dbo:Agent.**

# Reasoning by Querying – Query Rewriting:

```
SELECT ?X WHERE
{
  { ?X a dbo:Scientist } UNION { ?X a dbo:Medician } UNION { ?X a dbo:Entomologist } }
  ?X dbo:birthPlace dbr:Bologna .
}
```



DBpedia instance data:

dbr:Marta\_Grandi a dbo:Entomologist ;  
**dbo:birthPlace** dbr:Bologna .

dbr:Costanzo\_Varolio a dbo:Medician;  
**dbo:birthPlace** dbr:Bologna .

Alternatively, the rules can be used “backwards” to rewrite the original query to yield a more generic query!



DBpedia Ontology (schema data):

dbo:Entomologist rdfs:subClassOf dbo:Scientist.  
 dbo:Medician rdfs:subClassOf dbo:Scientist.  
 dbo:Scientist rdfs:subClassOf dbo:Person.  
 dbo:Person rdfs:subClassOf dbo:Agent.  
 dbo:Organisation rdfs:subClassOf dbo:Agent.  
 dbo:birthPlace rdfs:domain dbo:Person .  
 ...

# Reasoning by Querying – Query Rewriting:

```
SELECT ?X WHERE
{
  { {?X a/subclassOf* dbo:Scientist}
    ?X dbo:birthPlace dbr:Bologna . }
}
```



instance data:

dbr:Marta\_Grandi a **dbo:Entomologist** ;  
**dbo:birthPlace** dbr:Bologna .

dbr:Costanzo\_Varolio a **dbo:Medician**;  
**dbo:birthPlace** dbr:Bologna .

Alternatively, the rules can be used “backwards” to rewrite the original query to yield a more generic query!

You can also use SPARQL 1.1 path expressions in this query rewriting! [1]

 Ontology (schema data):

**dbo:Entomologist rdfs:subClassOf dbo:Scientist.**  
**dbo:Medician rdfs:subClassOf dbo:Scientist.**  
**dbo:Scientist rdfs:subClassOf dbo:Person.**  
**dbo:Person rdfs:subClassOf dbo:Agent.**  
**dbo:Organisation rdfs:subClassOf dbo:Agent.**  
**dbo:birthPlace rdfs:domain dbo:Person .**

...

# Reasoning by Querying – Query Rewriting:

```
SELECT {?X ?C1 ?C2}
WHERE { ?X a/subClassOf* ?C1;
         a/subClassOf* ?C2.
         ?C1 owl:disjointWith ?C2.}
```



instance data:

dbr:Marta\_Grandi a **dbo:Entomologist** ;  
**dbo:birthPlace** dbr:Bologna .

dbr:Costanzo\_Varolio a **dbo:Medician**;  
**dbo:birthPlace** dbr:Bologna .

Similarly, RDFS and OWL QL inconsistency checking can be done by querying! [1] (simplified)



Ontology (schema data):

**dbo:Entomologist rdfs:subClassOf dbo:Scientist.**  
**dbo:Medician rdfs:subClassOf dbo:Scientist.**  
**dbo:Scientist rdfs:subClassOf dbo:Person.**  
**dbo:Person rdfs:subClassOf dbo:Agent.**  
**dbo:Organisation rdfs:subClassOf dbo:Agent.**  
**dbo:birthPlace rdfs:domain dbo:Person .**

**dbo:Organisation owl:disjointWith dbo:Place.**

# Querying and Reasoning over Contextualised graph data

- Example: Wikidata
- Reification techniques and Property Graphs
- Some of our own work in this space:
  - AnQL [3]
  - Querying temporal information: BEAR [4]
  - Stefan Bischof's thesis work [5]

3. Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *Journal of Web Semantics (JWS)*, 12:72--95, March 2012. [[.pdf](#)]

4. Javier D. Fernandez, Jürgen Umbrich, Axel Polleres, and Magnus Knuth. Evaluating query and storage strategies for RDF archives. *Semantic Web -- Interoperability, Usability, Applicability (SWJ)*, 10(2):247--291, 2019. [[http](#)]

5. Stefan Bischof, Andreas Harth, Benedikt Kämpgen, Axel Polleres, and Patrik Schneider. Enriching integrated statistical open city data by combining equational knowledge and missing value imputation. *Journal of Web Semantics (JWS)*, 48:22--47, January 2018. [[DOI](#) | [.pdf](#)]

# Often, you also need to deal with contextualized information

- E.g. from



<https://en.wikipedia.org/wiki/Rome>

Rome

For other uses, see [Rome \(disambiguation\)](#).

**Rome** (Latin: [Roma](#)) is the capital city and a special comune of Italy situated between the Tiber and Aniene rivers. Rome is a city-state, located in Lazio, Italy. It has a population of about 3 million people, and is the second-most populous city in the European Union after Paris.

Rome is the capital of the Italian Republic and the seat of the Italian government. It is also the capital of the Lazio region, and the capital of the Metropolitan City of Rome, which covers most of the city's area. The city is located in the central-western portion of the Italian Peninsula, within the Lazio region, along the shores of the Tiber River. The Tiber is the only river in Italy that flows directly into the Mediterranean Sea.

**Automatic Extractors**

Automatic Extractors

Extracted data from Wikipedia:

Country	Italy
Demonym	Roman(s)
Type	Special Purpose (Italy)
Body	Repubblica Italiana
Region	Lazio
Area	1,285 km² (496 sq mi)
Elevation	23 m (76 ft)
Population	5,840,000 (2011 est.)
Rank	1st (Italy), 41st (EU), 41st (World)
Coordinates	41°54'N 12°29'E

Doesn't work!

„Cities in the **Italy** with more than 1M population“:

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

About: Rome

Rome (Italian: [ROMA](#); Latin: [ROMEA](#); Lazio: [Ròma](#)) is a city and comune in the central-western portion of the Italian Peninsula, within Lazio (Latium), along the shores of the Tiber River. The Tiber is the only river in Italy that flows directly into the Mediterranean Sea.

**Automatic Extractors**

Extracted data from DBpedia:

populationAsOf	2014 (xsd:integer)
populationBlank	2008481 (xsd:integer)
populationTotal	4321244 (xsd:integer)

Structured queries (SPARQL):

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

SELECT DISTINCT ?city ?pop WHERE {
  ?city a yago:City106524735 .
  ?city dbo:country :Italy .
  ?city dbo:populationTotal ?pop
  FILTER ( ?pop > 1000000 )
}
```

# Contextualised Information is better modeled in another Open Knowledge Graph: Wikidata

- Wikidata can also be queried as RDF with SPARQL!



# Wikidata as RDF ... can be queried by SPARQL

- “Simple” surface query:

```
SELECT DISTINCT ?city WHERE {
    ?city wdt:P31/wdt:P279* wd:Q515 .
    ?city wdt:P1082 ?population .
    ?city wdt:P17 wd:Q38 .
    FILTER (?population > 1000000) }
```

instance of (P31)

that class of which this subject is a particular example and member. (Subject typically an individual member with Proper Name label.) Different from P279 (subclass of).

subclass of (P279)

all instances of these items are instances of those items; this item is a class (subset) of that item. Not to be confused with Property:P31 (instance of).

city (Q515)

large and permanent human settlement

population (P1082)

number of people inhabiting the place; number of people of subject

country (P17)

sovereign state of this item

United Kingdom (Q145)

country in Europe

- What's this?

# Wikidata as RDF ... can be queried by SPARQL

- However, Wikidata has more complex info:  
**(temporal context, provenance,...)**
  - Rome:
  - <https://www.wikidata.org/wiki/Q220>

... Can I query that with SPARQL?

Yes!



The screenshot shows the Wikidata Query Service interface. At the top, there are navigation links for 'Examples', 'Help', and 'More tools'. Below the header, a SPARQL query is displayed in a code editor:

```

1
2 SELECT ?city (min(?time) as ?year) WHERE {
3   ?city wdt:P31/wdt:P279* wd:Q515.
4   ?city wdt:P17 wd:Q38 .
5   ?city p:P1082 ?statement .
6   ?statement <http://www.wikidata.org/prop/statement/value/P1082> ?value .
7   ?statement <http://www.wikidata.org/prop/qualifier/P585> ?time .
8   ?value <http://wikiba.se/ontology#quantityAmount> ?population .
9   FILTER (?population > 1000000 )
10 } GROUP BY ?city

```

The results pane shows two rows of data. The first row is for Rome (Q220), with a population of 8,415,635 in 2012. The second row is for Mexico City (Q100), with a population of 10,011,1679 in 2012.

What do we learn?

- Data and meta-data (context/provenance) at the same level → one RDF graph, mixing reification and plain data, cf. [Hernandez et al. 2015]
- Quite some Knowledge about the ontology required!

# Reification/Property Graphs:

- How to (best) describe statements about triples in RDF is a bit open... various options, inter-translateable but affect performance of querying:
  - Different Graph data models/Graph databases:
    - Labeled Directed graphs (plain RDF) - supported by RDF triple stores:



- Property graph – supported by Graph DBs, e.g. Neo4J, BlazeGraph, etc.:



# Reification/Property Graphs:

- How to (best) describe statements about triples in RDF is a bit open... various options, inter-translatable but affect performance of querying:

Annotated RDF [3]:    :Rome :capitalOf :Italy.    [1861, [

RDF reification:    [ a rdf:Statement;  
                       rdf:subject :Rome;  
                       rdf:predicate :capitalOf;  
                       rdf:object :Italy ]    :yearBegins 1861 .

“Named Graphs”    :G1 { :Rome :capitalOf :Italy. }  
                       :G1 :yearBegins 1861 .

“Singleton” properties:    :Rome :p1 :Italy.  
                               :p1 :subPropertyOf :capitalOf;  
                               :yearBegins 1861 .

Challenge1: How to reformulate the inference rules or rewritings from slide 17 to model ontological KG enrichment? [3,5]

Challenge2: How to do temporal queries efficiently? [4] E.g.  
*“Since when Is Berlin the capital of Germany, When did the capital of Germany change?”*

**:capitalOf rdfs:subPropertyOf :locatedIn. [1999, 2017]**

# Other issues (time allowed)

- Federation, Path Queries over Linked Data [6,7,8,9]
- SPARQL for KG Construction (scalability issues) [10]
- Updates [11]
- Compressed & Queryable RDF-format HDT [12]

6. Carlos Buil-Aranda, Marcelo Arenas, Oscar Corcho, and Axel Polleres. Federating queries in SPARQL1.1: Syntax, semantics and evaluation. *Journal of Web Semantics (JWS)*, 18(1), 2013. [ [http](#) ]

7. Carlos Buil-Aranda, Axel Polleres, and Jürgen Umbrich. Strategies for executing federated queries in SPARQL1.1. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, Lecture Notes in Computer Science (LNCS). Springer, October 2014. [ [.pdf](#) ]

8. Vadim Savenkov, Qaiser Mehmood, Jürgen Umbrich, and Axel Polleres. Counting to k, or how SPARQL 1.1 could be efficiently enhanced with top k shortest path queries. In *13th International Conference on Semantic Systems (SEMANTiCS)*, pages 97--103, Amsterdam, the Netherlands, September 2017. ACM. [ [.pdf](#) ]

9. [Thomas Minier](#), Hala Skaf-Molli, [Pascal Molli](#): SaGe: Web Preemption for Public SPARQL Query Services. [WWW 2019](#): 1268-1278

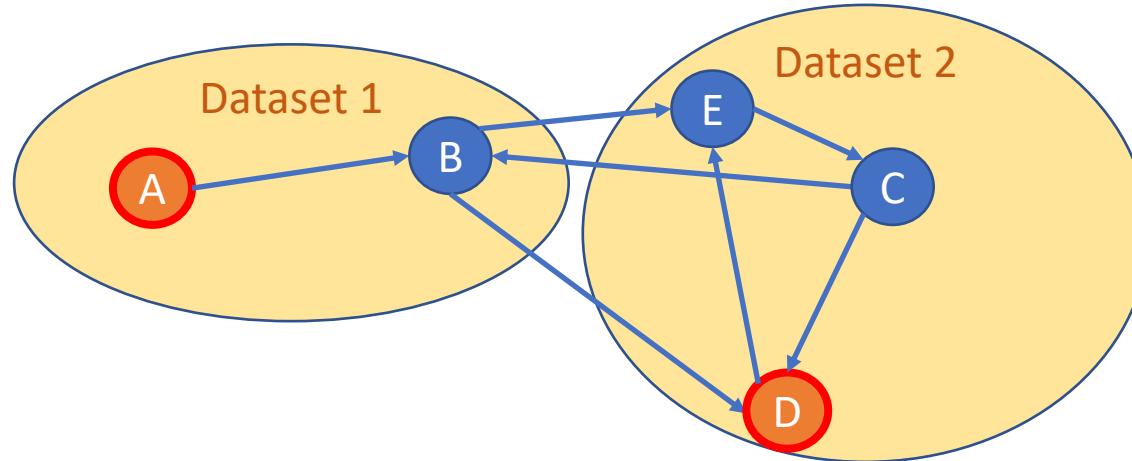
10. Sebastian Neumaier and Axel Polleres. Enabling spatio-temporal search in open data. *Journal of Web Semantics (JWS)*, 55, March 2019. [ [DOI](#) | [http](#) ]

11. Albin Ahmeti, Javier Fernández, Axel Polleres, and Vadim Savenkov. Updating wikipedia via DBpedia mappings and SPARQL. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *Proceedings of the 14th European Semantic Web Conference (ESWC2017)*, volume 10249 of *Lecture Notes in Computer Science (LNCS)*, pages 485--501, Portorož, Slovenia, May 2017. Springer. [ [.pdf](#) ]

12. Javier D. Fernández, Miguel A. Martínez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. Binary RDF Representation for Publication and Exchange (HDT). *Journal of Web Semantics (JWS)*, 19(2), 2013. [ [http](#) ]

# Federation/Path Queries

Common problem in graphs, not doable with SPARQL, but with extensions [8]:  
 “Give me the (k) shortest paths between two nodes?”



```
:a :p :b.  

:b :p :d, :e.  

:c :p :b, :d.  

:d :p :e.  

:e :p :c.
```

[rdf2hdt.sh](#) -rdftype turtle testgraph.ttl testgraph.hdt

[hdtsparql.sh](#) testgraph.hdt "PREFIX ppf: <[java:at.ac.wu.arqext.path.](#)>  
 SELECT \* WHERE{ ?path ppf:topk (:a :d 2) }"

You can solve this by extending SPARQL [8] with  
 bidirectional bfs over HDT [12]  
[https://bitbucket.org/vadim\\_savenkov/topk-pfn/](https://bitbucket.org/vadim_savenkov/topk-pfn/)

But how to do this  
 effectively in a Federated  
 setting? Open Research  
 question!!!

k=2

# Scalability of SPARQL endpoints?

```

CONSTRUCT {
  ?event rdfs:label ?label ; dcterms:isPartOf ?parent ; dcterms:coverage ?geocoordinates ;
    timex:hasStartTime ?startDateTime ; timex:hasEndTime ?endDateTime ; dcterms:spatial ?geoentity .
} WHERE {
  # find events with (for the moment) English, German, or non-language-specific labels:
  ?event wdt:P31/wdt:P279* wd:Q11906554 . ?event rdfs:label ?label .
  FILTER ( LANGMATCHES(LANG(?label), "EN") || LANGMATCHES(LANG(?label), "DE") || LANG(?label) = "" )

  { # restrict to certain event categories, e.g. (for the moment) elections and sports events:  #sports competitions
  { ?event wdt:P31/wdt:P279* wd:Q40231 } UNION { ?event wdt:P31/wdt:P279* wd:Q13406554 }
  }

  { ?event wdt:P585 ?startDateTime . FILTER ( ?startDateTime > "1900-01-01T00:00:00"^^xsd:dateTime ) }
  UNION
  { ?event wdt:P580 ?startDateTime . FILTER ( ?startDateTime > "1900-01-01T00:00:00"^^xsd:dateTime )
  ?event wdt:P582 ?endDateTime . FILTER ( DATATYPE(?endDateTime) = xsd:dateTime ) }
  BIND(IF(bound(?endDateTime), ?endDateTime, xsd:dateTime(CONCAT(STR(xsd:date(?startDateTime)), "T23:59:59"))) AS ?endDateTime)

  OPTIONAL { ?event wdt:P361 ?parent }
  OPTIONAL { ?event wdt:P276?/(wdt:P17|wdt:P131) ?geoentity }
  OPTIONAL { ?event wdt:P276?/wdt:P625 ?geocoordinates }
}

```

Figure 6: Conceptual SPARQL query to extract event data (from elections and sports competitions) from Wikidata – times out on <https://query.wikidata.org>.

Problem occurred for us when constructing another KG from Wikidata [10] You can solve this by:

1. extracting relevant triples to answer the query via HDT [12] and
2. executing targeted CONSTRUCT queries to the full SPARQL endpoint for specific sub-queries in order to materialize path expressions. Details cf. [here](#)

# References:

1. Stefan Bischof, Markus Krötzsch, Axel Polleres, and Sebastian Rudolph. Schema-agnostic query rewriting in SPARQL 1.1. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, Lecture Notes in Computer Science (LNCS). Springer, October 2014. [ [pdf](#) ]
2. Axel Polleres, Aidan Hogan, Renaud Delbru, and Jürgen Umbrich. RDFS & OWL reasoning for linked data. In *Reasoning Web 2013*, volume 8067 of *LNCS*, pages 91–149. Springer, Mannheim, Germany, July 2013. [ [pdf](#) ]
3. Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *Journal of Web Semantics (JWS)*, 12:72–95, March 2012. [ [pdf](#) ]
4. Javier D. Fernandez, Jürgen Umbrich, Axel Polleres, and Magnus Knuth. Evaluating query and storage strategies for RDF archives. *Semantic Web -- Interoperability, Usability, Applicability (SWJ)*, 10(2):247–291, 2019. [ [http](#) ]
5. Stefan Bischof, Andreas Harth, Benedikt Kämpgen, Axel Polleres, and Patrik Schneider. Enriching integrated statistical open city data by combining equational knowledge and missing value imputation. *Journal of Web Semantics (JWS)*, 48:22–47, January 2018. [ [DOI](#) | [pdf](#) ]
6. Carlos Buil-Aranda, Marcelo Arenas, Oscar Corcho, and Axel Polleres. Federating queries in SPARQL1.1: Syntax, semantics and evaluation. *Journal of Web Semantics (JWS)*, 18(1), 2013. [ [http](#) ]
7. Carlos Buil-Aranda, Axel Polleres, and Jürgen Umbrich. Strategies for executing federated queries in SPARQL1.1. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, Lecture Notes in Computer Science (LNCS). Springer, October 2014. [ [pdf](#) ]
8. Vadim Savenkov, Qaiser Mehmood, Jürgen Umbrich, and Axel Polleres. Counting to k, or how SPARQL 1.1 could be efficiently enhanced with top k shortest path queries. In *13th International Conference on Semantic Systems (SEMANTiCS)*, pages 97–103, Amsterdam, the Netherlands, September 2017. ACM. [ [pdf](#) ]
9. [Thomas Minier](#), Hala Skaf-Molli, [Pascal Molli](#): SaGe: Web Preemption for Public SPARQL Query Services. [WWW 2019](#): 1268–1278
10. Sebastian Neumaier and Axel Polleres. Enabling spatio-temporal search in open data. *Journal of Web Semantics (JWS)*, 55, March 2019. [ [DOI](#) | [http](#) ]
11. Albin Ahmeti, Javier Fernández, Axel Polleres, and Vadim Savenkov. Updating wikipedia via DBpedia mappings and SPARQL. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *Proceedings of the 14th European Semantic Web Conference (ESWC2017)*, volume 10249 of *Lecture Notes in Computer Science (LNCS)*, pages 485–501, Portorož, Slovenia, May 2017. Springer. [ [pdf](#) ]
12. Javier D. Fernández, Miguel A. Martinez-Prieto, Claudio Gutiérrez, Axel Polleres, and Mario Arias. Binary RDF Representation for Publication and Exchange (HDT). *Journal of Web Semantics (JWS)*, 19(2), 2013. [ [http](#) ]