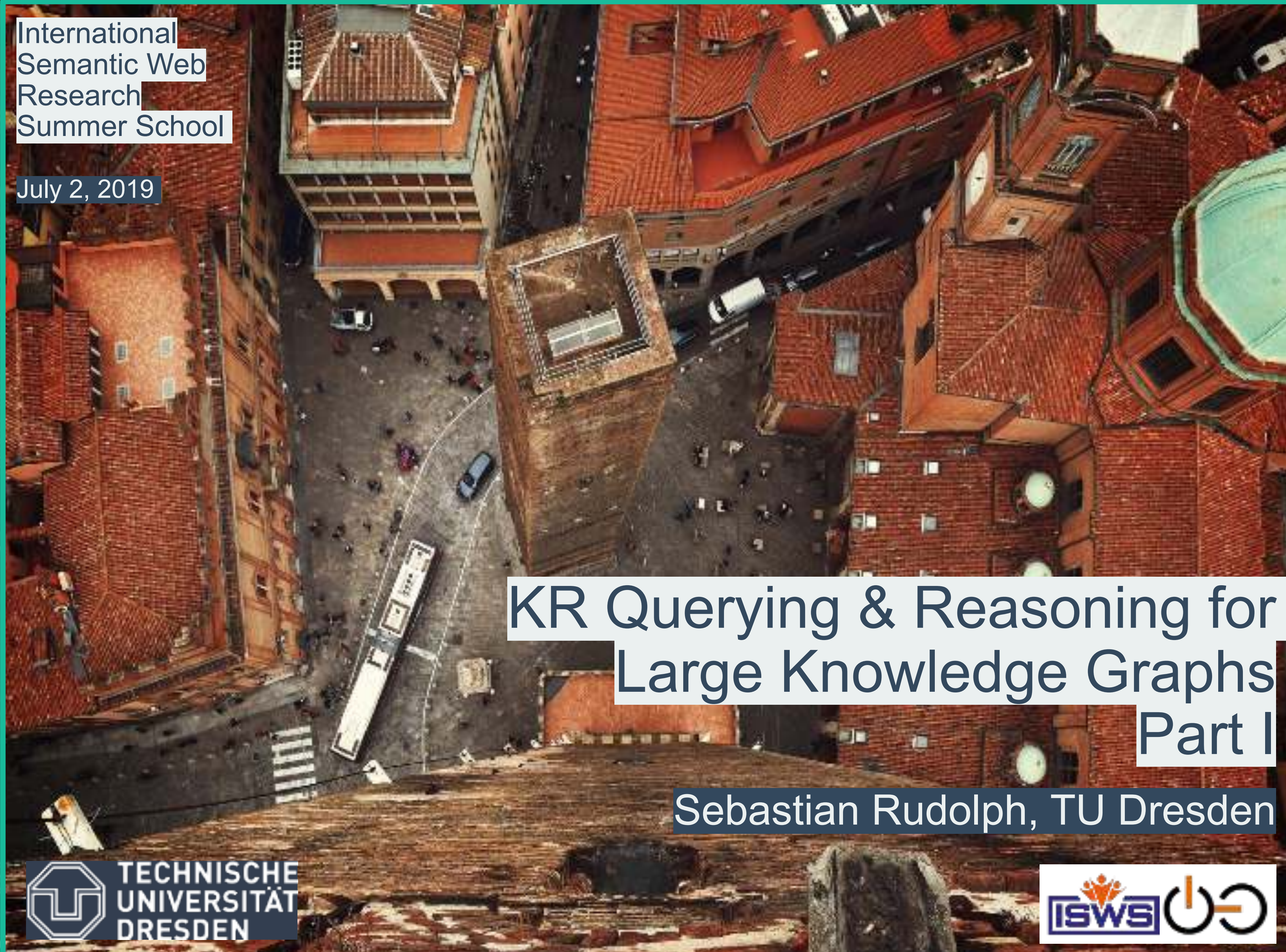


International Semantic Web Research Summer School

July 2, 2019

KR Querying & Reasoning for Large Knowledge Graphs Part I

Sebastian Rudolph, TU Dresden

TECHNISCHE UNIVERSITÄT DRESDEN

ISWS

International Semantic Web Research Summer School

July 2, 2019

KR **Querying** & Reasoning for Large Knowledge Graphs Part II

Axel Polleres, WU Vienna

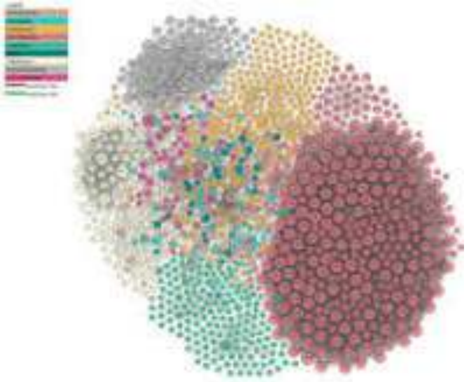
WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

ISWS

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 DBpedia

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

- 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“:

Automatic Exctractors

http://dbpedia.org/resource/London

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

Structured queries (**SPARQL**):

http://yasgui.org/short/UVOyhX8ft

```
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 DBpedia

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

Automatic Exctractors

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

http://dbpedia.org/resource/Rome

Structured queries (SPARQL):

Doesn't work!

http://yasgui.org/short/UVOyhX8ft

```
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. DBpedia

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]

<u>Querying</u> 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

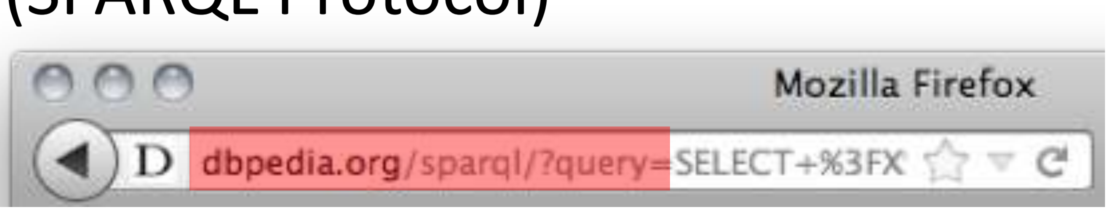
Find some more detailed material to introduce SPARQL on my Webpage:
http://polleres.net/presentations/

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)



Try it!

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
```

	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

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

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"@en	
2	"Alberto Carpintieri"@en	
3	"Amico Bignami"@en	http://dbpedia.org/resource/Rome
4	"Augusto Righi"@en	http://dbpedia.org/resource/Bologna
5	"Carlo Alberto Nucci"@en	
6	"Costanzo Varolio"@en	http://dbpedia.org/resource/Rome

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
1	http://dbpedia.org/resource/Valentina_Fago	"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 :BolgnaScientist . ?Y a :BolgonaAuthor . }
WHERE { { ?X a dbo:Scientist }
        UNION
        { ?Y a dbo:Writer } }
```

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

...

dbr:Mario_Finzi :type :BolgonaAuthor .
```

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 ☹

DBpedia instance data:

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

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

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 .

...
```

RDFS deduction rules:

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

$$\frac{}{\texttt{u a x}}\ \text{rdfsax}$$

$$\frac{\texttt{u a \_:n .}}{\texttt{u a l .}}\ \text{gl}$$

$$\frac{\texttt{u a l .}}{\texttt{\_:n rdf:type rdfs:Literal .}}\ \text{rdfs1}$$

$$\frac{\texttt{a rdfs:domain x .}\quad \texttt{u a y .}}{\texttt{u rdf:type x .}}\ \text{rdfs2}$$

$$\frac{\texttt{a rdfs:range x .}\quad \texttt{u a v .}}{\texttt{v rdf:type x .}}\ \text{rdfs3}$$

$$\frac{\texttt{u a x .}}{\texttt{u rdf:type rdfs:Resource .}}\ \text{rdfs4a}$$

$$\frac{\texttt{u a v .}}{\texttt{v rdf:type rdfs:Resource .}}\ \text{rdfs4b}$$

$$\frac{\texttt{u rdfs:subPropertyOf v .}\quad \texttt{v rdfs:subPropertyOf x .}}{\texttt{u rdfs:subPropertyOf x .}}\ \text{rdfs5}$$

$$\frac{\texttt{u rdf:type rdf:Property .}}{\texttt{u rdfs:subPropertyOf u .}}\ \text{rdfs6}$$

$$\frac{\texttt{a rdfs:subPropertyOf b .}\quad \texttt{u a y .}}{\texttt{u b y .}}\ \text{rdfs7}$$

$$\frac{\texttt{u rdf:type rdfs:Class .}}{\texttt{u rdfs:subClassOf rdfs:Resource .}}\ \text{rdfs8}$$

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

$$\frac{\texttt{u rdf:type rdfs:Class .}}{\texttt{u rdfs:subClassOf u .}}\ \text{rdfs10}$$

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

$$\frac{\texttt{u rdf:type rdfs:ContainerMembershipProperty .}}{\texttt{u rdfs:subPropertyOf rdfs:member .}}\ \text{rdfs12}$$

$$\frac{\texttt{u rdf:type rdfs:Datatype .}}{\texttt{u rdfs:subClassOf rdfs:Literal .}}\ \text{rdfs13}$$

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/

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

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

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/

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

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

… 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 rdfs: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 .
}
```

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

DBpedia instance data:

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

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

```
dbr:Marta_Grandi a dbo:Scientist,
dbo:Person, dbo:Agent.

dbr:Costanzo_Varolio  a
dbo:Scientist, dbo:Person, dbo:Agent.
```

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 .

...
```

```
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 .
}
```

DBpedia instance data:

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

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

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 .
...
```

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]

Reasoning by Querying – Query Rewriting:

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

DBpedia 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)

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 .
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]

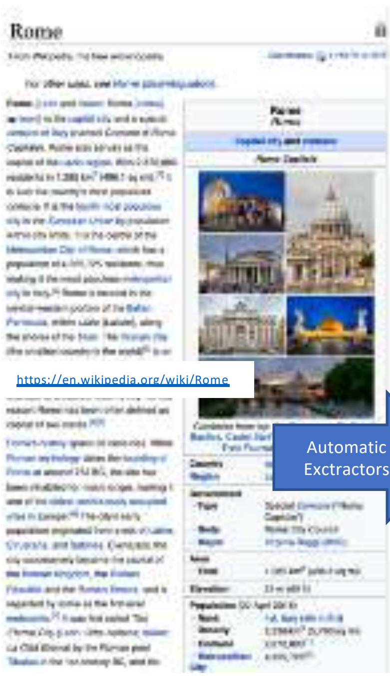
3. Antoine Zimmermann, Nuno Lopes, Axel Polleres, and Umberto Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *Journal of Web Semantics (JWS)*, 12:72--95, March 2012. [.pdf]

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 DBpedia



Automatic Exctractors

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

Doesn’t work!

Structured queries (SPARQL):

http://yasgui.org/short/UVOyhX8ft

```
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 )
}
```

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) }
```

- What’s this?

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

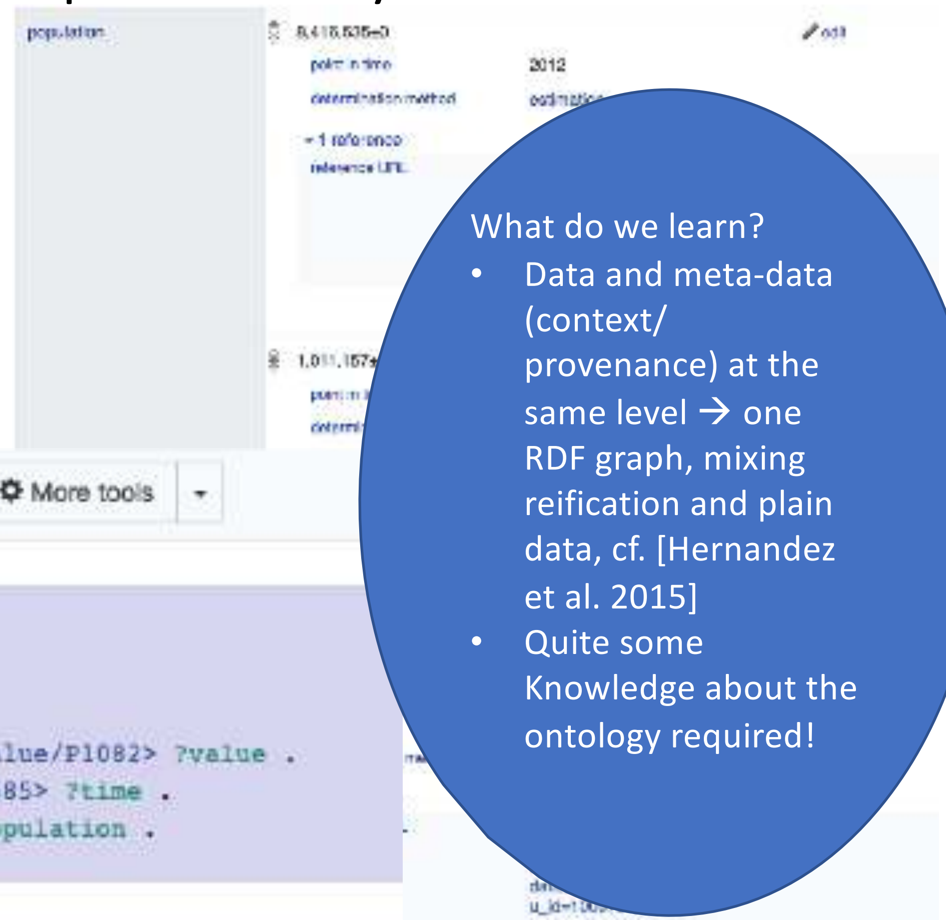
population (P1082)
number of people inhabiting the place; number of people of subject

country (P17)
sovereign state of this item

United Kingdom (Q145)
country in Europe

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!

What do we learn?

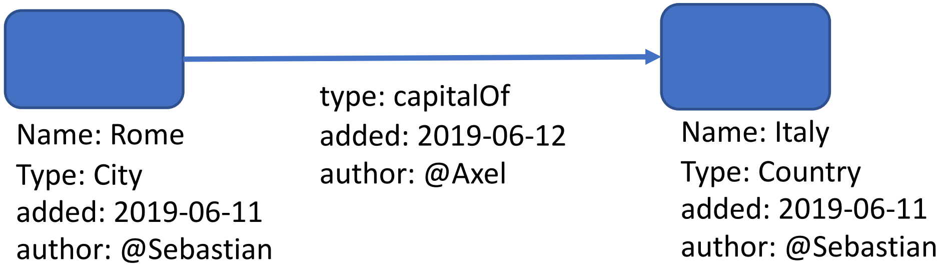
- Data and meta-data (context/ provenance) at the same level → one RDF graph, mixing reification and plain data, cf. [Hernandez et al. 2015]
- Quite some Knowledge about the ontology required!

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 efficently? [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

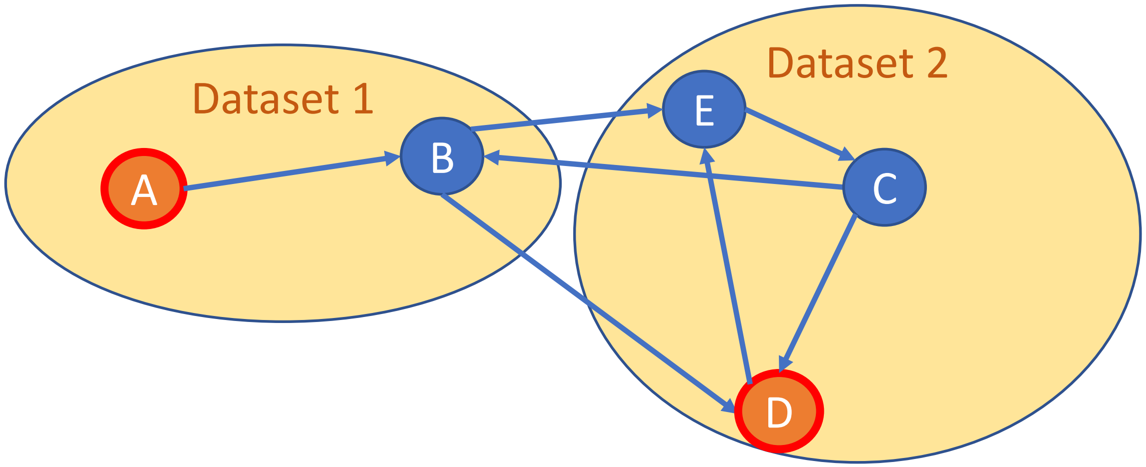
10. Sebastian Neumaier and Axel Polleres. Enabling spatio-temporal search in open data. *Journal of Web Semantics (JWS)*, 55, March 2019. [DOI | http]

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. 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]

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/

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

k=2

Scalability of SPARQL endpoints?

```
CONSTRUCT {                                                                          knowledge graph model
  ?event rdfs:label ?label ; dcterms:isPartOf ?parent ; dcterms:coverage ?geocoordinates ;
         timex:hasStartTime ?startDateTime ; timex:hasEndTime ?endDateTime ; dcterms:spatial ?geoentity .
} WHERE {                                                                                           labels
  # find events with (for the moment) English, German, or non-language-specific labels:
  ?event wdt:P31/wdt:P279* wd:Q1190554 . ?event rdfs:label ?label .
  FILTER ( LANGMATCHES(LANG(?label), "EN") || LANGMATCHES(LANG(?label), "DE") || LANG(?label) = "" )
                                                                                             filter events
  { # 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 }
  }
                                                                                                temporal
  {
    { ?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)
                                                                                              geospatial
  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]