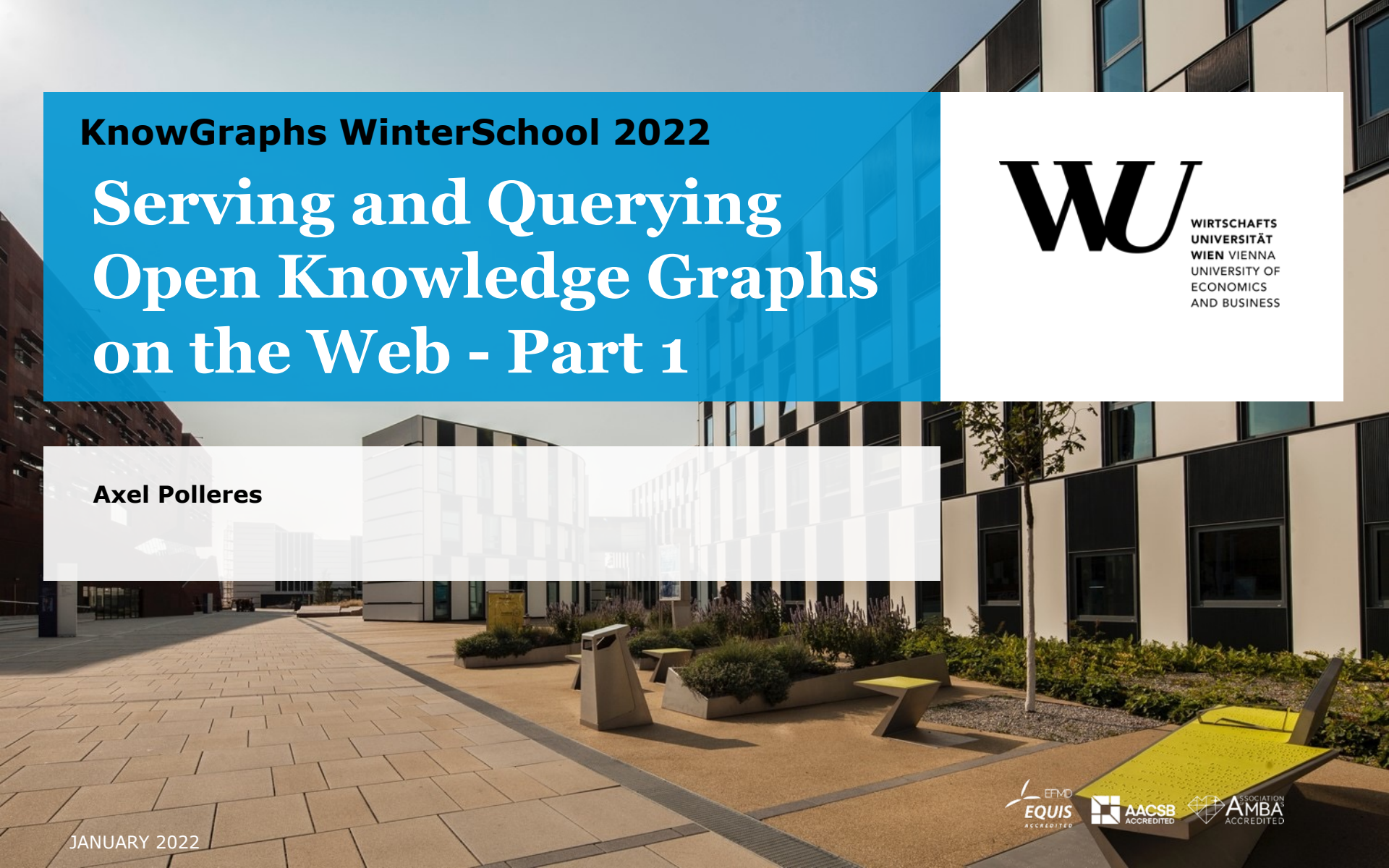


KnowGraphs WinterSchool 2022

Serving and Querying Open Knowledge Graphs on the Web - Part 1

Axel Polleres

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

JANUARY 2022

What I've planned for today:

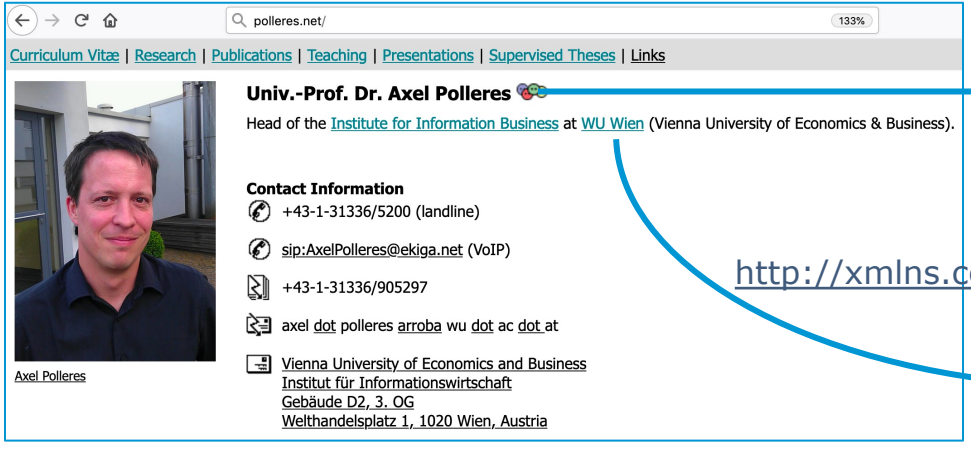
- **Part 1:**
  - Interlude
  - Practical examples on querying Open KGs with SPARQL
  - Challenges/limitations of SPARQL over public endpoints
- **Part 2:**
  - Serve and query KGs for local processing – HDT
  - Addressing the SPARQL endpoint bottleneck – where are we?
    - Linked Data Fragments
    - Smart-KG
    - Wise-KG

Standard format (RDF) & Standard Query language (SPARQL) for Graph Data

- Data representation
  - RDF (= **R**esource **D**escription **F**ramework)
    - a standard Format for publishing Graph Data on the Web.
    - Can be seen as a labeled graph

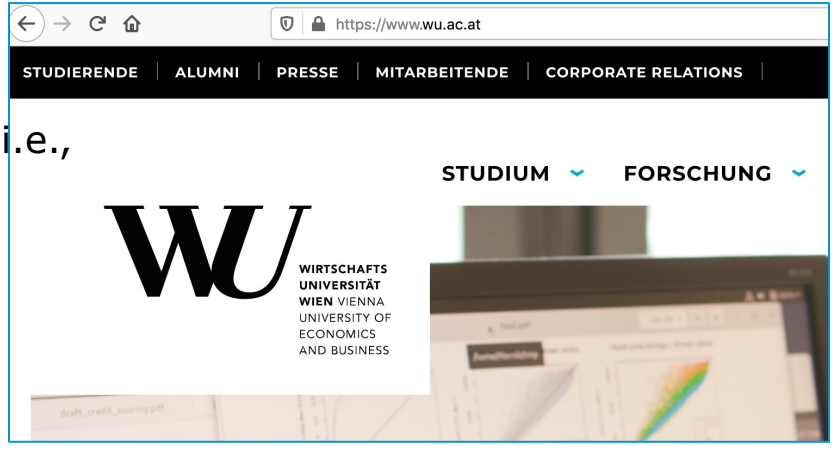
- **Querying**
  - **SPARQL**
    - **a query language (similar to SQL) for RDF data**

RDF was invented for the Web, for annotating Webpages with "typed links"



isA → http://xmlns.com/foaf/0.1/Person

http://xmlns.com/foaf/0.1/workplaceHomepage →



RDF uses URIs for the links to define their **Semantics**, i.e., when you look up these links, you can find out what http://xmlns.com/foaf/0.1/workplaceHomepage means!

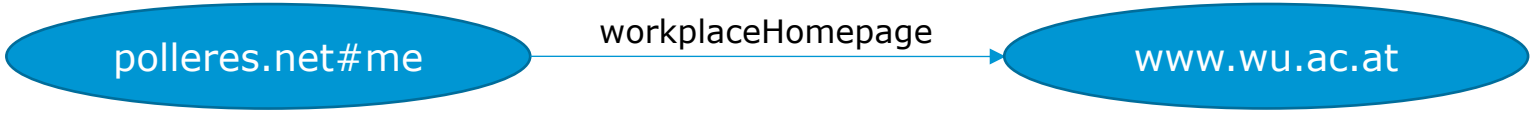
You find such RDF links on my Homepage!
Click on  on http://polleres.net

RDF Triple:

- A triple of URLs

```
<http://www.polleres.net#me> <http://xmlns.com/foaf/0.1/workplaceHomepage> <http://www.wu.ac.at> .
```

- … can be seen as an edge in a Graph:



RDF vocabularies 1/2:

Vocabularies (collections of URIs to define meaning for Links) are identified by a common **URI prefix:**

The

RDF Core (rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#**)** and

RDFS Schema (rdfs: http://www.w3.org/2000/01/rdf-schema#**)**

vocabularies define basic meaning for relations such as is-A, subclasses/subproperties, (human-readable) labels, etc. according to the RDF specification:

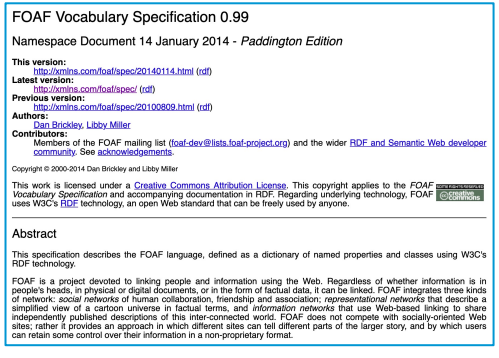
- Important URIs that used for links (in many KGs):
  - http://www.w3.org/1999/02/22-rdf-syntax-ns#type (or short **rdf:type**)
  - http://www.w3.org/2000/01/rdf-schema#label (or short **rdfs:label**)
  - http://www.w3.org/2000/01/rdf-schema#subPropertyOf (or short **rdfs:subClassOf**)
  - http://www.w3.org/2000/01/rdf-schema#subClassOf (or short **rdfs:subPropertyOf**)
  - http://www.w3.org/2000/01/rdf-schema#domain (or short **rdfs:domain**)
  - http://www.w3.org/2000/01/rdf-schema#range (or short **rdfs:range**)

RDF vocabularies 2/2:

- **Other vocabularies:**
  - **foaf:** Prefix: http://xmlns.com/foaf/0.1/ … The "Friend-of-a-friend' vocabulary models common properties of and classses relating to Persons and social relationships**. E.g.:**

    Properties:
    - **name**
    - **nickname**
    - **workplaceHomepage**
    - **knows**

    Classes:
    - **Agent**
    - **Person**
    - **Document**
    - **Image**



FOAF Vocabulary Specification 0.99

Namespace Document 14 January 2014 - *Paddington Edition*

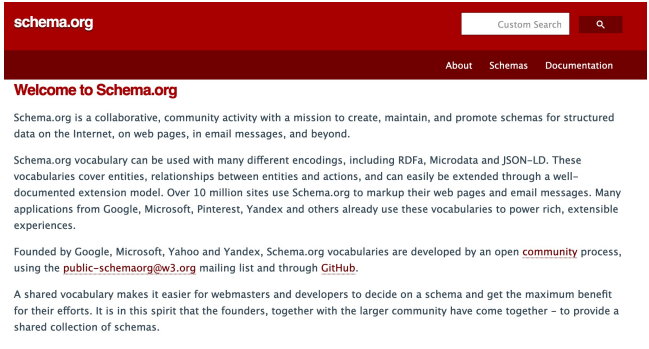
This version:
http://xmlns.com/foaf/spec/20140114.html (rdf)
Latest version:
http://xmlns.com/foaf/spec/ (rdf)
Previous version:
http://xmlns.com/foaf/spec/20100809.html (rdf)
Authors:
Dan Brickley, Libby Miller
Contributors:
Members of the FOAF mailing list (foaf-dev@lists.foaf-project.org) and the wider RDF and Semantic Web developer community. See acknowledgements.

Abstract

This specification describes the FOAF language, defined as a dictionary of named properties and classes using W3C's RDF technology.

FOAF is a project devoted to linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network: *social networks* of human collaboration, friendship and association; *representational networks* that describe a simplified view of a cartoon universe in factual terms, and *information networks* that use Web-based linking to share independently published descriptions of this inter-connected world. FOAF does not compete with socially-oriented Web sites; rather it provides an approach in which different sites can tell different parts of the larger story, and by which users can retain some control over their information in a non-proprietary format.

- **schema:** Prefix: http://schema.org/ …
  - Classes and properties important for search engines
  - (founded by Google, Microsoft, Yahoo and Yandex)
- or domain/KG-specific vocabularies, eg.
  - **dbo:** (***DBpedia*** Ontology)
  - **wd:**, **wdt:** (***Wikidata*** entities and properties)



schema.org

Custom Search

About Schemas Documentation

Welcome to Schema.org

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD. These vocabularies cover entities, relationships between entities and actions, and can easily be extended through a well-documented extension model. Over 10 million sites use Schema.org to markup their web pages and email messages. Many applications from Google, Microsoft, Pinterest, Yandex and others already use these vocabularies to power rich, extensible experiences.

Founded by Google, Microsoft, Yahoo and Yandex, Schema.org vocabularies are developed by an open community process, using the public-schemaorg@w3.org mailing list and through GitHub.

A shared vocabulary makes it easier for webmasters and developers to decide on a schema and get the maximum benefit for their efforts. It is in this spirit that the founders, together with the larger community have come together – to provide a shared collection of schemas.

RDF Syntaxes – A simple RDF file:

simple1.nt in NTriples Syntax :

```
<http://www.example.org/klaus> <http://xmlns.com/foaf/0.1/knows> <http://www.example.org/karl> .
<http://www.example.org/klaus> <http://xmlns.com/foaf/0.1/nickname> "Niki" .
<http://www.example.org/alice> <http://xmlns.com/foaf/0.1/knows> <http://www.example.org/bob> .
<http://www.example.org/alice> <http://xmlns.com/foaf/0.1/knows> <http://www.example.org/karl> .
<http://www.example.org/alice> <http://xmlns.com/foaf/0.1/name> "Alice Wonderland" .
<http://www.example.org/karl> <http://xmlns.com/foaf/0.1/name> "Karl Mustermann" .
<http://www.example.org/karl> <http://xmlns.com/foaf/0.1/knows> <http://www.example.org/joan> .
<http://www.example.org/bob> <http://xmlns.com/foaf/0.1/name> "Robert Mustermann" .
<http://www.example.org/bob> <http://xmlns.com/foaf/0.1/nickname> "Bobby" .
```

RDF Syntaxes – A simple RDF file:

simple1.ttl in Turtle (Terse RDF Language) Syntax is a bit more readable:

```
# using the FOAF vocabulary, see http://xmlns.com/foaf/spec/

@prefix : <http://www.example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

:klaus foaf:knows :karl .
:klaus foaf:nickname "Niki".
:alice foaf:knows :bob .
:alice foaf:knows :karl .
:alice foaf:name "Alice Wonderland" .
:karl foaf:name "Karl Mustermann" .
:karl foaf:knows :joan.
:bob foaf:name "Robert Mustermann" .
:bob foaf:nickname "Bobby" .
```

RDF Syntaxes – A simple RDF file:

simple1.ttl in Turtle (Terse RDF Language) Syntax is a bit more readable –

Turtle Syntax also allows some **shortcuts** to **group Triples with common subjects:**

```
# using the FOAF vocabulary, see http://xmlns.com/foaf/spec/

@prefix : <http://www.example.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

:klaus foaf:knows :karl ;
       foaf:nickname "Niki".
:alice foaf:knows :bob , :karl ; foaf:name "Alice Wonderland" .
:karl foaf:name "Karl Mustermann" ; foaf:knows :joan.
:bob foaf:name "Robert Mustermann" ; foaf:nickname "Bobby" .
```

Note: We will need Turtle Syntax for querying RDF data!

Standards have lead to big open KGs…

- … some of which availabe on the Web
- … queryable via SPARQL endpoints!

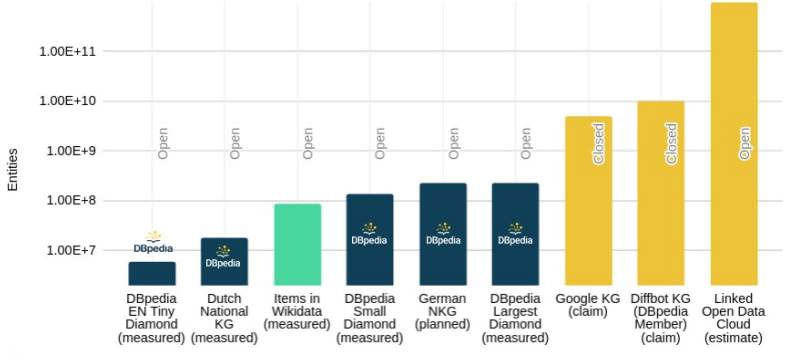
1,101,215,718 triples/edges

13,602,048,837 triples/edges

- plus useful convenience tools:
  - http://prefix.cc/ … find out common URI prefixes for formulating queries
  - http://yasgui.triply.cc/ … really nice frontend for querying SPARQL endpoints, e.g. DBpedia
  - https://query.wikidata.org/ … really nice frontend specifically for querying Wikidata
  - Plus tons of APIs (e.g. Python, R packages, etc.)

Standards have lead to big open KGs…
… plus (even bigger?) closed KGs

- … plus growing interest of Enterprises in KGs
  - Success stories of mainly monolithic (but huge) Knowledge Graphs rather than a network of Linked small KGs:



https://www.dbpedia.org 2021

https://www.slideshare.net/Frank.van.Harmelen/adoption-of-knowledge-graphs-late-2019

Open KGs (April 2021)	
DBpedia	~4.58m e
Yago4	~50m enti
Wikidata	~93m enti

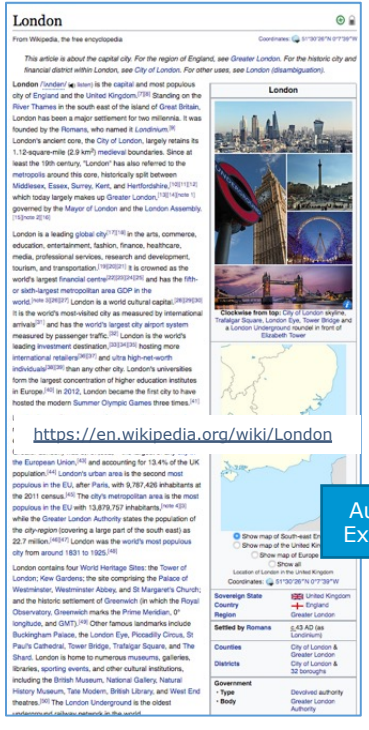
N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Hogan et al.: Knowledge Graphs. Co

How do we query these KGs in practice?

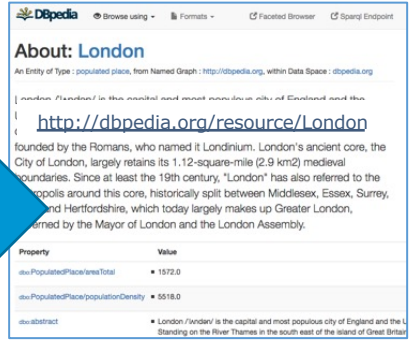
- … SPARQL!

RDF used in practice on the Web: DBpedia - a "Database-version" of Wikipedia:

- E.g. from DBpedia



Automatic Exctractors



- One of the central datasets of the Linked Open Data-Cloud
- RDF extracted from Wikipedia-Infoboxes
- You can use a language called SPARQL endpoint (roughly: SQL for RDF) to do ***structured queries*** over RDF:
  - „Cities in the UK with more than 1M population“:

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 )
}
```

RDF used in practice on the Web: Another Open Knowledge Graph: Wikidata

- slightly different idea than DBpedia:
  - a Wikimedia foundation project itself
  - put simply: "replace factual data within Wikipedia by a (graph) Database"

- Wikidata can also be queried as RDF with SPARQL!

Let's learn some SPARQL with Wikidata

- "Simple" surface query:

  Which cities in the UK have more than 1M people?

```
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

Let's learn some SPARQL with Wikidata

- You can try out the queries on http://query.wikidata.org/

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) … Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) … instanceOf

Triple Patterns (TPs): Try this query for

"Give me 10 buildings"

https://w.wiki/4TAP

Let's learn some SPARQL with Wikidata

- You can try out the queries on http://query.wikidata.org/

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) … Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) … instanceOf

Basic Graph patterns (BGPs): "**Join**" between edges/triples:

"Give me 10 buildings **in Austria**"

https://w.wiki/4TAY

Let's learn some SPARQL with wikidata

- You can try out the queries on http://query.wikidata.org/

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) … Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) … instanceOf

UNION between patterns:

Try this query for "Give me 10 buildings in **Austria or Germany**"

https://w.wiki/4TAf

Let's learn some SPARQL with wikidata

- You can try out the queries on http://query.wikidata.org/

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) … Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) … instanceOf

***FILTER*s** (similar to WHERE conditions in SQL):

"Give me **the German labels of** 10 buildings in Austria or Germany"

https://w.wiki/4TAk

Let's learn some SPARQL with wikidata

- You can try out the queries on http://query.wikidata.org/

  https://www.wikidata.org/entity/Q41176 (wd:Q41176) … Building
  http://www.wikidata.org/prop/direct/P31 (wdt:P31) … instanceOf

OPTIONAL (similar to OUTER JOIN in SQL):

"Give me the German labels of 10 buildings in Austria **and their architect (if available)**"

https://w.wiki/4TAn

Full details of SPARQL and many more examples:

- https://www.w3.org/TR/sparql11-query/

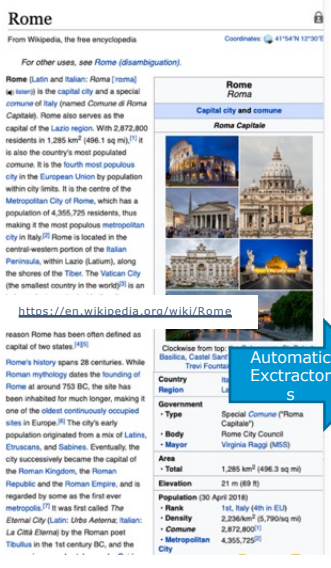
- Supported by various modern graph databases.

What I've planned for today:

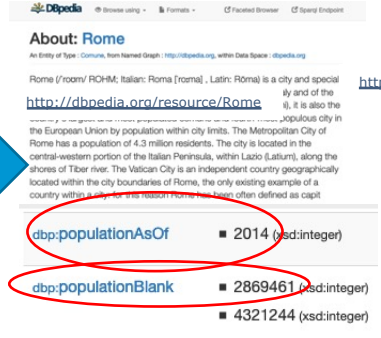
- **Part 1:**
  - Interlude
  - Practical tutorial on querying Open KGs with SPARQL
  - ***(some) Challenges/limitations of SPARQL over public endpoints***
- **Part 2:**
  - Serve and query KGs for local processing – HDT
  - Addressing the SPARQL endpoint bottleneck – where are we?
    - Linked Data Fragments
    - Smart-KG
    - Wise-KG

Challenge 1: Often, you also need to deal with *contextualized* information

- E.g. from DBpedia



Automatic Exctractors



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

Structured queri (**SPARQL**):

http://yasgui.org/short/UVOyhX8ft

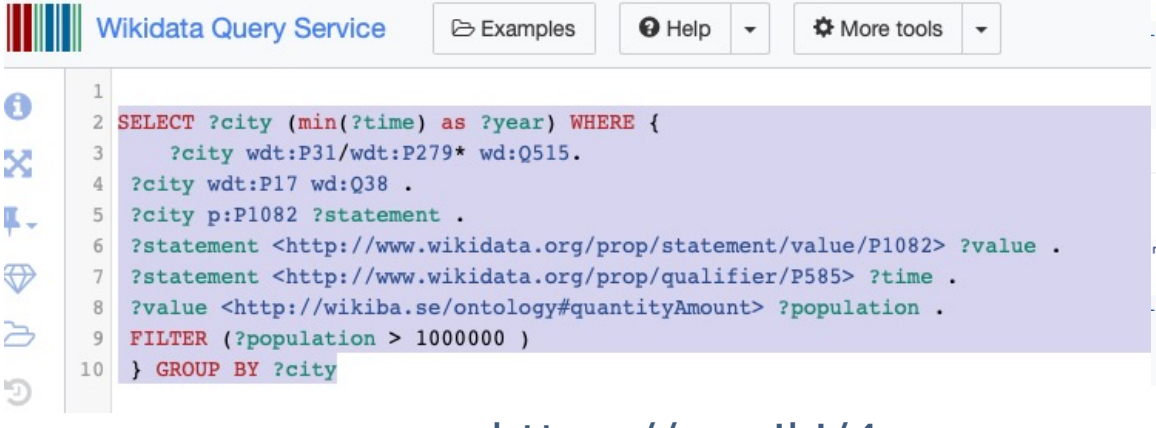
```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX yago: <http://dbpedia.org/class/yago/>

SELECT DISTINCT ?city ?pop WHERE {
   ?city a yago:City108524735 .
   ?city dbo:country :Italy.
   ?city dbo:populationTotal ?pop

   FILTER ( ?pop > 1000000 )
}
```

Doesn't work!

Challenge 1: Wikidata as RDF … In Wikidata even context information can be queried by SPARQL

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

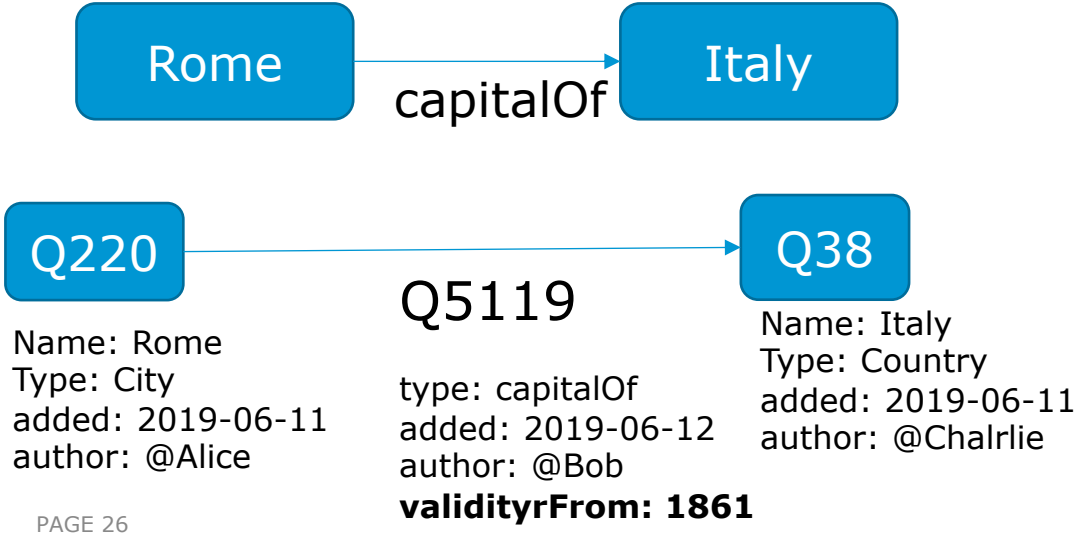
… Can I query that with SPARQL? Yes!

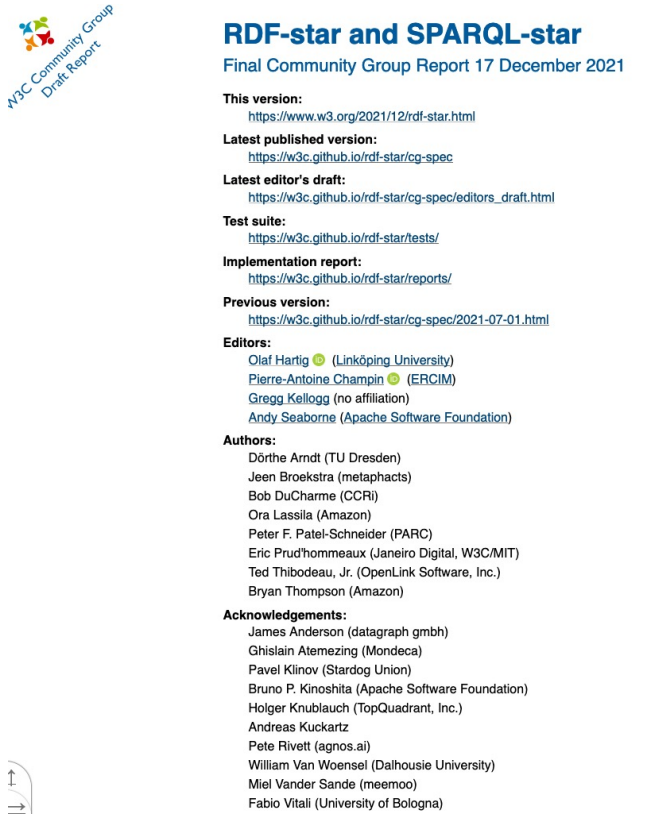


https://w.wiki/4rs

Challenge 1: Contextualized information in RDF

- no standard as of yet. State of affairs:
  - Wikidata has its own proprietary extension (cf. last slide)
  - Alternative representations/engines involve **Property Graphs**
  - ongoing work: RDF*/SPARQL* community group





W3C Community Group Draft Report

RDF-star and SPARQL-star

Final Community Group Report 17 December 2021

This version:
https://www.w3.org/2021/12/rdf-star.html

Latest published version:
https://w3c.github.io/rdf-star/cg-spec

Latest editor's draft:
https://w3c.github.io/rdf-star/cg-spec/editors_draft.html

Test suite:
https://w3c.github.io/rdf-star/tests/

Implementation report:
https://w3c.github.io/rdf-star/reports/

Previous version:
https://w3c.github.io/rdf-star/cg-spec/2021-07-01.html

Editors:
Olaf Hartig (Linköping University)
Pierre-Antoine Champin (ERCIM)
Gregg Kellogg (no affiliation)
Andy Seaborne (Apache Software Foundation)

Authors:
Dörthe Arndt (TU Dresden)
Jeen Broekstra (metaphacts)
Bob DuCharme (CCRi)
Ora Lassila (Amazon)
Peter F. Patel-Schneider (PARC)
Eric Prud'hommeaux (Janeiro Digital, W3C/MIT)
Ted Thibodeau, Jr. (OpenLink Software, Inc.)
Bryan Thompson (Amazon)

Acknowledgements:
James Anderson (datagraph gmbh)
Ghislain Atemezing (Mondeca)
Pavel Klinov (Stardog Union)
Bruno P. Kinoshita (Apache Software Foundation)
Holger Knublauch (TopQuadrant, Inc.)
Andreas Kuckartz
Pete Rivett (agnos.ai)
William Van Woensel (Dalhousie University)
Miel Vander Sande (meemoo)
Fabio Vitali (University of Bologna)

Challenge 2: Path queries

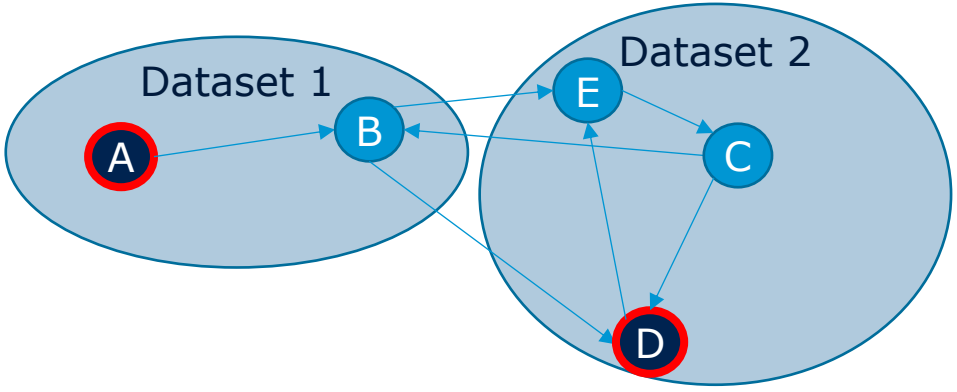
- While it is possible to do path queries in SPARQL via property path expressions, it is still not possible to **return** paths in SPARQL1.1:

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

i.e.: what is the (shortest) path `?Path` connecting `?city` and `wd:Q515`?

Challenge 2: Path queries – prototype solution

Common problem in graphs, not doable with SPARQL, but with extensions [1]:
"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) }"
```

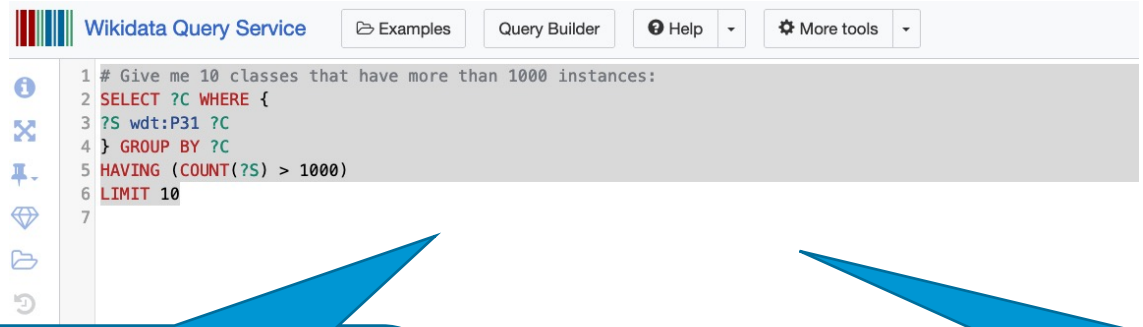
We solved this by extending SPARQL [1] with bidirectional BFS over HDT
https://bitbucket.org/vadim_savenkov/topk-pfn/

Open research question(s): e.g. But how to do this effectively in a Federated setting?

k=2

1. 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]

Challenge 3: Scalability of SPARQL endpoints?



Challenge 3.1: serve complex/long runnign queries to single users

Challenge 3.2: serve many queries to many users ***concurrently***

https://w.wiki/4mTj

more on that in Part 2

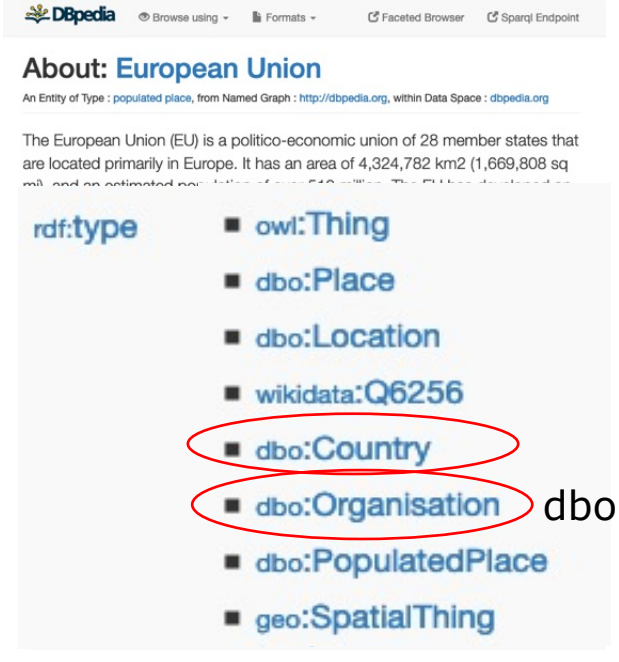
Challenge 4: Reasoning and Inconsistencies

- A lot of work has been done in the past on (deductive reasoning over KGs) in particular to retrieve implicit answers through exploiting the **OWL** and **RDFS** semantics.
- … e.g. by query rewriting or materialisation.

- Howewer:
  1) existing KGs are inconsistent
  2) some important KGs don't use OWL and RDFS

Challenge 4: Reasoning and Inconsistencies
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]

Challenge 4: Reasoning and Inconsistencies important KGs don't use OWL and RDFS

- Wikidata!

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

use "somewhat similar" properties:
wdt:P31 ~ rdf:type
wdt:P279 ~ rdfs:subClassOf