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

Axel Polleres
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 (= Resource Description Framework)
    - 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"

RDF uses URIs for the links to define their Semantics, i.e., when you look up these links, you can find out what they mean!

You find such RDF links on my Homepage! Click on http://polleres.net on http://xmlns.com/foaf/0.1/workplaceHomepage
RDF Triple:

- A triple of URLs


- ... can be seen as an edge in a Graph:
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#](http://www.w3.org/1999/02/22-rdf-syntax-ns#)) and

**RDFS Schema** (rdfs: [http://www.w3.org/2000/01/rdf-schema#](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](http://www.w3.org/1999/02/22-rdf-syntax-ns#type) (or short rdf:type)
  - [http://www.w3.org/2000/01/rdf-schema#label](http://www.w3.org/2000/01/rdf-schema#label) (or short rdfs:label)
  - [http://www.w3.org/2000/01/rdf-schema#subPropertyOf](http://www.w3.org/2000/01/rdf-schema#subPropertyOf) (or short rdfs:subPropertyOf)
  - [http://www.w3.org/2000/01/rdf-schema#subClassOf](http://www.w3.org/2000/01/rdf-schema#subClassOf) (or short rdfs:subClassOf)
  - [http://www.w3.org/2000/01/rdf-schema#domain](http://www.w3.org/2000/01/rdf-schema#domain) (or short rdfs:domain)
  - [http://www.w3.org/2000/01/rdf-schema#range](http://www.w3.org/2000/01/rdf-schema#range) (or short rdfs:range)
Other vocabularies:

- **foaf**: Prefix: [http://xmlns.com/foaf/0.1/](http://xmlns.com/foaf/0.1/) ... The "Friend-of-a-friend' vocabulary models common properties of and classes relating to Persons and social relationships. E.g.:

  Properties:
  - name
  - nickname
  - workplaceHomepage
  - knows

  Classes:
  - Agent
  - Person
  - Document
  - Image

- **schema**: Prefix: [http://schema.org/](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)
RDF Syntaxes – A simple RDF file:

simple1.nt in NTriples Syntax:

```
```
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:name "Alice Wonderland" .
:karl foaf:name "Karl Mustermann" .
:karl foaf:knows :joan.
:bob foaf:name "Robert Mustermann" .
:boob 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".
: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 available on the Web
- ... queryable via SPARQL endpoints!

**DBpedia**

1,101,215,718 triples/edges  
13,602,048,837 triples/edges

- plus useful convenience tools:
  - [http://prefix.cc/](http://prefix.cc/) ... find out common URI prefixes for formulating queries
  - [http://yasgui.triply.cc/](http://yasgui.triply.cc/) ... really nice frontend for querying SPARQL endpoints, e.g. DBpedia
  - [https://query.wikidata.org/](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.slideshare.net/Frank.van.Harmelen/adoption-of-knowledge-graphs-late-2019

https://www.dbpedia.org 2021
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

- 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":

```
http://yasgui.org/short/UVOyX8ft
```

```
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: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?

  ```sparql
  SELECT DISTINCT ?city WHERE {
    FILTER (?population > 1000000) }
  ```

- What’s this?
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/](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

**FILTERs** (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/](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/](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

```
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 yago:Italy .
  ?pop > 1000000
}
```

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

- Rome

Structured query (SPARQL):

http://yasgui.org/short/UVOyhX8ft

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
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:

```sparql
SELECT DISTINCT ?city ?Path WHERE {
  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?”

We solved this by extending SPARQL [1] with bidirectional BFS over HDT

```bash
rdf2hdt.sh -rdftype turtle testgraph.ttl testgraph.hdt
```

```bash
hdt2sparql.sh testgraph.hdt "PREFIX pfp: <java:at.ac.wu.arqext.path> SELECT * WHERE{ ?path pfp:topk (:a :d 2 )}"
```

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 running queries to single users

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

https://w.wiki/4mTj

more on that in Part 2
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.

However:
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

<table>
<thead>
<tr>
<th>Dbpedia Ontology:</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo:Agent owl:disjointWith dbo:Place.</td>
</tr>
<tr>
<td>dbo:Country rdfs:subClassOf dbo:Place.</td>
</tr>
<tr>
<td>dbo:Organisation rdfs:subClassOf dbo:Agent.</td>
</tr>
</tbody>
</table>

Challenge 4: Reasoning and Inconsistencies

important KGs don't use OWL and RDFS

- Wikidata!

```
SELECT DISTINCT ?city ?Path WHERE {
  FILTER (?population > 1000000) }
```

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

SAME
SAME

BUT
DIFFERENT