Biomedical semantics, information retrieval and knowledge discovery - (11)

SPARQL

Dr. Dietrich Rebholz-Schuhmann
Dr. Leyla Jael Garcia Castro

December 19, 2020

Outline

Objectives and Learning Outcomes

Objectives

- We will query the Semantic Web with SPARQL, we will show some examples and explain the different query elements.

Learning outcomes

- Naming the elements part of a SPARQL query.
- Explaining the relation between SPARQL and RDF / RDF-Schema.
- Using SPARQL queries to retrieve data from RDF graphs.
- Comparing SPARQL queries as different possibilities can be used to answer the same question.

Triple stores

What is a triple store?

A triple store is a non-SQL graph-based database for the storage and retrieval of RDF triples.

- Multiple options, from free to paid, from small to big. Some of the most known triple stores (aka knowledge graph stores) are:
  Virtuoso, StarDog, GraphDB and Apache Jena Fuseki
- Commonly support all the RDF available formats/syntaxes (i.e. RDF/XML, JSON-LD, Turtle, N3)
- You can upload your files and modify them later or create them directly on the triple store
- You will have classes and properties but also instances reflecting the real world wrt your domain
- They support SPARQL queries!!!

Property graphs

Triple stores are not the only "semantic/graph-based" option

- Graph properties are also popular but they do not support SPARQL queries (at least not natively)
- The most popular property graph nowadays is probably Neo4J but there are others
- Property graphs come with their own query language (e.g. Cypher for Ne4J), there is no standard
- In addition to nodes and edges, you have properties/attributes on top of either of them (attributes are commonly attached to nodes only in RDF —there is a way to have them on edges as well but it is a bit convoluted: reification)

SPARQL – a query language for the Semantic Web

What is SPARQL?

SPARQL is the "SPARQL Protocol And RDF Query Language".

- SPARQL uses RDF syntax to query the content of a collection of RDF statements.
- The RDF statements have to be accessible from one or several SPARQL endpoints, which is an engine that processes and loads the RDF statements, and which offers an interface to specify queries.
- The SPARQL endpoint reads all RDF statements, indexes them but also generates an internal graph structure for the traversal of the RDF statements.
- SPARQL syntax is similar to other database query languages (e.g., SQL) but uses also triples as basic constructs.
- Be reminded that RDF makes extensive use of URIs which enables the identification of unique data entries even if they are distributed over different databases and even sites.

Composition of SPARQL queries

- The basic query retrieves the subject (`?s`), the object (`?o`), the predicate (`?p`) or any other variable (`?v`) specified in the query which satisfies one or more of the given RDF statements:

  ```
  SELECT ?v WHERE { pizza:Teig pizza:hatZutat ?v . }
  ```

- All prefixes for the namespaces have to be properly specified (according to RDF).
- Further syntactical constructs enable to specify the SPARQL query more in detail:
  - `FROM` + URL: denotes one or more SPARQL endpoints for the retrieval.
  - `OPTIONAL`: allows to specify triples that specify further details, but do not restrict the retrieval if the RDF statements cannot be satisfied.
  - `LIMIT` + Number: Retrieve/Produce only a limited set of entries.
  - Blank nodes (also brackets) may be used as well.

Triple stores Example

Example data form a triple store: `lit` and `geo` are special namespaces.

Subject	Predicate	Object
lit:Shakespeare	lit:wrote	lit:AsYouLike
lit:Shakespeare	lit:wrote	lit:HenryV
...	...	...
geo:Scotland	geo:partOf	geo:UK
geo:England	geo:partOf	geo:UK
...	...	...
lit:Shakespeare	rdf:type	lit:Playwright
lit:Ibsen	rdf:type	lit:Playwright
lit:Simon	rdf:type	lit:Playwright
...	...	...

rdf:Property, Assertions

The list of properties that have been defined for the above-mentioned triple store. All properties are derived from `rdf:Property`.

Subject	Predicate	Object
lit:wrote	rdf:type	rdf:Property
geo:partof	rdf:type	rdf:Property
bio:married	rdf:type	rdf:Property
bio:lifedIn	rdf:type	rdf:Property
go:isIn	rdf:type	rdf:Property

SPARQL queries (1)

`?s` queries for the subject, i.e. the author in the statement, and the predicate (`?p`) or the object (`?o`).

1. Querying the subject:
   `?s lit:wrote lit:Hamlet .`
2. Querying the predicate:
   `lit:Shakespear ?p lit:TwelfthNight .`
3. Querying the object:
   `lit:Shakespear lit:wrote ?o .`

SPARQL queries (2)

More complex queries identify several components in the statement and bind the same variable in different statements, i.e. `?person` queries for the author of `lit:Hamlet` married to a person known in the database (`?s`).

```
[?person bio:married ?s .
 ?person lit:wrote lit:Hamlet . ]

[?person bio:livedIn ?place .
 ?place geo:isIn geo:England .
 ?person lit:wrote lit:Hamlet . ]
```

SPARQL queries (3)

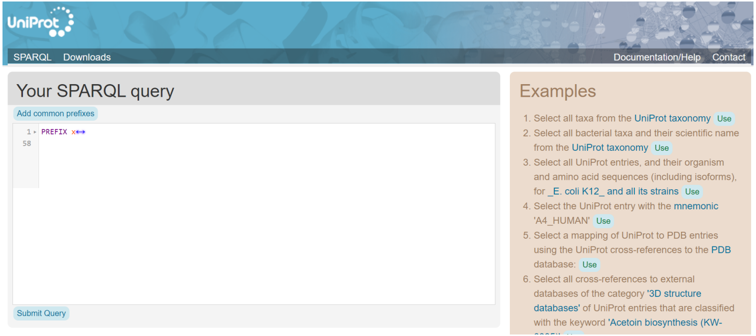
A very unspecific query with no fixed arguments queries for the content of the whole database.

```
CONSTRUCT [ ?s ?p ?o ]
WHERE [ ?r rdf:subject ?s .
            ?r rdf:predicate ?p .
            ?r rdf:object ?o . ]
```

Biomedical SPARQL endpoints

UniProt SPARQL endpoint (1)

`https://sparql.uniprot.org/`



UniProt SPARQL endpoint (2)

Your SPARQL query

Add common prefixes

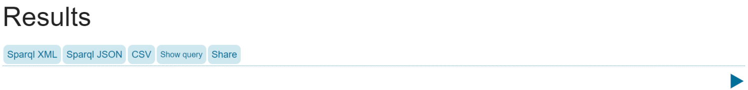
```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX taxon: <http://purl.uniprot.org/taxonomy/>
PREFIX up: <http://purl.uniprot.org/core/>
SELECT ?protein ?organism ?isoform ?aa_sequence
WHERE
{
    ?protein a up:Protein .
    ?protein up:organism ?organism .
    ?organism rdfs:subClassOf taxon:83333 .
    ?protein up:sequence ?isoform .
    ?isoform rdf:value ?aa_sequence .
}
```

Submit Query

Examples

1. Select all taxa from the UniProt taxonomy Use
2. Select all bacterial taxa and their scientific name from the UniProt taxonomy Use
3. Select all UniProt entries, and their organism and amino acid sequences (including isoforms), for _E. coli K12_ and all its strains Use
4. Select the UniProt entry with the mnemonic 'A4_HUMAN' Use
5. Select a mapping of UniProt to PDB entries using the UniProt cross-references to the PDB database: Use
6. Select all cross-references to external databases of the category '3D structure databases' of UniProt entries that are classified with the keyword 'Acetoin biosynthesis (KW-

UniProt SPARQL endpoint (3)



protein	organism	isoform	aa_sequence
http://purl.uniprot.org/uniprot/V6BC57	http://purl.uniprot.org/taxonomy/1110693	http://purl.uniprot.org/isoforms/V6BC57-1	"ANDENYALAA"xsd:string
http://purl.uniprot.org/uniprot/V6BC57	http://purl.uniprot.org/taxonomy/1110693	http://purl.uniprot.org/isoforms/V6BC57-1	"ANDENYALAA"xsd:string
http://purl.uniprot.org/uniprot/A0A0J9X1X4	http://purl.uniprot.org/taxonomy/1110693	http://purl.uniprot.org/isoforms/A0A0J9X1X4-1	"MRIILLGAPGAGKGTQAQFIMEKYGIP(
http://purl.uniprot.org/uniprot/A0A0J9X1X4	http://purl.uniprot.org/taxonomy/1110693	http://purl.uniprot.org/isoforms/A0A0J9X1X4-1	"MRIILLGAPGAGKGTQAQFIMEKYGIP(
http://purl.uniprot.org/uniprot/A0A0H2UKY9	http://purl.uniprot.org/taxonomy/1403831	http://purl.uniprot.org/isoforms/A0A0H2UKY9-1	"GPASSNLIKQLQERGLVAQVTDEEALA
http://purl.uniprot.org/uniprot/A0A0H2UKY9	http://purl.uniprot.org/taxonomy/1403831	http://purl.uniprot.org/isoforms/A0A0H2UKY9-1	"GPASSNLIKQLQERGLVAQVTDEEALA
http://purl.uniprot.org/uniprot/A0A0H2UKZ0	http://purl.uniprot.org/taxonomy/1403831	http://purl.uniprot.org/isoforms/A0A0H2UKZ0-1	"GPASSNLIKQLQERGLVAQVTDEEALA

UniProt Schema

`https://www.uniprot.org/core/`

UniProt RDF schema ontology

Navigation panel

Classes (177)	up:Absorption_Annotation
Properties (161)	
Object properties (77)	faldo:location
Datatype properties (84)	up:abstract
Properties (62)	up:API

Ontology description

http://purl.uniprot.org/core/ (rdf:type owl:Ontology)	
rdfs:label	UniProt RDF schema ontology
rdfs:comment	Properties and classes used for protein annotation. xsd:string
owl:imports	http://www.w3.org/2004/02/skos/core
owl:imports	foaf:
owl:imports	http://biohackathon.org/resource/faldo
owl:imports	http://www.w3.org/1999/02/22-rdf-syntax-ns

BioPortal SPARQL endpoint (1)

`http://sparql.bioontology.org/`

BioPortal SPARQL endpoint (2)

BioPortal SPARQL Examples

This page contains interactive SPARQL examples. More documentation is avalaible in our Wiki documentation

Get all graphs IDs that contain ontology content. test this query

```
PREFIX meta: <http://bioportal.bioontology.org/metadata/def/>

SELECT DISTINCT ?vrtID ?graph
WHERE {
    ?vrtID meta:hasVersion ?version .
    ?version meta:hasDataGraph ?graph .
}
```

Note: This public SPARQL endpoint only contains the latest version of each BioPortal ontology. Graph IDs can be used to target queries to a specific ontology. For example, to query SNOMED, the graph ID is http://bioportal.bioontology.org/ontologies/SNOMEDCT

Get term direct sub-classes with labels. SNOMED Example. test this query

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX snomed-term: <http://purl.bioontology.org/ontology/SNOMEDCT/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
SELECT DISTINCT ?x ?label
FROM <http://bioportal.bioontology.org/ontologies/SNOMEDCT>
WHERE
{
    ?x rdfs:subClassOf snomed-term:363664003 .
    ?x skos:prefLabel  ?label.
}
```

Wikidata – Semantic data for Wikipedia

What is Wikidata

Wikidata is a free, collaborative, multilingual, secondary database, collecting structured data to provide support for Wikipedia, Wikimedia Commons, the other wikis of the Wikimedia movement, and to anyone in the world.

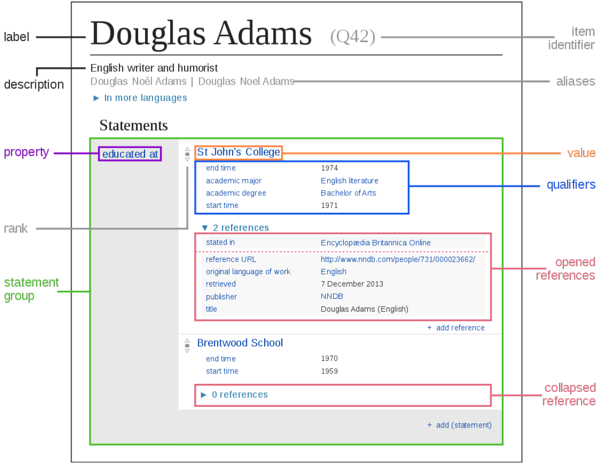
What does this mean?

Let's look at the opening statement in more detail:

- **Free.** The data in Wikidata is published under the Creative Commons Public Domain Dedication 1.0, allowing the reuse of the data in many different scenarios. You can copy, modify, distribute and perform the data, even for commercial purposes, without asking for permission.
- **Collaborative.** Data is entered and maintained by Wikidata editors, who decide on the rules of content creation and management. Automated bots also enter data into Wikidata.
- **Multilingual.** Editing, consuming, browsing, and reusing the data is fully multilingual. Data entered in any language is immediately available in all other languages. Editing in any language is possible and encouraged.
- **A secondary database.** Wikidata records not just statements, but also their sources, and connections to other databases. This reflects the diversity of knowledge available and supports the notion of verifiability.
- **Collecting structured data.** Imposing a high degree of structured organization allows for easy reuse of data by Wikimedia projects and third parties, and enables computers to process and "understand" it.
- **Support for Wikimedia wikis.** Wikidata assists Wikipedia with more easily maintainable information boxes and links to other languages, thus reducing editing workload while improving quality. Updates in one language are made available to all other languages.
- **Anyone in the world.** Anyone can use Wikidata for any number of different ways by using its application programming interface.

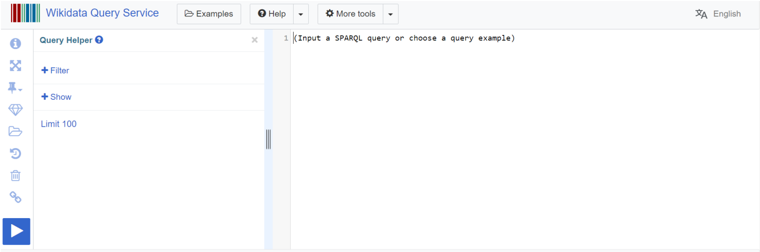
Contents [hide]

Wikidata model

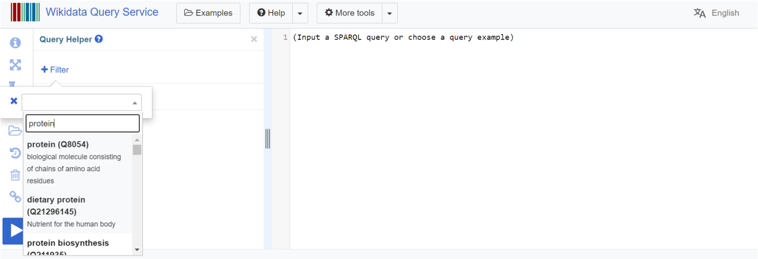


Querying wikidata (1)

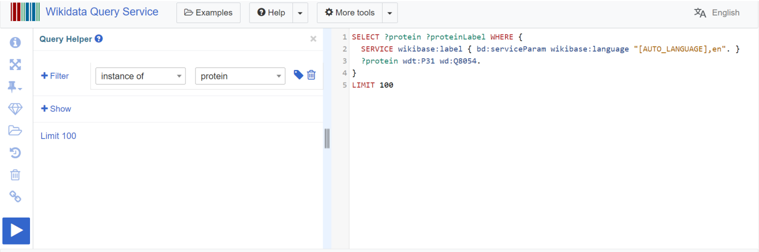
`https://query.wikidata.org/`



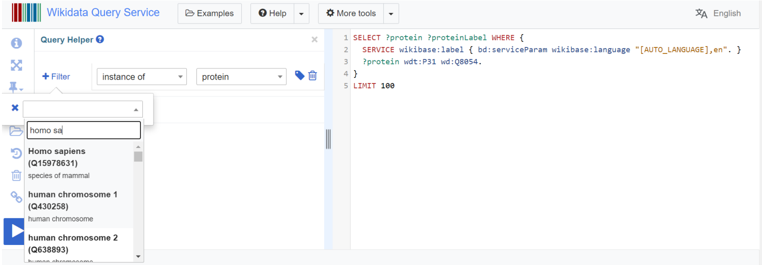
Querying wikidata (2)



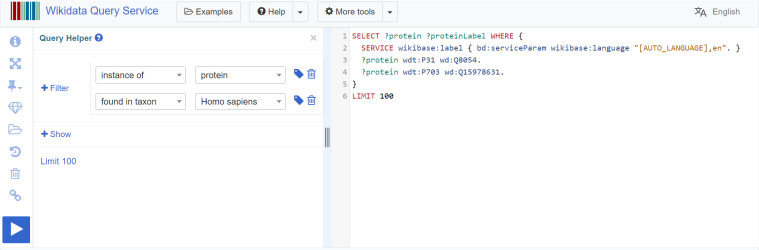
Querying wikidata (3)



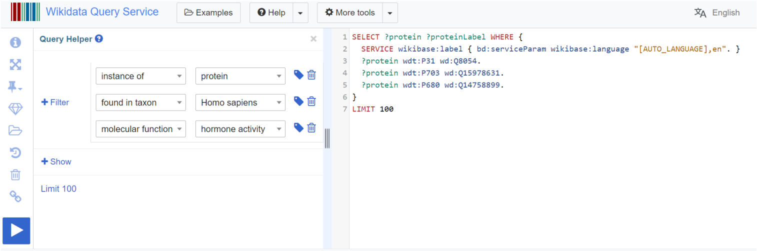
Querying wikidata (4)



Querying wikidata (5)



Querying wikidata (6)



Querying wikidata (7)

100 results in 668 ms | Code | Download | Link

protein	proteinLabel
wd:Q63398	Chromogranin B
wd:Q170617	glucagon
wd:Q192191	Prolactin
wd:Q202476	parathyroid hormone
wd:Q218706	erythropoietin
wd:Q223739	leptin
wd:Q246978	Pancreatic polypeptide
wd:Q290293	Peptide YY
wd:Q302495	Natriuretic peptide A
wd:Q315860	Calcitonin related polypeptide alpha
wd:Q318896	Gonadotropin releasing hormone 1

Querying wikidata (8)

leptin (Q223739)

mammalian protein found in Homo sapiens — edit

leptin (murine obesity homolog) | leptin (obesity homolog, mouse) | obesity factor | LEP | leptin | obese protein | obese, mouse, homolog of

Statements

instance of	protein 1 reference	edit
		+ add value
found in taxon	Homo sapiens 1 reference	edit
		+ add value
encoded by	LEP 1 reference	edit
		+ add value
molecular function	hormone activity determination method: IEA, IBA 3 references	edit

Querying wikidata (9)

wikidata.org/wiki/Special:EntityData/Q223739.ttl

```
        schema:about wd:Q223739 ;
        schema:inLanguage "ru" ;
        schema:isPartOf <https://ru.wikipedia.org/> ;
        schema:name "Лептин"@ru .

<https://ru.wikipedia.org/> wikibase:wikiGroup "wikipedia" .

<https://sh.wikipedia.org/wiki/Leptin> a schema:Article ;
        schema:about wd:Q223739 ;
        schema:inLanguage "sh" ;
        schema:isPartOf <https://sh.wikipedia.org/> ;
        schema:name "Leptin"@sh .

<https://sh.wikipedia.org/> wikibase:wikiGroup "wikipedia" .

<https://sl.wikipedia.org/wiki/Leptin> a schema:Article ;
        schema:about wd:Q223739 ;
        schema:inLanguage "sl" ;
        schema:isPartOf <https://sl.wikipedia.org/> ;
        schema:name "Leptin"@sl .

<https://sl.wikipedia.org/> wikibase:wikiGroup "wikipedia" .
```

Wikidata namespaces and links

- Entities (S - P - O):
  `http://www.wikidata.org/entity/Q7240673`
  `http://www.wikidata.org/entity/P17`
- Items (s, O): https://www.wikidata.org/wiki/
  `https://www.wikidata.org/wiki/Q7240673`
- Properties: https://www.wikidata.org/wiki/Property:
  `https://www.wikidata.org/wiki/Property:P17`
- Content negotiation to get the actual data for items:
  `https://www.wikidata.org/wiki/Special:EntityData/Q724067.rdf` (or .ttl or .json)

Summary

Summary

- SPARQL queries have to correspond to the semantic representation of the triple store to produce results (= fitting to the contained RDF statements).
- The syntax of the SPARQL query is very generic: RDF syntax + query language keywords.
- The SPARQL query is fitted to all statements (in the graph) and those fitting to the query "slots" are being produced.
- Caution: Since the query can be seen as constraints in the specification of all statements, the properties (and thus statements) fitting to the intersection of all constraints may be empty (i.e., no results are being produced)
  $\Rightarrow$ It is – occasionally – advantageous to put constraints as optional constraints to the query.

So where are we now?

- With regards to biomedical semantics, we have learnt about SPARQL resources (UniProt, BioPortal and Wikidata) useful to get insights on biomedical data
- With regards to data integration we have seen a practical example with Wikidata. New data is added to Wikidata on collaborative/community basis.
- SPARQL also supports federated queries (not covered here) so we can integrate ourselves data from multiple sources.
- With regards to knowledge discovery we have more elements now to work with.