|  |
| --- |
| **Visual Exploitation of SPARQL Endpoints[[1]](#footnote-1)** |
|  |
| **Daria Schmidt** (5684379) **and Nkonye Gbadegoye** (5410570) |
| *Eberhard Karl University Tübingen*  Seminar: Comparing Query Languages  WS21/22  Lecturer: Thorsten Trippel |

Abstract

Nowadays data exploration has a crucial role and not only IT companies are interested in it. A lot of enterprises, government and educational institutions etc., where non-tech users are also engaged, use a huge amount of data every day. Therefore, there is a need for efficient and accessible visualization tools for better understanding of the data. In this research we compare several latest programs that enable graph visualization of SPARQL endpoints. The goal of this work is to give a small survey of tools and find the limitations. Nonetheless, the findings are relevant for both the users and the developers, who are interested in more user-friendly design and efficient approaches.

Introduction

The world is governed by data. The potential of data and its importance grew increasingly in these last few decades. The urgency and importance of knowledge management is recognized by most professionals involved in corporate governance and IT technologies for management purposes. Universities, enterprises and individuals use a great amount of data from different sources and for different purposes every day. And that is why it is important to have the possibility to manage this data, to query it and to visualize the information for better understanding. The focus is now not only on high levels of reliability, accessibility and security of the databases, but also on the systems that can help to explore, unify, integrate data in real-life applications and to provide the relationships within it.

Currently, there are several approaches to storing knowledge graphs. One of these approaches involves representing the graph as a semantic RDF graph. In this approach, storage and processing relies more on the edges of the graph. In this project we focus on the query language SPARQL which is able to effectively use the graph model of knowledge representation and has basic inference mechanisms. The tasks of the user are to query the data and create a SPARQL endpoint – to ask for specific data.

There are different languages and systems for describing and managing databases, however, the most promising presents a visual approach that allows the user to create directly visualization of SPARQL endpoints, which help to clearly formulate and explain the nature and structure of phenomena. Visual models, such as graphs, have a special cognitive power presenting the resources of cognitive graphics to structure information. The diversity and complexity of information the endpoints contain challenges the development of such systems. The developers search for different solutions on how to make their programs be the best on the market – the programs that support querying, exploration of raw, unique data from many endpoints and have case-specific visualization features.

Nowadays there is a necessity to research the requirements in data exploration and visualization, as well as limitations of existing tools. Modern systems face numerous challenges, some of them can process only small data sets, others cannot offer all useful features in order to handle a variety of tasks in the contemporary world ([Bikakis et al., 2016](#Bikakis)). In this work we compare several programs that allow SPARQL endpoints exploitation and visualization in order to find some advantages and limitations of those tools.

Related work

At the moment, a lot of knowledge is stored in Resource Description Format (RDF). Many institutions use such a format as it allows the data to be linkable and exchangeable ([Chawuthai et al., 2016](#Chawuthai)). But the data has minimum value if it is impossible to find relevant information in it. For this reason, the interest for visual and interactive approaches to data exploration only increases. The current goals are 1) to investigate the concepts of application domain via ontology depiction; 2) to explore RDF Graphs; and 3) to analyze the occurrences based on the types, classes ([Menin et al., 2021](#Menin)).

The exploration of the datasets with unknown structure and without appropriate knowledge about Semantic Web can be rather struggling ([Chawuthai et al., 2016](#Chawuthai)). Many researches were conducted in order to find out what problems the users can get using different tools that explore and visualize RDF graphs. At first, we should understand that the application should be understandable for both groups such as IT-Specialists and mainstream end users. Different government institutions, media and public people can use Linked Open Data for many purposes ([Dadzie et al., 2011](#Dadzie)). Non-tech users are not able to visualize suitable information from RDF format, since the data is highly connected and requires complex processing. The users can only get lost in the huge amount of information ([Chawuthai et al., 2016](#Chawuthai)). It is inevitable to write correct construction of queries and have understanding of the construction of dataset logic. In the data retrieval process the information can originate from different endpoints that only expands the complexity of the task and may lead to reduction of search quality ([Menin et al., 2021](#Menin)). Therefore, it is required to develop user-friendly and high-quality applications which are able to visualize SPARQL endpoints and help to retrieve useful information.

In order to make the exploration in SPARQL simpler and more understandable for all users, some developers create web-based viewers. For example, [SPARQL playground[[2]](#footnote-2)](https://sparql-playground.sib.swiss) is publicly available, provides a user-friendly environment and has either online or offline versions. The user can work with and query any RDF data and has the possibility to visualize it in turtle format. But the limitation of such an API is that there is no graph visualization option ([Bonduel et al., 2018](#Bonduel)).

Numerous applications that can be used for ontology engineering and data exploration were developed in the last years. But not all of them can stay on the market for a long period of time. There are several reasons for this: 1) lack of user-friendly functionality; 2) lack of feedback and communication between the user and the developer or a weak support of the developer; 3) the applications cannot be longer maintained or are very complicated for ontology engineering ([Bonduel et al., 2018](#Bonduel)). The researchers try to analyze what requirements for endpoints visualization each application should fulfil. Many different visualization techniques in different domains as well as purposes of their users are studied in order to create some visualization and design guidelines for developers ([Tufte, 1990](#Tufte); [Dadzie et al., 2011](#Dadzie)).

Visualization of SPARQL endpoints

In this section we would like to describe several tools that provide the possibility of graph visualization of SPARQL endpoints and focus on some limitations of the programs. For this research we studied the following programs: JSON-LD playground, Stardog Studio, Protégé, Gephi and GraphDB.

* 1. JSON-LD playground

The [JSON-LD playground](https://json-ld.org) [[3]](#footnote-3) is a web-based JSON-LD viewer. The JSON-LD is publicly available online API. The user has the possibility to debug, normalize and share JSON-LD data. With the help of expansion, the user can remove "@context"

from the file and make the data structure more regular. The difference between expanded and compacted forms lies only in "@context"[[4]](#footnote-4) (set of rules for interpreting the document). This feature we also needed in our research in order to expand the JSON-LD format data for Protégé. Also, this viewer supports graph visualization of endpoints.

To show this feature in work we analyzed CMDI.jsonld and CMDI\_05.jsonld files. Figure 01 illustrates graph visualization of CMDI.jsonld. The data is rather small and simple, so we get a clear graph with all information that fits in a small output window. Figure 02 visualizes CMDI\_05.jsonld data. The graph is more complex and it is impossible to see all the information we can retrieve from the data. The navigation in such graphs is rather tricky: it is possible to expand the nodes, but it is difficult to get the information the user needs. There is no querying support, so the output is a complex graph, which is sometimes overloaded with information. Another limitation of this API is difficulty in data extraction. The user cannot save processed data or visualization output in a separate file.

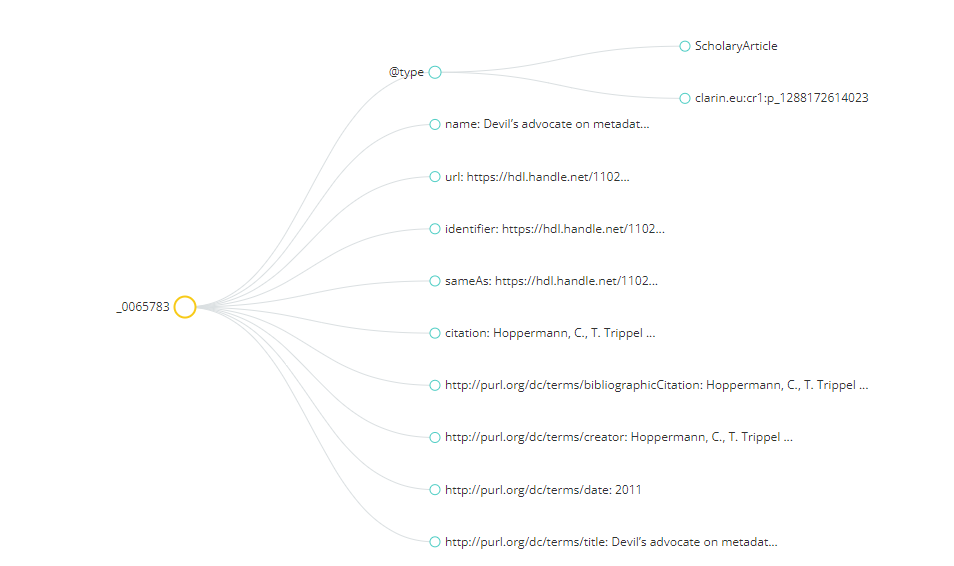


Figure 01: JSON-LD playground visualization of CMDI.jsonld.

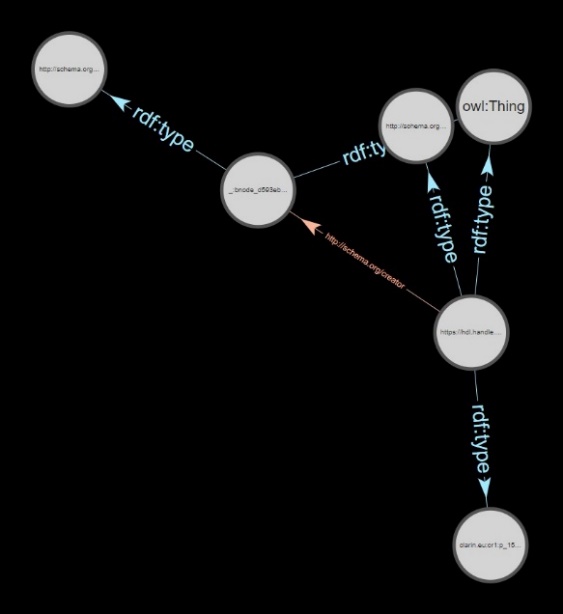


Figure 03: CMDI.jsonld in Stardog Studio

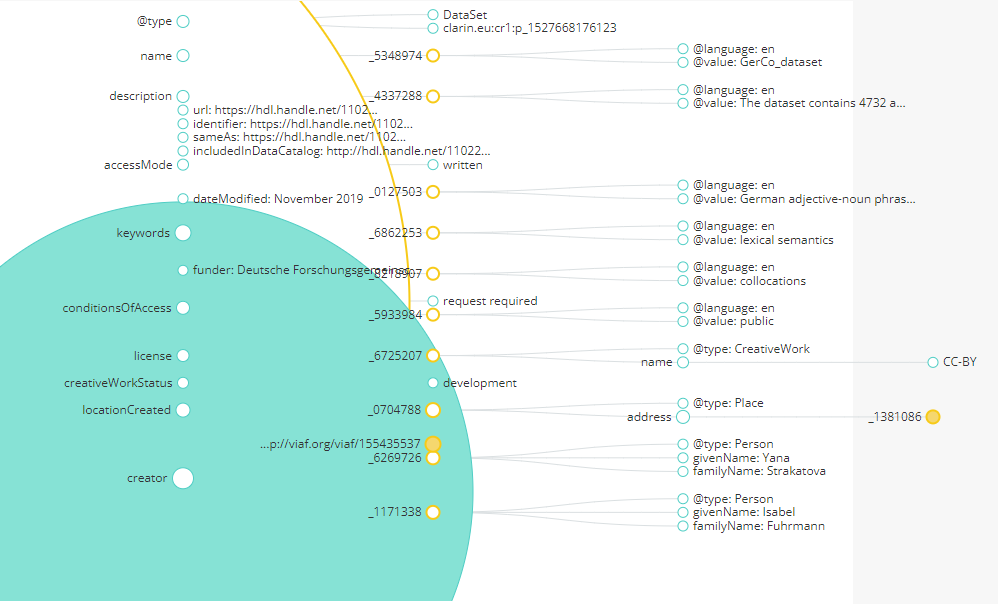


Figure 02: JSON-LD playground visualization of CMDI\_05.jsonld

Stardog Studio

Stardog Studio is a commercial tool that helps to manage data fabric and to create its own database. The program is free, the user should only proceed with the registration and install Stardog Studio locally. The requirement is that the user knows how to apply service from the W3C SPARQL federated query recommendation ([Xiao et al., 2019](#Xiao)). The tool provides the user with a set of common features that help to visualize large amounts of complex data. It is possible to query and filter out less important data. The feature called Stardog Integrity Constraint Validation validates data stored in the database. And with the help of Machine Learning and predictive analytics it is possible to fill out some missing data. But even these high-qualitative and essential functions cannot prevent the user from facing some limitations of the program. For example, Stardog does not support every type of data.

We proceeded to query CMDI\_05.jsonld file. In order to make it work in Stardog the user should convert “.jsonld” to “.ttl”, “.rdf” or “.xml” formats, that is possible with help of some online converters, for example [EASYRDF](https://www.easyrdf.org/converter)[[5]](#footnote-5). After loading the data, the user can query and visualize it with the help of different query languages (for our project we used SPARQL).

To visualize the results, it is required to select some IRI in the output after running the query. Stardog creates the schema[[6]](#footnote-6), the elements of which are classes and relations. They and also datatype properties, some information about the instances of classes can be shown (see Figure 03). Stardog Studio allows the user to change the visual effects of the schema, it is possible to move the nodes and navigate them. The next limitation we could find here is zooming in on the schema. If the data is complex and the user becomes a large number of visualized endpoints, it is very difficult to recognize and analyze something. To show this limitation we used Music\_Data.ttl. After querying the data, we became the list of persons who work in the music branch. The output of the visualization of the information about :Bobby\_Robinson is quite simple and easy to understand (see Figure 04), but on the other side the visualization of :David\_Bowie data cannot help us very much. The graph is very complex (see Figure 05) and the user can not recognize some information from it, because of the impossibility to zoom in on the output. But still there are some advantages in using Stardog Studio: 1) processing of large and complex data; 2) the possibility to save the queries and reuse them with different databases; 3) querying and visualization output extraction for further use.

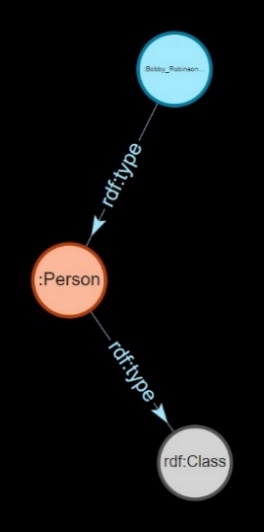
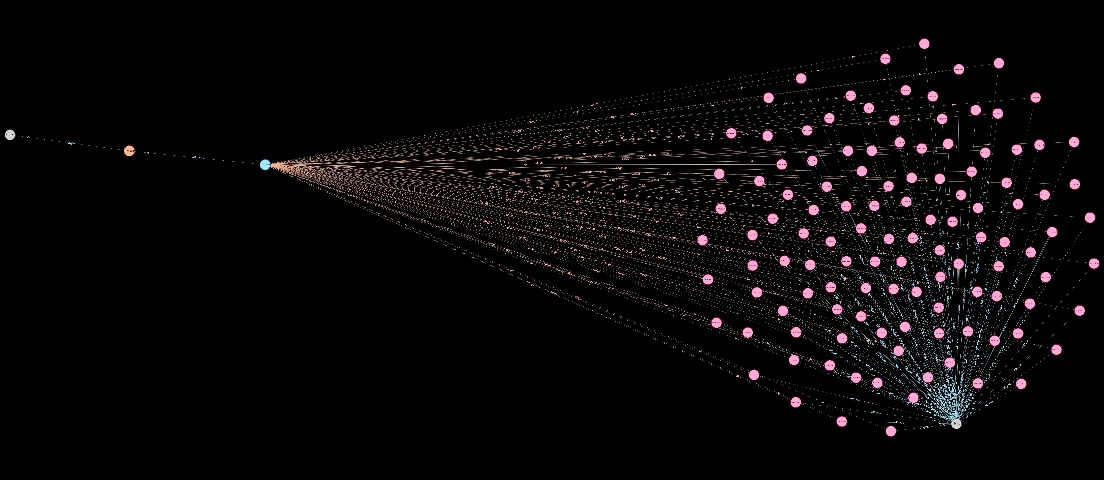
 

Figure 04 (left): Visualization of Bobby\_Robinson data from Music\_Data.ttl in Stardog Studio. Figure 05 (right): Visualization of :David\_Bowie data from Music\_Data.ttl in Stardog Studio.

* 1. Protégé

A team of researchers from Stanford University (California, USA) developed the Protégé ontology editor and its web interface ([Musen et al., 2015](#Musen)). The tool is free to use, but requires registration for a personal account. Protégé is the most widely-used software for building ontologies and exists in a variety of frameworks. But not only building ontologies function is attractive for the customers, it is possible to load and explore existing files as well. It provides the user with the query services and the visualization support of the file.

A lot of open source and commercial [Protégé plugins](https://protegewiki.stanford.edu/wiki/Visualization) [[7]](#footnote-7) enhance the Protégé application. Only for visualization purposes there are more than 20 plugins. In this project we used [OWLViz](https://protegewiki.stanford.edu/wiki/OWLViz)[[8]](#footnote-8) and [OntoGraf](https://protegewiki.stanford.edu/wiki/OntoGraf) [[9]](#footnote-9) visualizer. But at the beginning we have already faced the first limitation of the program – it does not support all file formats. To proceed the work with JSON-LD format, the user should use an expanded form. In order to achieve our goal, we converted our CMDI\_05.jsonld to CMDI\_05\_expanded.jsond with the help of JSON-LD playground. Documentation, tutorial videos, help for installation of plugins and their settings are provided. We can call this tool as user-friendly as even non-tech users can explore and visualize the data following the given instruction. Nonetheless, we found one of the shortcomings of this program mentioned in the paper of [Bonduel et al.](#Bonduel) – the lack of user-friendly functions concerning feedback.

Figure 06 shows the visualization of CMDI\_05\_expanded.jsonld with the help of OWLViz. It provides the view of class hierarchy and navigation in the graph. The colours help to distinguish primitive classes from defined classes and computed changes. OWLViz supports zooming in and out function, but there is no possibility to move the nodes. OWLViz does not have such a rich variety of visualization layouts as OntoGraf; here the user can choose only between Left to Right and Top to Bottom layouts. The user can save the results in graphic formats including PNG, JPEG, and SVG.

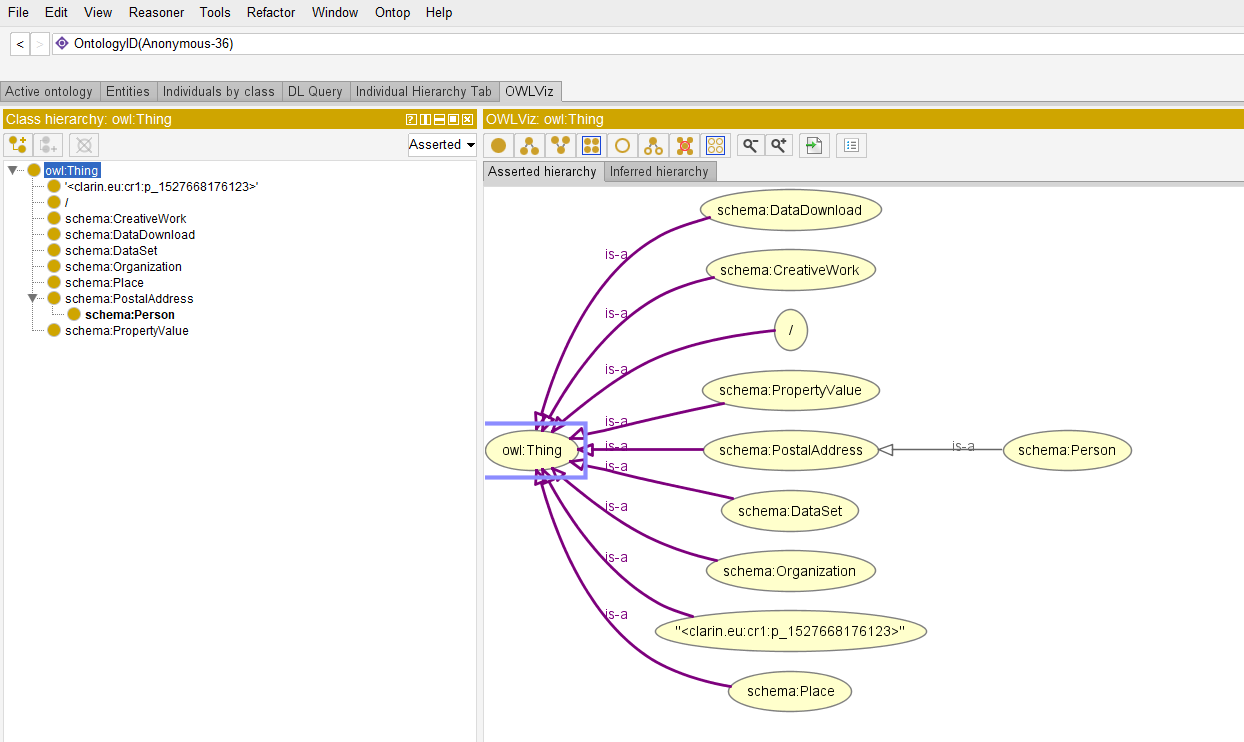


Figure 06: Protégé visualization with OWLViz

The visualization with OntoGraf gives similar results. Here it is possible to navigate and move the relationships of data, to filter the relationships and node types in order to get only the information we need (very useful feature for large datasets). The user has a variety of layout options: alphabetical, radial, spring, tree-vertical, tree-horizontal, vertical directed, horizontal directed. Figure 07 illustrates the visualization with radial layout. The tooltips show some information about URI and Annotations or some other extensive details about classes and individuals. For better understanding of the graph, the user can zoom in and out or move the nodes in different places.

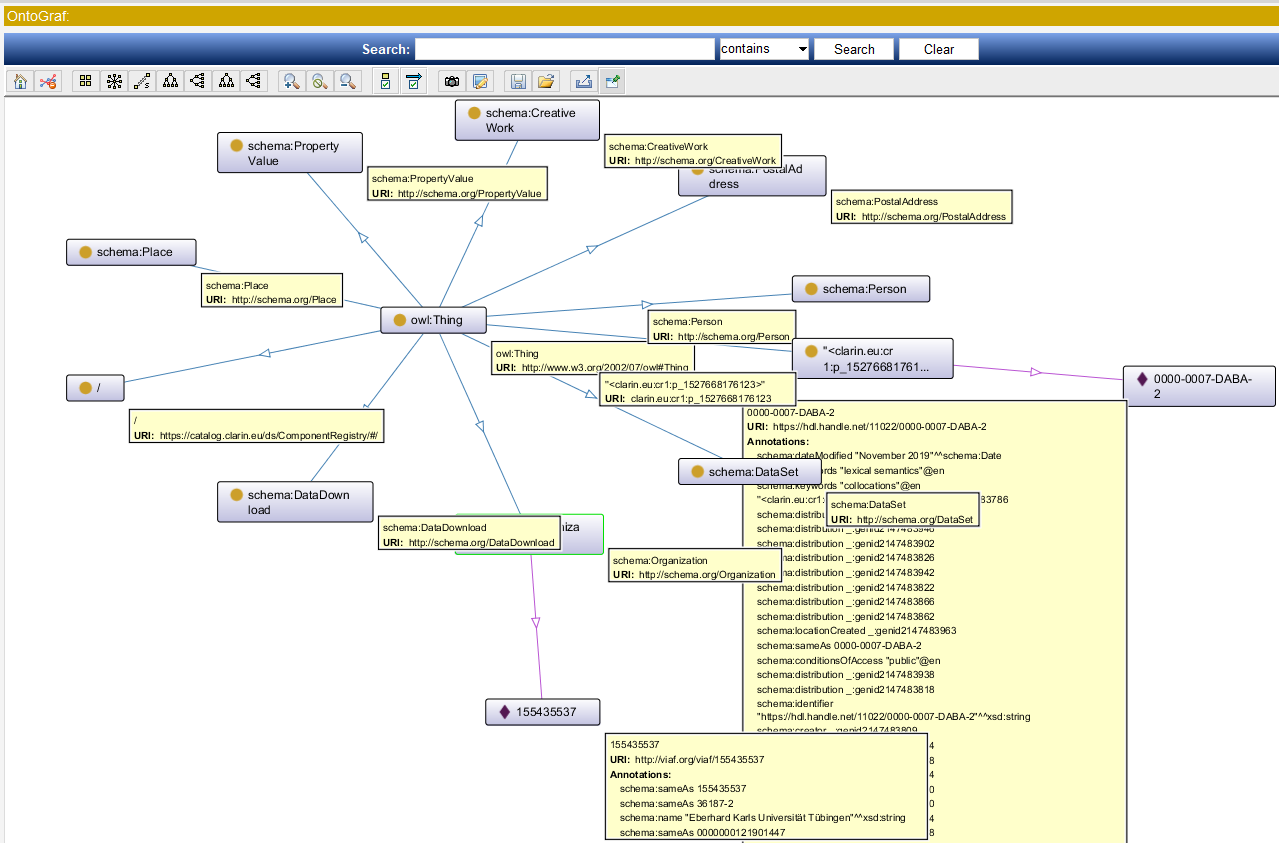


Figure 07: Protégé visualization with OntoGraf

Gephi

…

GraphDB

…

Discussion

…

Conclusion

…

References

Bikakis, N., Sellis, T.K. 2016. [Exploration and visualization in the web of big linked data: A survey of the state of the art.](http://ceur2ws.org/Vol-1558/paper28.pdf) *In: Proceedings of the Workshops of the EDBT/ICDT 2016.* Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016

Bonduel, M., Rasmussen, M.H., Pauwels, P., Vergauwen, M., Klein, R. 2018. [SPARQL-visualizer: A Communication Tool for Collaborative Ontology Engineering Processes](https://www.researchgate.net/publication/326294354_SPARQL-visualizer_A_Communication_Tool_for_Collaborative_Ontology_Engineering_Processes). 10.13140/RG.2.2.13687.93603.

Chawuthai R., Takeda H. 2016. [RDF Graph Visualization by Interpreting Linked Data as Knowledge*.*](https://doi.org/10.1007/978-3-319-31676-5_2) In: *Qi G., Kozaki K., Pan J., Yu S. (eds) Semantic Technology. JIST 2015. Lecture Notes in Computer Science*, vol 9544. Springer, Cham.

Dadzie, A. and Rowe, M. 2011. [Approaches to visualising Linked Data: A survey](https://www.researchgate.net/publication/220575556_Approaches_to_visualising_Linked_Data_A_survey). Semantic Web. 2. 89-124. 10.3233/SW-2011-0037.

Menin, A., Zucker, C.F., Corby, O., Dal Sasso Freitas, C., Gandon, F., et al. 2021. [From Linked Data Querying to Visual Search: Towards a Visualization Pipeline for LOD Exploration](https://www.scitepress.org/Papers/2021/106546/106546.pdf). *WEBIST 2021 - 17th International Conference on Web Information Systems and Technologies*, Oct 2021, Online Streaming, France. (10.5220/0010654600003058)

Musen, M. A. and Protégé Team. 2015. [The Protégé Project: A Look Back and a Look Forward](https://doi.org/10.1145/2757001.2757003). AI matters, 1(4), 4–12.

Tufte, E. R. 1990. [Envisioning Information.](https://journals.sagepub.com/doi/10.1177/027046769301300113) *Graphics Press: Cheshire*, Connecticut, 1990. ISBN 0-9613921-1-8

Xiao, G., Ding, L., Cogrel, B., Calvanese, D. 2019. [Virtual Knowledge Graphs: An Overview of Systems and Use Cases. Data Intelligence](https://doi.org/10.1162/dint_a_00011); 1 (3): 201–223.

1. [https://github.com/Visualisation-of-SPARQL-Endpoints](https://github.com/Nkonye-okh/Visualisation-of-SPARQL-Endpoints) [↑](#footnote-ref-1)
2. https://sparql-playground.sib.swiss [↑](#footnote-ref-2)
3. https://json-ld.org [↑](#footnote-ref-3)
4. https://www.w3.org/TR/2014/REC-json-ld-api-20140116/#dfn-context [↑](#footnote-ref-4)
5. https://www.easyrdf.org/converter [↑](#footnote-ref-5)
6. [https://github.com/.../Startdog\_schema\_for\_visualization.txt](https://github.com/Nkonye-okh/Visualisation-of-SPARQL-Endpoints/blob/main/queries%20and%20results/Startdog_schema_for_visualization.txt) [↑](#footnote-ref-6)
7. https://protegewiki.stanford.edu/wiki/Visualization [↑](#footnote-ref-7)
8. https://protegewiki.stanford.edu/wiki/OWLViz [↑](#footnote-ref-8)
9. https://protegewiki.stanford.edu/wiki/OntoGraf [↑](#footnote-ref-9)