# Visual Exploitation of SPARQL Endpoints[1]

**Daria Schmidt** (5684379) **and Nkonye Gbadegoye** (5410570)
*daria.schmidt@student.uni-tuebingen.de, nkonye.gbadegoye@student.uni-tuebingen.de*

*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, governments, 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 a better understanding of the data. In this research, we compare several latest programs that enable graph visualization of SPARQL endpoints. The goal of this study is to give a preliminary survey of these tools, examining their strengths as well as limitations. The findings are relevant for both users and developers, who are interested in more user-friendly design and efficient approaches.

## 1 Introduction

The goal of our research project is to explore the options of the following tools that allow SPARQL endpoints exploitation and visualization: JSON-LD playground, Stardog Studio, Protégé, Gephi, and GraphDB. Each of these tools was installed on our local machines and tested to see how various data sets can be explored and visualized. To find the advantages and limitations of these tools, we compared them according to the following criteria: *Data Exploration through Queries, Configuration, Query Results and Output Storage, Performance and Scalability, Vastness, Relationship Definition and Classification, Tutorials,* and *Documentation.* We expected that we would face some limitations in each of the studied programs, but still, it could be difficult to make some general recommendations, since the relevance of the use of a concrete program depends on the specific preferences of the users.

Needless to say, 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 are 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, query it, and 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, and integrate data in real-life applications and 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 rely more on the edges of the graph. In this project, we focus on the query language SPARQL, which can efficiently 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. Many of them use a visual approach that allows the user to create direct visualization of SPARQL endpoints, which help to formulate and explain the nature and structure of phenomena. Visual models, such as

[1] https://github.com/Visualisation-of-SPARQL-Endpoints

graphs, have a special cognitive power presenting the resources of cognitive graphics to structure information. The diversity and complexity of information the endpoints contain challenge the development of such systems. The developers search for different solutions on how to make their programs the best on the market – the programs that support querying, exploration of raw, unique data from many endpoints, and have case-specific visualization features.

The research of the requirements in data exploration and visualization and the limitations of existing tools can help the developer make some improvements and the user chooses an appropriate tool. Modern systems face numerous challenges, some of them can process only small data sets, and others cannot offer all useful features to handle a variety of tasks in the contemporary world (Bikakis et al., 2016).

## 2 Related work

A lot of knowledge is stored in Resource Description Format (RDF). RDF format is a representation of data in the form of triples: subject-predicate-object, describing a directed relationship from the subject to the object. These relationships are defined in ontologies. The Uniform Resource Identifier (URI) is used to identify subjects, objects, and predicates (Bergman, 2009). Many institutions use such a format as it allows the data to be linkable and exchangeable (Chawuthai et al., 2016). However, the data has a minimum value if it is impossible to find relevant information in it. For this reason, the interest in 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, and classes (Menin et al., 2021).

The advantage of RDF is that it can be serialized in many ways: RDF/XML, N3, RDFa, Turtle, N-triples (Bergman, 2009), JSON, JSON-LD formats, etc. It is difficult to answer which serialization is the best one because it depends on the preferences of the user. For example, RDF/XML shows the graph as an XML document and is convenient for the user who needs to work with XML (Gandon et al., 2014); JSON serialization allows to store data set more compactly and is supported by most browsers as opposed to RDF/XML; JSON-LD is used for serialization of Linked Data in JSON and disambiguates the keys of JSON documents with the help of context, in other words, this serialization is more convenient since it can be associated with human-readable text and supports both RDF and JSON (Purohit et al., 2014; Kellogg et al., 2020).

The exploration of the datasets with unknown structures and without appropriate knowledge about the Semantic Web can be rather struggling (Chawuthai et al., 2016). Many researches were conducted 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 usable 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). 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). It is inevitable to write the correct construction of queries and have an understanding of the construction of dataset logic. In the data retrieval process, there could be instances of data federation that only expand the complexity of the task and may lead to a reduction in search quality (Menin et al., 2021). Therefore, it is required to develop user-friendly and high-quality applications which can visualize SPARQL endpoints and help to retrieve useful information.

To make the exploration in SPARQL simpler and more understandable for all users, some developers create web-based viewers. For example, SPARQL playground [2] 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 set 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).

Numerous applications that can be used for ontology engineering and data exploration have been developed in the last years. But not all of them can stay on the market for a long period. There are several reasons for this: 1) lack of user-friendly

[2] https://sparql-playground.sib.swiss

functionality; 2) lack of feedback and communication between the user and the developer or weak support of the developer; 3) the applications cannot be longer maintained or are very complicated for ontology engineering (Bonduel et al., 2018). The researchers try to analyze what requirements for endpoint visualization each application should fulfill. Many different visualization techniques in different domains as well as the purposes of their users are studied to create some visualization and design guidelines for developers (Tufte, 1990; Dadzie et al., 2011).

## 3 Visualization of SPARQL endpoints

In this section, we would give a brief description of the criteria we have used to analyze the various tools we looked at. For an ideal graph visualization tool, there should be the ability to ingest diverse data, provide flexibility with regards to schema changes and mappings, efficiently handle powerful queries and at the same time, serve unforeseen information needs. Such tools contribute to advanced data management solutions. We, therefore, summarize the features an ideal graph visualization tool should possess as follows:

*Data Exploration through Queries* - It is central for graph visualization tools to possess an interface for querying and exploration of data.

*Configuration* – It should be possible to load and visualize query outputs without complicated configuration protocols.

*Query Results and Output Storage* – Users may be more akin to tools that can export files for local storage.

*Performance and Scalability* – Graph visualization tools should constitute scalable solutions that can handle very large sets of triples. Also, it should be able to handle a huge amount of queries per time, without crashing (for those hosted on a web interface).

*Vastness* – Handling commonly used file formats should be proprietary in the design of graph visualization tools.

*Relationship Definition and Classification* – Relationships between entities should be identifiable e.g., between subjects and objects. This would enable organizations to understand and scrutinize their content and data at a much finer grain of detail.

*Tutorials and Documentation* – There should be user guides and documentation that ensure a smooth set-up and usage of these technologies.

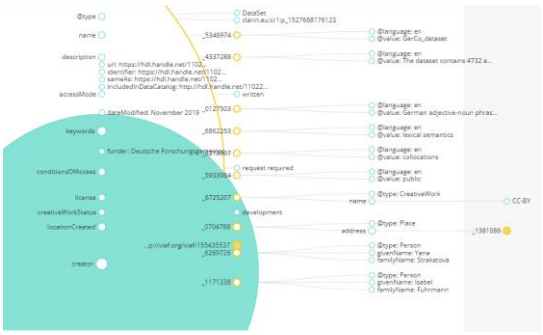


Figure 02: JSON-LD playground visualization of CMDI_05.jsonld

### 3.1 JSON-LD playground

The JSON-LD playground[3] is a web-based JSON-LD viewer. The JSON-LD playground is a publicly available online API. The user can 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 conventional. The difference between expanded and compacted forms lies only in "@context"[4] (a set of rules for interpreting the document). Also, this viewer supports graph visualization of endpoints.

To show this feature at 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

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

[3] https://json-ld.org

[4] https://www.w3.org/TR/2014/REC-json-ld-api-20140116/#dfn-context

window. Figure 02 visualizes CMDI_05.jsonld data. The hierarchical structure of 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.

### 3.2 Stardog Studio

Stardog Studio is a tool that helps to manage data fabric and to create its 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). 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 the CMDI_05.jsonld file. The problem we faced is that Stardog does not support JSON-LD serialization. The program requires an RDF serialization in Turtle, RDF-XML, or RDF-Triple notation. The conversion of ".jsonld" format is possible with the help of some online converters, for example, EASYRDF[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).

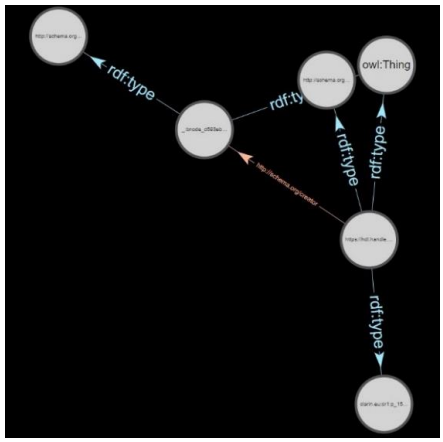
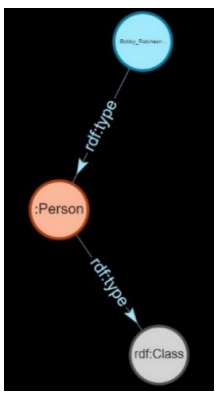



Figure 03: CMDI.jsonld in Stardog Studio

To visualize the results, it is required to select some IRI in the output after running the query. Stardog creates the schema[6], the elements of which are classes and relations. Datatype properties, some information about the instances of classes can be shown as well (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



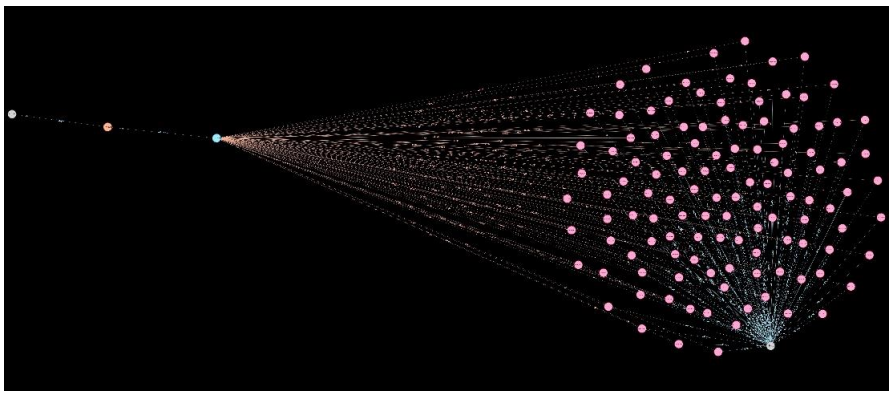



Figure 04 (left): Visualization of Bobby_Robinson data from Music_Data.ttl in Stardog Studio. Figure 05 (right): Visualization of :David_Bowie data from Music_Data.ttl in Stardog Studio.

[5] https://www.easyrdf.org/converter
[6] https://github.com/.../Startdog_schema_for_visualization.txt

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 industry. 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 many information items 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.

### 3.3 Protégé

A team of researchers from Stanford University (California, USA) developed the Protégé ontology editor and its web interface (Musen et al., 2015). 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. Building ontologies is not the only function that is attractive for the customers, it is possible to load and explore existing files as well. It provides the user with query services and visualization support of the file.

A lot of open source and commercial Protégé plugins[7] enhance the Protégé application. Only for visualization purposes, there are more than 20 plugins. In this project, we used OWLViz[8] and OntoGraf[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 with the work with JSON-LD format, the user should use an expanded form. To achieve our goal, we converted our CMDI_05.jsonld to CMDI_05_expanded.jsonld with the help of the JSON-LD playground. Documentation, tutorial videos, and help for the installation of plugins and their settings are provided. Nonetheless, we found one of the shortcomings of this program mentioned in the paper of Bonduel et al. – the lack of user-friendly functions concerning feedback.

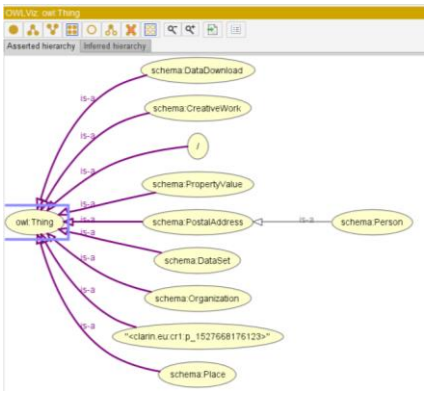



Figure 06: Protégé visualization with OWLViz

Figure 06 shows the visualization of CMDI_05_expanded.jsonld with the help of OWLViz. It provides a view of class hierarchy and navigation in the graph. The colours help to distinguish primitive classes from defined classes and computed changes. OWLViz supports the 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.

The visualization with OntoGraf gives similar results. Here it is possible to navigate and move the relationships of data, filter the relationships and node types to get only the information we need (a very useful feature for large datasets). The user has a variety of layout options: alphabetical, radial, spring, tree-vertical, tree-horizontal, vertical directed, and horizontal directed. Figure 07 illustrates the visualization with a radial layout.

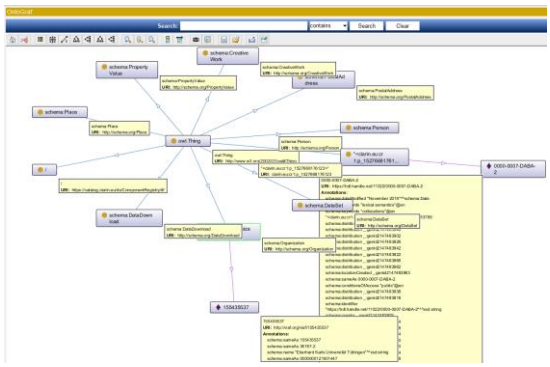


Figure 07: Protégé visualization with OntoGraf

[7] https://protegewiki.stanford.edu/wiki/Visualization
[8] https://protegewiki.stanford.edu/wiki/OWLViz
[9] https://protegewiki.stanford.edu/wiki/OntoGraf

The tooltips show some information about URI and Annotations or some other extensive details about classes and individuals. For a better understanding of the graph, the user can zoom in and out or move the nodes in different places.

### 3.4 Gephi

Gephi is an open-source and free visualization software for graph and network analysis. To be able to visualize, spatialize, filter, and manipulate graphs in Gephi[10], the user is required to import specific modules, plugins, and libraries that are locally available for import upon installation. Some of which include the SemanticWebImport, GraphViz Layout, Oracle Driver, Kleinberg Generator, Network Splitter 3D, etc. The SemanticWebImport plugin[11] enables data to be imported from the computer's local drive or the web using a URL. It also contains the SPARQL query editor through which queries can be used to filter graphs.

The visualization interface uses a 3D engine to display graphs in real-time (Bastian et al., 2009). It can render a network of over 20,000 nodes instantaneously. Gephi possesses rich layout algorithms that can be configured in real-time on the graph window e.g., Label Adjust, which can be run to avoid label overlapping.

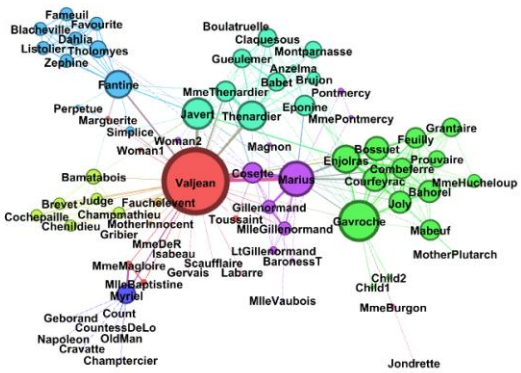



Figure 08: Colour Clustering of the Les Miserables dataset

To explore these features, we analysed the "Les Miserables.gexf" dataset. It constitutes a network of characters in the novel Les Miserables by Victor Hugo. The graphical module's plugin contains parameters such as Size Gradient, Colour Gradient, and Colour Clusters which can be applied to modify the graph display, aiding comprehension of the data structure and content.

Furthermore, visual graphs can be exported as SVG, PDF, or PNG files. They can also be exported as Sigma.js to be visualised on a web interface.

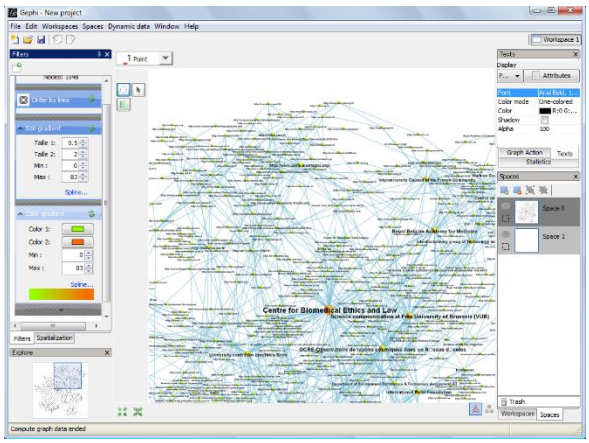


Figure 09: A screenshot of Gephi showing a graph network of multiplex nodes

With the huge configurable layout of algorithms for graph querying and visualization encapsulated in Gephi, we discovered that it has certain limitations, which can be improved upon to increase its robustness. One is the fact that acceptable file formats are limited to only ".gexf", ".gml", ".gv", ".graphml", or ".net". In other words, Gephi does not support RDF formats natively. Besides this, the graph visualization interface also contains multiple plugins and layouts that are not easy to maneuver without following a user guide. Furthermore, the software can be very slow if other applications are running while it is in use. Aside from these limitations, Gephi continues to possess numerous features that stand it out, including the fact that queries can be saved for re-use and graph visualization can be zoomed in and out for better visibility.

### 3.5 GraphDB

GraphDB is a high-performance triple store produced by Ontotext, formerly referred to as OWLIM. Its current version supports RDF 1.1, SPARQL 1.1, and OWL 2. GraphDB is unique in its product-specific tools for visualization, navigation, analysis, and federation. It supplies web-accessible APIs alongside the SPARQL protocol for end-points.

[10] http://gephi.org

[11] https://www.w3.org/2001/sw/wiki/GephiSemanticWebImportPlugin

We explored GraphDB's workbench [12] using CMDI_5 and the Airports database, which contains over 10,000 airports, train stations, and ferry terminals across the globe. The data was converted into RDF triples using GraphDB's 'OntoRefine' feature which helps to convert tabular data into RDF triples.

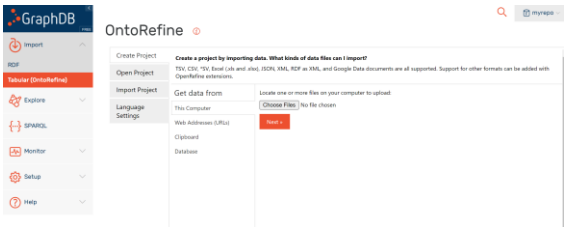



Figure 10: GraphDB's OntoRefine interface

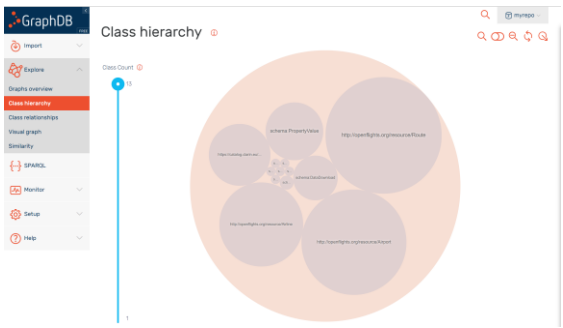



Figure 11: Class hierarchy in the CMDI_5 Database

To use GraphDB, the first step is to create a repository. This is done by clicking the Setup icon then 'Repositories' and then 'Create Repository'. The next is to import the RDF graph. GraphDB's workbench contains a provision to view the data in class hierarchies. It displays components of the data as sub-classes engraved in their super-class. A similar feature in GraphDB's workbench is the 'Class relationship' feature which enables the exploration of class relationships in the data. This can be visualized without having to write queries. Figure 12 shows the top relationships in the CMDI_5 dataset with https://catalog.clarin.eu/ds/ComponentRegistr/#/ (represented in the blue segment of the circle) containing the biggest number of nodes.

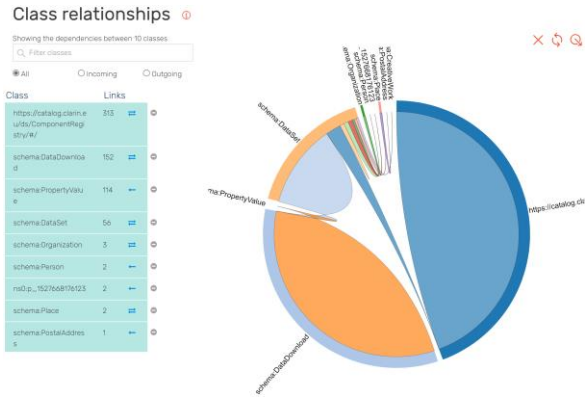



Figure 12: Class relationships in the CMDI_5 Database

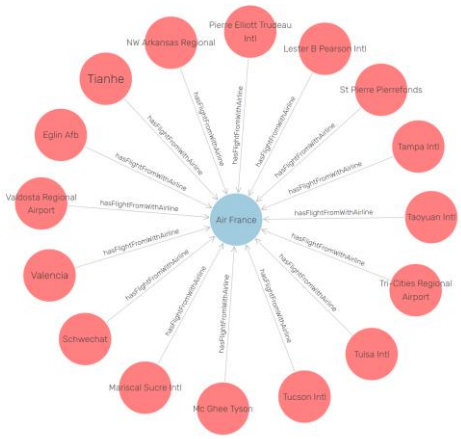



Figure 13: Visual Graph of SPARQL Query

SPARQL queries in GraphDB require a 'CONSTRUCT' statement to be able to display results in a visual graph. The visual graph enables multiple graph views including a provision for query expansion. Figure 13 is the visual graph of a SPARQL query on the Airports database. [13]

## 4 Discussion

Because companies are saddled with the task of managing data that is too diverse, dispersed, and in unthinkable volumes, knowledge graphs have been incorporated into organizational systems and processes causing a paradigm shift in the management of enterprise data.

"*By 2025, graph technologies will be used in 80% of data and analytics innovations, up from 10% in 2021, facilitating rapid decision making across the enterprise.*" (Adrian, M., Jaffri, A., Feinberg, D., 2021)

[12] General — GraphDB Free 9.10.0 documentation (ontotext.com)
[13] Query is available in the following Github repository

No doubt, the use of triplestores has grown rapidly in recent times. These databases also known as RDF, OWL, or Graph databases are now commonly used to manage structured and unstructured data in different industries. A question was posed in an Information Management article (Aasman, J. 2011), "Will triplestores replace relational databases in three or five years?". The author gave two responses. The first response was "yes" because triplestores provide 100 times more flexibility compared to relational databases, and make it possible to easily add new information. The second response was "no" because they would be used alongside relational databases creating an opportunity for smart integration of data. The discussion summary is that triplestore databases are becoming an essential part of database architecture, and the need to create tools and engines that properly handle data queries and cater to other information management needs have become paramount. One of such needs as we have seen is graph visualization.

We have explored the options of different tools and how they allow for the possibility of graph visualization of SPARQL endpoints. In the table below, we attempt to summarize their performance

Table 1. Overview of graph visualization tools.

| | **JSON-LD playground** | **Stardog Studio** | **Protégé** | **Gephi** | **GraphDB** |
|---|---|---|---|---|---|
| *Data Exploration through Queries* | not available | available | available | available | available |
| *Configuration* | easy | complex | complex | complex | easy |
| *Query Results and Output Storage* | not available | available | available | available | available |
| *Performance* (e.g., for large sets of triples*) and Scalability* | not available for processing big amount of data. Can work only with one data file | good performance with large data set | good performance with large data set | slow performance with large data sets in instances where a lot of other applications are running on the local machine at the same time. | good performance with the large data set |
| *Vastness* | supported file formats are limited to only JSON-LD | supports loading data from compressed files directly. Supports: N-Triples, RDF/XML, Turtle, TriG, N-Quads, JSON-LD | supported file formats are limited to only: Protégé Files, Protégé Database Format, XML (more information); or works with many file extensions | supported file formats are limited to only GEXF, GDF, GML, GraphML, Pajek NET, GraphViz DOT, CSV, UCINET DL, Tulip TPL, Netdraw VNA, Spreadsheet | supports the conversion of tabular data into RDF triples. The supported formats are TSV, CSV, Excel |
| *Relationship Definition and Classification* | clearly identifiable | clearly identifiable | clearly identifiable | clearly identifiable | clearly identifiable |
| *Tutorials and Documentation* | available | available | available | available | available |

with respect to the criteria mentioned earlier in Section 3.

We found out that each of the studied programs has a tutorial and documentation support that facilitates working with the tools. The data exploration through queries is available in all tools except JSON-LD playground. It is possible to load the data set into JSON-LD playground and GraphDB without reading complicated configuration protocols, in contrast to Stardog Studio, Protégé, and Gephi. Our other observation was that the performance of the tools differs while processing a large amount of data. For example, JSON-LD playground cannot work with a large data set and Gephi can be very slow if several applications are running simultaneously.

Regarding the vastness of the tools, we noticed that Stardog Studio, Protégé, Gephi, and GraphDB can be more convenient for the user, since these programs support the most commonly used file formats, or at least it is possible to make the conversion or the extension of the file in order to work with it in the tool. Another advantage of the studied tools is that the explored relationships between entities are identifiable and visible in the graph visualization.

The results of the study showed that it is impossible to recommend only one specific visualization tool for the user since the choice of the user should depend on his/her preferences and the features or limitations of a program.

## 5 Conclusion

Graph visualization tools offer companies, businesses, and institutions the opportunity to make sense of data and easily scrutinize data. In this paper, we have discussed the need for efficient and accessible visualization tools and compared state-of-the-art tools that enable graph visualization of SPARQL endpoints. We have also pointed out the limitations these tools have and proposed features an ideal graph visualization engine should have. The ability to handle queries faster, accurately analyze data, and uncover hidden relationships across data should be a priority in the design of graph visualization tools.

## References


Aasman, J. 2011. Will Triple Stores Replace Relational Databases? Information Management and SourceMedia Inc.

Adrian, M., Jaffri, A., Feinberg, D. 2021. Market Guide for Graph Database Management Solutions. Gartner®

Bastian, M., Heymann, S., Jacomy, M. 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks. 10.13140/2.1.1341.1520.

Bergman, M. 2009. Advantages and Myths of RDF.

Bikakis, N., Sellis, T.K. 2016. Exploration and visualization in the web of big linked data: A survey of the state of the art. *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. 10.13140/RG.2.2.13687.93603.

Chawuthai R., Takeda H. 2016. RDF Graph Visualization by Interpreting Linked Data as Knowledge. 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. Semantic Web. 2. 89-124. 10.3233/SW-2011-0037.

Gandon, F., Schreiber, G., Beckett, D. 2014. RDF/XML Syntax Specification. W3C.

Kellogg, G., Champin, P.-A., Longley, D. 2020. https://www.w3.org/TR/2020/REC-json-ld11-20200716/. W3C.

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. *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. AI matters, 1(4), 4–12.

Purohit, S., Harrison, M. 2014. https://www.w3.org/2013/dwbp/wiki/RDF_AND_JSON-LD_UseCases. W3C.

Tufte, E. R. 1990. Envisioning Information. *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; 1 (3): 201–223.

## Appendix A. Graph visualization tools.

Gephi.
http://gephi.org

GraphDB.
https://www.ontotext.com/products/graphdb/

JSON-LDplayground.
https://json-ld.org

Protégé.
https://protege.stanford.edu

SPARQL Playground.
https://sparql-playground.sib.swiss

Stardog Studio.
https://www.stardog.com/studio/

## Appendix B. Documentation.

Gephi SemanticWebPlugIn.
https://www.w3.org/2001/sw/wiki/GephiSemanticWebImportPlugin

GraphDB Documentation.
https://graphdb.ontotext.com/documentation/free/

JSON-LB playground @context.
https://www.w3.org/TR/2014/REC-json-ld-api-20140116/#dfn-context

Protégé.
https://protegewiki.stanford.edu/wiki/Main_Page

Protégé file extension.
https://file.tips/software/protg

Protégé OntoGraph.
https://protegewiki.stanford.edu/wiki/OntoGraf

Protégé OWLViz.
https://protegewiki.stanford.edu/wiki/OWLViz

Protégé supported files.
https://protegewiki.stanford.edu/wiki/PrF_UG_files_project_types

Protégé visualization.
https://protegewiki.stanford.edu/wiki/Visualization

Stardog Studio Schema for visualization.
https://github.com/.../Startdog_schema_for_visualization.txt