# Visual Exploitation of SPARQL Endpoints[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.

## 1 Introduction

The world is governed by data. The potential of data and its importance grew increasingly in the 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, 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, 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 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 a direct visualization of SPARQL endpoints, which helps 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 the 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). 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.

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

## 2 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). 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).

The exploration of the datasets with unknown structure and without appropriate knowledge about Semantic Web can be rather struggling (Chawuthai et al., 2016). 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). 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, the information can originate from different endpoints that only expand the complexity of the task and may lead to reduction of search quality (Menin et al., 2021). 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] 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).

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). 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; Dadzie et al., 2011).

## 3 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.

### 3.1 JSON-LD playground

The JSON-LD playground[3] is a web-based JSON-LD viewer. The JSON-LD is a 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 expanded JSON-LD file to make the data structure more regular. The difference between expanded and compacted forms lies only in the "@context"[4] syntax (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

[2] https://sparql-playground.sib.swiss
[3] https://json-ld.org
[4] https://www.w3.org/TR/2014/REC-json-ld-api-20140116/#dfn-context

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

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

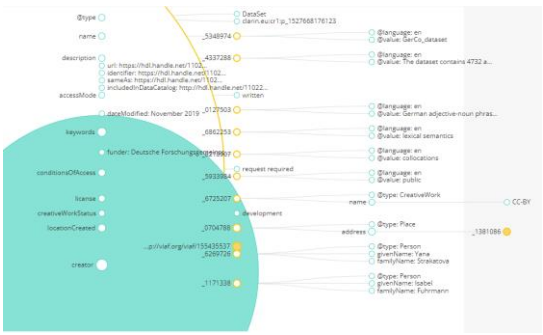


Figure 02: JSON-LD playground visualization of CMDI_05.jsonld

in a separate file.

### 3.2 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). 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. In order to make it work in Stardog, the user should convert ".jsonld" to ".ttl", ".rdf" or ".xml" formats, which is possible with 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).

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 output 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

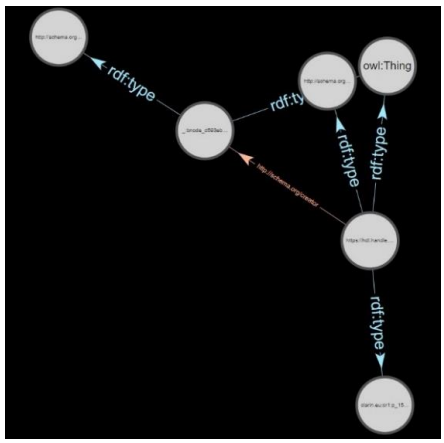
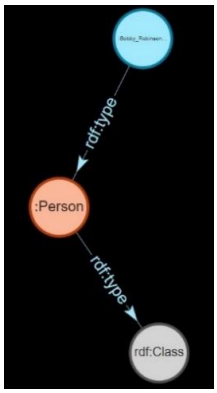



Figure 03: CMDI.jsonld in Stardog Studio

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



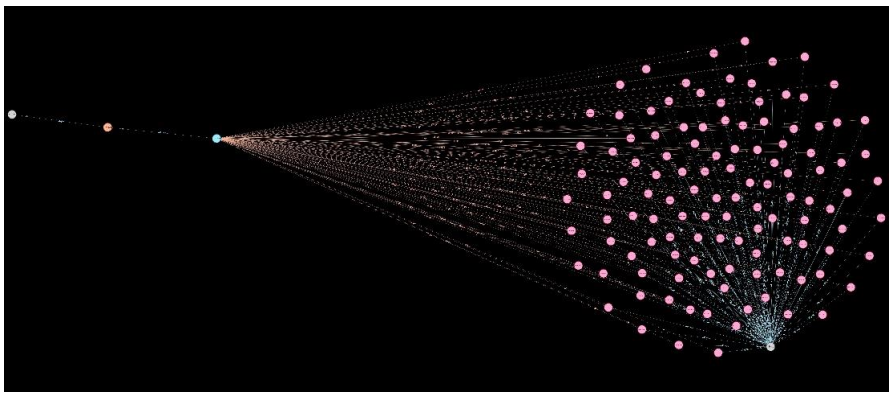



Figure 04 (left): Visualization of Bobby_Robinson data from Music_Data.ttl in Stardog Studio. Figure 05 (right): Visualization of :David_Bowie data from Music_Data.ttl in Stardog Studio.

the data, we get 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 to 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. 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 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 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 the JSON-LD playground. Documentation, tutorial videos, help for installation of plugins, and their settings are provided. We can call this tool user-friendly since 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. – 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 provides a zooming in and out function, but there is no

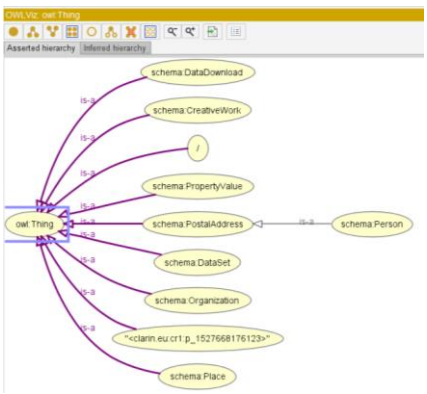



Figure 06: Protégé visualization with OWLViz

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

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, to filter the relationships and node types in order 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, 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.

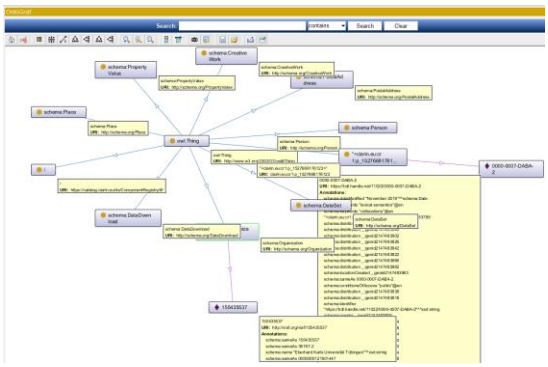


Figure 07: Protégé visualization with OntoGraf

### 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 Gephi10, 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 a SPARQL query editor through which queries can be used to output visual 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.

In order to explore these features, we analysed the "Les Miserables.gexf" dataset provided in Gephi. 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.

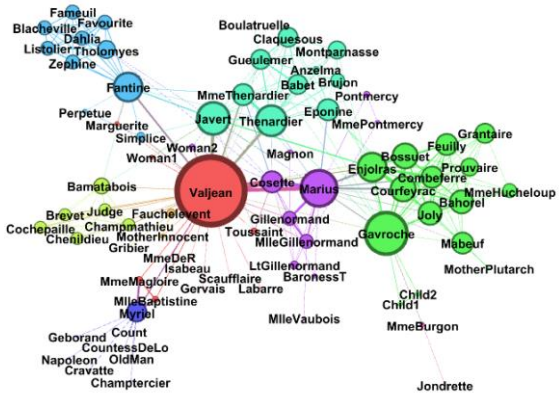



Figure 08: Colour Clustering of the Les Miserables dataset

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.

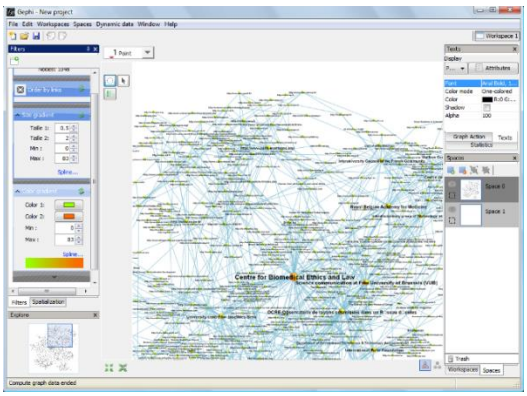


Figure 09: A screenshot of Gephi showing a graph network of multiplex nodes

[10] http://gephi.org

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

With the huge configurable layout of algorithms for graph querying and visualisation encapsulated in Gephi, we discovered that it still 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". Besides, the graph visualization interface also contains multiple plugins and layouts that are not easy to manoeuvre 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. It 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.

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

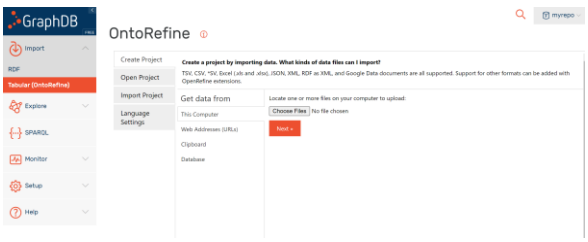



Figure 10: GraphDB's OntoRefine interface

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. If the graph is not originally in RDF format, it can be converted using GraphDB's OntoRefine feature. 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. Another feature in GraphDB's workbench is the

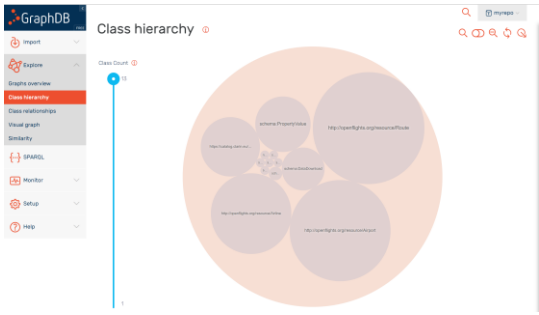



Figure 11: Class hierarchy in the CMDI_5 Database

'Class relationship' feature which enables the exploration of class relationships within the data. This can be visualized without having to write queries. Figure 12 shows the top relationships in the CMDI_5 dataset with the https://catalog.clarin.eu/ds/ComponentRegistr/#/ URI (represented in the blue segment of the circle) containing the biggest number of nodes.

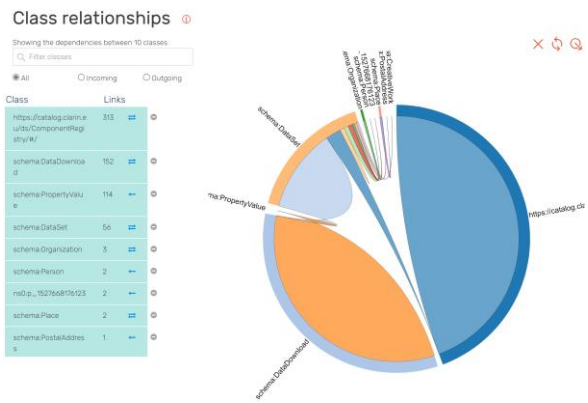



Figure 12: Class relationships in the CMDI_5 Database

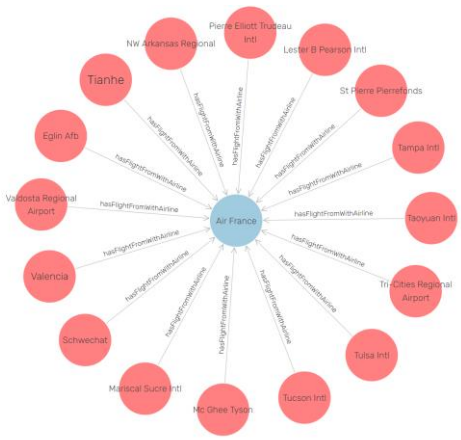



Figure 13: Visual Graph of SPARQL Query

[12] General — GraphDB Free 9.10.0 documentation (ontotext.com)

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.

## 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 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)

No doubt, the use of triplestores have 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 to data. The second response was "no" and that's 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 has become paramount.

An ideal graph visualization tool should have 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 we can say, constitute intelligent data management solutions. We summarize features we think an ideal graph visualization tool should possess based on what we have seen so far:

*Data Exploration through Queries* - It is central for graph visualization tools to possess an interface for query 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.

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

*Relationship Definition and Classification* – Relationship between entities should be clearly identifiable e.g., between subjects and objects. This would enable organizations to understand and scrutinize their content and data with 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.

Graph visualization is very important for pattern detection, anomaly identification and network analysis. It displays how entities are linked together in a way that cannot be expressed by charts. It can also be useful in social network analysis, fraud analysis, territorial defence, supply chain analysis, adverse incident analysis and many more.

## 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. In a nutshell, 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.

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.

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.

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.