GLENDA: Querying RDF Archives with full SPARQL

Olivier Pelgrin[1][0000−0002−1025−9687], Ruben Taelman[2][0000−0001−5118−256X], Luis Galárraga[3][0000−0002−0241−5379], and Katja Hose[1,4][0000−0001−7025−8099]

[1] Aalborg University, Denmark, {olivier,khose}@cs.aau.dk
[2] Ghent University, ruben.taelman@ugent.be
[3] Inria, France, luis.galarraga@inria.fr
[4] TU Wien, Austria, katja.hose@tuwien.ac.at

Abstract. The dynamicity of semantic data has propelled the research on *RDF Archiving*, i.e., the task of storing and making the full history of large RDF datasets accessible. However, existing archiving techniques fail to scale when confronted with very large RDF datasets and support only simple SPARQL queries. In this demonstration, we therefore showcase GLENDA, a system that can run full SPARQL 1.1 compliant queries over large RDF archives. We achieve this through a multi-snapshot change-based storage architecture that we interface using the Comunica query engine. Thanks to this integration we demonstrate that fast SPARQL query processing over multiple versions of a knowledge graph is possible. Moreover, our demonstration provides different statistics about the history of RDF datasets that can be useful for tasks beyond querying and by providing insights about the evolution dynamics of the data.

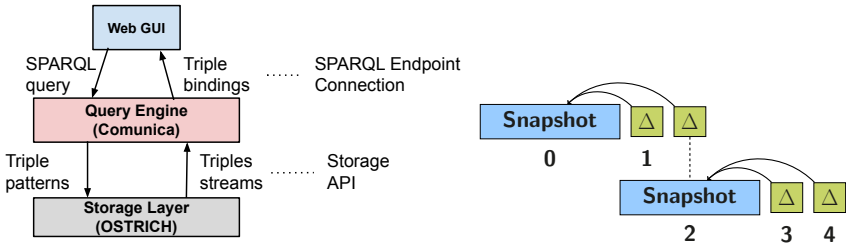
Keywords: RDF archives · SPARQL · RDF · temporal queries · versioned queries · time-travel queries · versioning

1 Introduction

Despite most approaches assuming RDF datasets on the Web to be static and providing optimizations for this case, in reality most RDF datasets are consistently evolving [3,5]. Although there has been some work on archiving, where the focus has been more on storing previous versions, research has not yet been paid much attention to efficiently querying past versions of a knowledge graph without depending on specific system setups [1].

A straightforward way to keep track of the history of RDF data is to store each revision of the dataset as an independent copy. Intuitively, this does not scale well and can become prohibitive for large RDF datasets with long histories. While few efficient solutions for RDF archiving have been proposed [6,9], they support queries on single triple patterns only. This means that executing full SPARQL queries on RDF archives still requires additional post-processing.

In this demo paper, we therefore present GLENDA, a system for executing full SPARQL queries over RDF archives. GLENDA is built on top of a



(a) GLENDA components. (b) Storage layer's architecture.

Fig. 1: GLENDA architecture and components

multi-snapshot change-based storage system for RDF archives [6] that has been integrated with the Comunica [11] SPARQL engine. In the remainder of this paper, we first detail the technical architecture of GLENDA in Section 2. Then, we describe and illustrate GLENDA's main functionalities in Section 3. Finally, we conclude and discuss future work in Section 4.

2 The GLENDA system

Overview. At its core, GLENDA is composed of three distinct and independents components, namely (i) a storage layer composed and an RDF archive store, (ii) a query engine that communicates with the storage layer via an API, and (iii) a user interface in the form of a web application. The query engine is accessible by the client through a SPARQL endpoint.

Figure 1a illustrates the high level architecture of GLENDA. The user interacts with a web-based GUI, where they can write SPARQL 1.1 [8] compliant queries. The query engine is exposed through a SPARQL endpoint with support for versioned queries. The query engine decomposes the full SPARQL query written by the user into versioned triple pattern queries that can be executed natively by the storage layer, which returns answers as triple streams.

Storage layer. We make use of an extension of the OSTRICH [6] system as storage layer. OSTRICH is a scalable engine for RDF archiving that stores the history of an RDF dataset in a single delta chain. A delta chain is comprised of an initial snapshot followed by a sequence of aggregated changesets (Figure 1b). OSTRICH supports versioned queries on single triple patterns with optional offsets. It also provides efficient cardinality estimations for triple patterns. We resort to an extension of OSTRICH, presented in [6], that models revision histories using multiple delta chains. As shown in [6], this improves the ingestion time of new revisions drastically – in particular for very long histories.

Query engine. We chose the *Comunica* [11] query engine to build our SPARQL endpoint. Comunica is a modular, high-performance RDF query engine with full support for the SPARQL 1.1 standard. Building on top of the work from Taelman et al. [10], we opted for a minimal change to the SPARQL language, as

a full extension is outside the scope of this demonstration. The semantic of the GRAPH keywork is changed so that it references versions instead of graphs. We implemented support for three standard types of versioned SPARQL queries [2] described in the following.

- **Version Materialization (VM).** These are queries over a specific version of the RDF Archive. These queries use the notation *GRAPH <version:k>* for $k \in \{0, 1, \ldots\}$.
- **Delta Materialization (DM).** These are SPARQL queries over the changeset between two versions. This is achieved by using the notation for VM queries in combination with the *FILTER (NOT EXISTS)* construct.
- **Version Queries (VQ).** These are SPARQL queries that yield version-annotated query results. They resort to the notation *GRAPH ?version*.

User Interface. We build our GUI as a regular web-page using HTML, CSS, and Javascript. We resort to the Yasgui[5] library for the SPARQL query interface, and the Plotly[6] library for our graphics and visualizations. More details about the user interface and its functionalities can be found next in Section 3.

3 Demonstration of GLENDA

We now demonstrate the capabilities of GLENDA on the BEAR-C dataset [2], which provides 32 snapshots from the Open Data Portal Watch project [4] together with ten full SPARQL queries . To the best of our knowledge, no publicly available system is currently capable of running the queries of this benchmark.

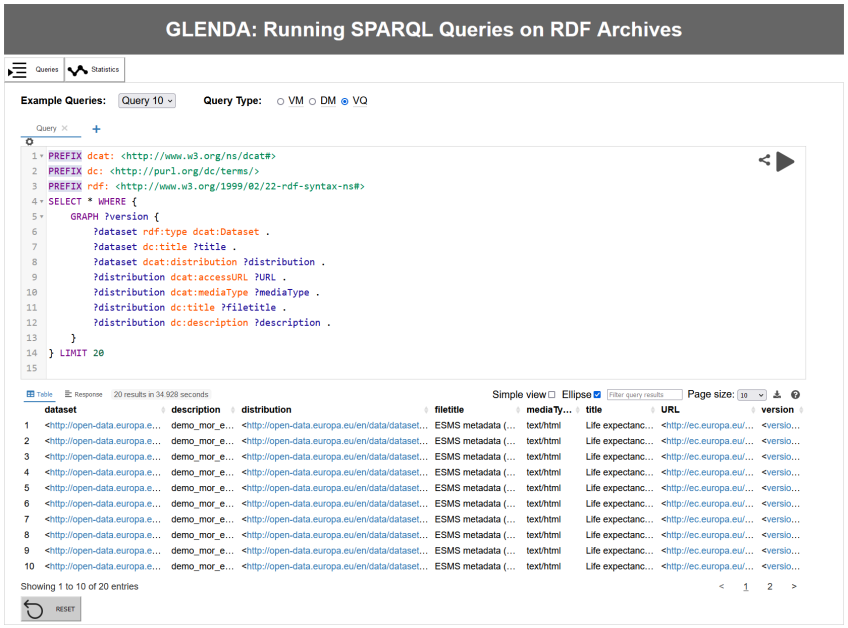
Figure 2a depicts GLENDA's query interface, where the user can write and execute SPARQL 1.1 queries, optionally using our versioning constructs. The queries from the BEAR-C benchmark can be chosen from the dropdown menu on top. The query type can be chosen among VM, DM and VQ queries, and the provided sliders can help the user chose the versions to query.

By selecting the tab "Statistics", the user can have access to various statistics about the underlying dataset (Figure 2b). These are state-of-the-art metrics that describe the dynamics of an RDF archive [5]. Explanations for the metrics are available as tooltips triggered by hovering the mouse over the metric's name. A video showing all the capabilities of GLENDA can be found on YouTube[7]. The system is publicly available at https://glenda.cs.aau.dk and more information can be found on our project webpage[8].

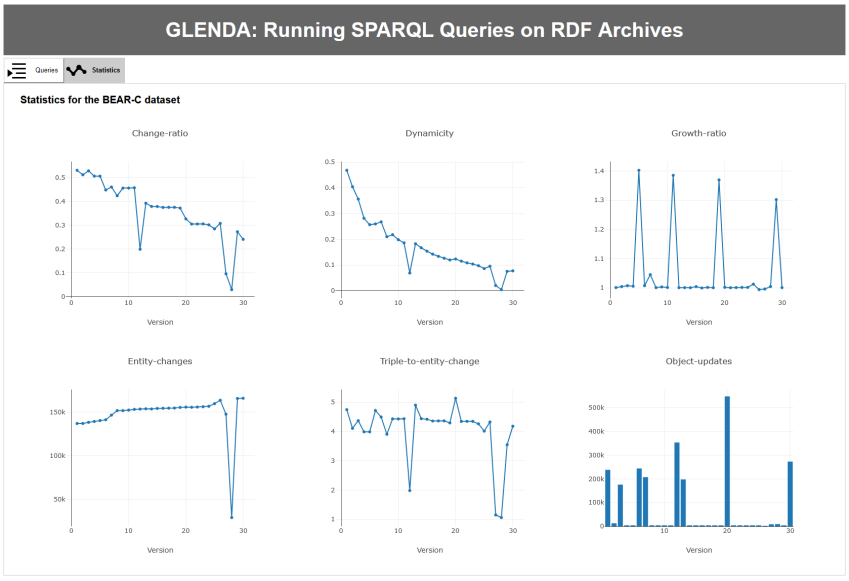
4 Conclusion

We have presented GLENDA, a system to execute full SPARQL queries on RDF archives. We detailed the technical makeup of the system and how its different

[5] https://triply.cc/docs/yasgui-api
[6] https://plotly.com/javascript/
[7] https://youtu.be/DoNjw3V6oSo
[8] https://relweb.cs.aau.dk/glenda/



(a) GLENDA main page and query interface.



(b) GLENDA statistics page.

Fig. 2: GLENDA's user interface

components interact with each other. We explained how queries over archives can be executed with full SPARQL 1.1 via the use of special URIs for named graphs. GLENDA presents itself as a web interface to the user, with user-friendly tools to build and execute queries over RDF archives. We have demonstrated, GLENDA's capabilities on the BEAR-C dataset and queries, which no other system can currently fully support.

In our future work we have planned to consider the development and study of alternative snapshot strategies. Moreover, we envision to reduce the required storage space via more efficient serialization techniques for timestamped deltas. We also expect to improve query processing with advanced RDF representations and novel indexing approaches [7]. Similarly, we envision to study the use of dedicated extensions to the SPARQL language for versioned queries, which would allow for greater flexibility in the querying process, while enabling the simultaneous use of graphs and versions. Finally, we plan to improve the performance of the system further by implementing a more efficient streaming of the results from the storage layer to the query engine.

Acknowledgements. This research was partially funded by the Danish Council for Independent Research (DFF) under grant agreement no. DFF-8048-00051B, the Poul Due Jensen Foundation, and the TAILOR Network (EU Horizon 2020 research and innovation program under GA 952215). Ruben Taelman is a postdoctoral fellow of the Research Foundation – Flanders (FWO) (1274521N).

References

1. Aebeloe, C., Montoya, G., Hose, K.: ColChain: Collaborative Linked Data Networks. In: WWW. pp. 1385–1396. ACM / IW3C2 (2021)
2. Fernández, J.D., Umbrich, J., Polleres, A., Knuth, M.: Evaluating query and storage strategies for RDF archives. JWS **10**(2), 247–291 (2019)
3. Hose, K.: Knowledge Graph (R)Evolution and the Web of Data. In: MEPDaW@ISWC. pp. 1–7 (2021)
4. Neumaier, S., Umbrich, J., Polleres, A.: Automated quality assessment of metadata across open data portals. JDIQ **8**(1), 2:1–2:29 (2016)
5. Pelgrin, O., Galárraga, L., Hose, K.: Towards fully-fledged archiving for RDF datasets. SWJ **12**(6), 903–925 (2021)
6. Pelgrin, O., Taelman, R., Galárraga, L., Hose, K.: Scaling Large RDF Archives To Very Long Histories. In: ICSC (2023)
7. Sagi, T., Lissandrini, M., Pedersen, T.B., Hose, K.: A design space for RDF data representations. VLDB Journal **31**(2), 347–373 (2022)
8. Seaborne, A., Harris, S.: SPARQL 1.1 query language. W3C recommendation, W3C (2013), http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
9. Taelman, R., Sande, M.V., Herwegen, J.V., Mannens, E., Verborgh, R.: Triple Storage for Random-access Versioned Querying of RDF Archives. JWS **54**, 4–28 (2019)
10. Taelman, R., Sande, M.V., Verborgh, R.: Versioned querying with OSTRICH and comunica in MOCHA 2018. In: SemWebEval@ESWC. vol. 927, pp. 17–23 (2018)
11. Taelman, R., Van Herwegen, J., Vander Sande, M., Verborgh, R.: Comunica: a modular sparql query engine for the web. In: ISWC (2018)