Towards Rich Data Queries From Schema Diagrams

Deniss Fjodorovs[1,*], Kārlis Čerāns[1,*]

[1]Institute of Mathematics and Computer Science, University of Latvia, Raiņa bulvāris 29, Rīga, LV-1459, Latvia

Abstract

We present a new approach in which users can turn data schema views into detailed data queries without requiring users to write their own SPARQL queries. This work proposes multicardinal table whose implementation is showcased in ViziQuer, thus providing smooth user experience from schema diagrams into data table examples.

Keywords

Knowledge graphs, Visual schema diagrams, Visual queries

1. Introduction

There are various ways of exploring RDF graphs and schema diagram visualization is one of the options. Schema diagrams provide users a quick and easy-to-understand overview of a knowledge graph by displaying classes, attributes and relationships between classes.

Once a general overview is established, the user most likely will wish to obtain a deeper understanding by fetching specific data. Perhaps there is one class to be queried. Perhaps there are multiple classes that are linked through several attributes.

Data access is possible through SPARQL queries, either by typing raw requests or by using SPARQL editors like YASGUI [1]. Writing SPARQL by hand demands knowledge from the user that many might not possess.

In order to aid the querying process, there are facet-based environments such as SampoUI [2] [3], S-Paths [4], FERASAT [5], LD-Reactor [6], PepeSearch [7] and visual query tools such as ViziQuer [8, 9], OptiqueVQS [10], RDF Explorer [11]. Another distinct approach of querying is by using natural language snippets as seen in Sparklis [12]. Recent advances in LLM technology accelerate natural language querying [13] even further.

Many of the aforementioned tools require preliminary configuration in order to obtain an easy-to-use interface. For example, even though SampoUI [2] provides dashboards that present rich tables, charts and maps, SampoUI currently requires manual JSON configuration that includes writing SPARQL queries and describing dashboard layout.

One noteworthy exception to configurable experiences is S-Paths which only requires defining SPARQL endpoint and a graph. Once that is set, S-Paths automatically fetches potentially useful data paths that are then displayed to user. In order to obtain the best user experience, new solutions should aim to reduce configuration needed before they can be used.

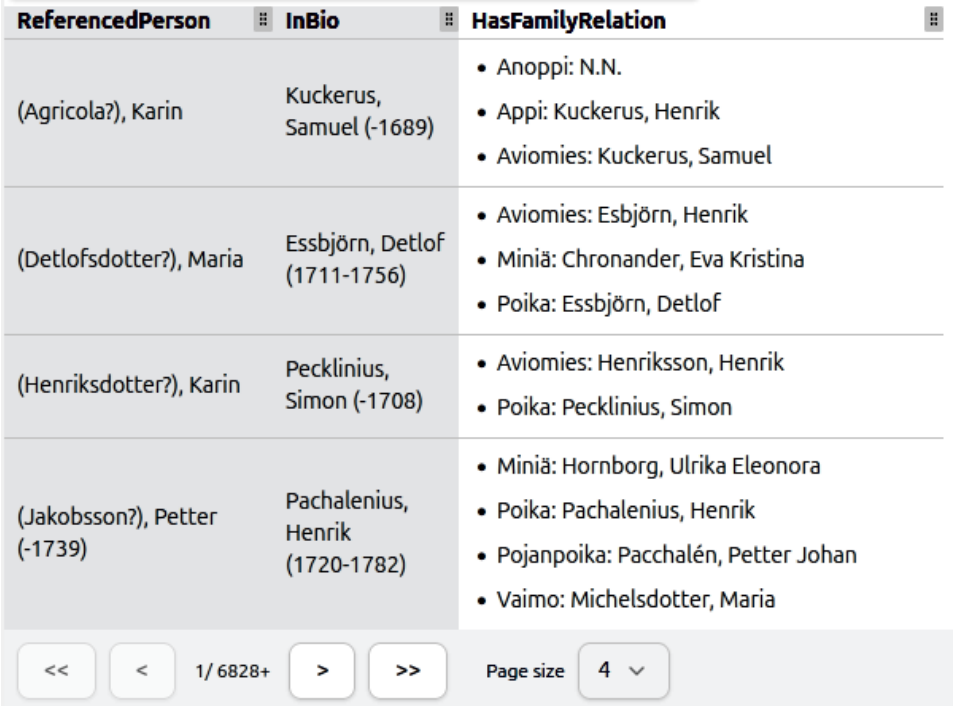
In this paper we present a novel approach in data query creation and result presentation. This includes two components – multicardinal table (Figure 1) and property selector. Property selector provides a simple user interface to select properties to be returned by a SPARQL query (on the basis of the schema of the data that is to be queried). Whereas multicardinal table adds ability to show grouped results given any valid SPARQL query as a reference. Multicardinal table is able to use multiple columns as keys. It avoids the cartesian product problem which generates duplicate information in query results in the case of multi-valued attributes.

The 25th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2026) - Satellite Events (Workshops, Tutorials, Demos, Posters).

*Corresponding author.

deniss.fjodorovs@lumii.lv (D. Fjodorovs); karlis.cerans@lumii.lv (K. Čerāns)

0009-0003-7812-2016 (D. Fjodorovs); 0000-0002-0154-5294 (K. Čerāns)



ReferencedPerson	InBio	HasFamilyRelation
(Agricola?), Karin	Kuckerus, Samuel (-1689)	• Anoppi: N.N. • Appi: Kuckerus, Henrik • Aviomies: Kuckerus, Samuel
(Detlofsdotter?), Maria	Essbjörn, Detlof (1711-1756)	• Aviomies: Esbjörn, Henrik • Miniä: Chronander, Eva Kristina • Poika: Essbjörn, Detlof
(Henriksdotter?), Karin	Pecklinius, Simon (-1708)	• Aviomies: Henriksson, Henrik • Poika: Pecklinius, Simon
(Jakobsson?), Petter (-1739)	Pachalenius, Henrik (1720-1782)	• Miniä: Hornborg, Ulrika Eleonora • Poika: Pachalenius, Henrik • Pojanpoika: Pacchalén, Petter Johan • Vaimo: Michelsdotter, Maria

(a) Multicardinal table

```
PREFIX y: <http://ldf.fi/schema/yoma/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX bioc: <http://ldf.fi/schema/bioc/>
SELECT ?ReferencedPerson ?InBio ?HasFamilyRelation WHERE {
  ?this rdf:type y:ReferencedPerson .
  ?this skos:prefLabel ?ReferencedPerson .
  ?this bioc:has_family_relation ?has_family_relation .
  ?has_family_relation skos:prefLabel ?HasFamilyRelation .
  ?this y:in_bio ?in_bio .
  ?in_bio skos:prefLabel ?InBio .
} ORDER BY ?ReferencedPerson
```

(b) Base query used in focused fragment

Figure 1: Multicardinal table displaying data from Academy Sampo dataset[1]

In order to present the solution within a meaningful context, it will be demonstrated within ViziQuer which provides a foundation for schema diagram navigation and interaction. Nonetheless, the table component is independent and can be integrated within various contexts.

2. Multicardinal table

Multicardinal table shows grouped rows in which value cells can have multiple values. Given a SPARQL query the user can select one or more keys and in return get grouped results, as well as grouped row count information.

One of the key issues that the multicardinal table solves is the cartesian product which results from selecting properties whose cardinality exceeds 1. The issue is demonstrated within Figure 2. A query generates 3x3 cartesian product where the specified award has 3 labels and 3 laureates. The resulting redundancy is highly unfavorable because it makes unique information harder to notice and requires more data to be fetched.

Multicardinal table solves this problem by dynamically rewriting the SPARQL query. Instead of requesting results in a table where every selected SPARQL variable gets its own column, the new query

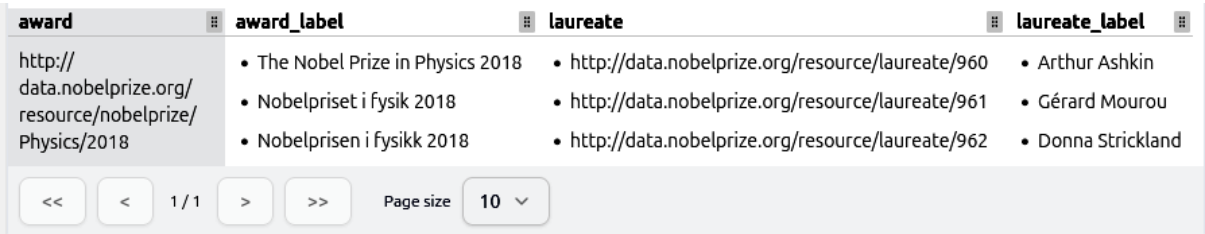
[1]https://ldf.fi/yoma/sparql

```
SELECT DISTINCT * WHERE {
  BIND(<http://data.nobelprize.org/resource/nobelprize/Physics/2018> AS ?award)
  ?award a <http://dbpedia.org/ontology/Award> .
  ?award <http://www.w3.org/2000/01/rdf-schema#label> ?award_label .
  ?award <http://data.nobelprize.org/terms/laureate> ?laureate .
  ?laureate <http://www.w3.org/2000/01/rdf-schema#label> ?laureate_label .
}
```

(a) Query

#	award	award_label	laureate	laureate_label
1	http://data.nobelprize.org/resource/no.../2018	The Nobel Prize in Physics 2018	http://data.nobelprize.org/resource/lau.../960	Arthur Ashkin
2	http://data.nobelprize.org/resource/no.../2018	Nobelpriset i fysik 2018	http://data.nobelprize.org/resource/lau.../960	Arthur Ashkin
3	http://data.nobelprize.org/resource/no.../2018	Nobelprisen i fysikk 2018	http://data.nobelprize.org/resource/lau.../960	Arthur Ashkin
4	http://data.nobelprize.org/resource/no.../2018	The Nobel Prize in Physics 2018	http://data.nobelprize.org/resource/lau.../961	Gérard Mourou
5	http://data.nobelprize.org/resource/no.../2018	Nobelpriset i fysik 2018	http://data.nobelprize.org/resource/lau.../961	Gérard Mourou
6	http://data.nobelprize.org/resource/no.../2018	Nobelprisen i fysikk 2018	http://data.nobelprize.org/resource/lau.../961	Gérard Mourou
7	http://data.nobelprize.org/resource/no.../2018	The Nobel Prize in Physics 2018	http://data.nobelprize.org/resource/lau.../962	Donna Strickland
8	http://data.nobelprize.org/resource/no.../2018	Nobelpriset i fysik 2018	http://data.nobelprize.org/resource/lau.../962	Donna Strickland
9	http://data.nobelprize.org/resource/no.../2018	Nobelprisen i fysikk 2018	http://data.nobelprize.org/resource/lau.../962	Donna Strickland

(b) Raw results



(c) Deduplicated results within multicardinal table

Figure 2: Cartesian product problem comparison against raw and deduplicated results

makes a selection that follows the structure key-property-value. The query result is then reshaped and is presented in the table component.

In addition to data deduplication, multicardinal table provides grouped row pagination. By selecting key columns and page size, multicardinal table fetches only those rows that will be shown. Since each grouped row can have varying amount of cells, special care must be taken during query transformation.

For improved user experience, several additional features are added. Key columns are noticeably different from value columns. Column size can be dynamically adjusted to fit the resulting data better.

Multicardinal table is provided by multicardinal-table[2] repository.

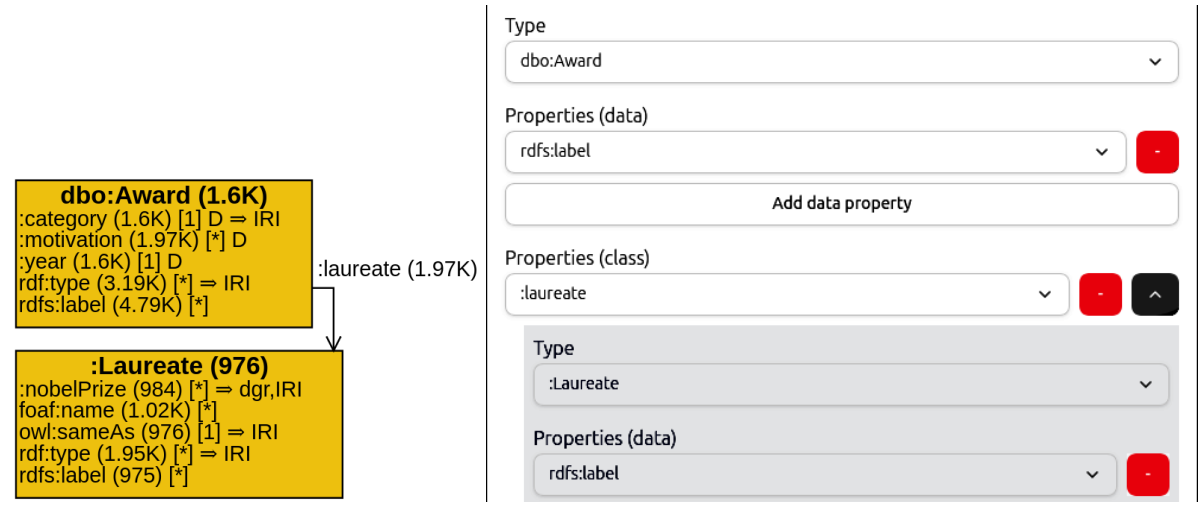
3. From Schema Diagrams to Results

Schema diagrams serve as a general introduction to an RDF graph that is to be examined. One option to obtain workable data schema diagrams in the ViziQuer tool[4] [9].

Once the user has found one or more classes that they wish to receive more information about, they can invoke selection dialog. Within ViziQuer this can be achieved by right-clicking a class node or object property link and selecting ”Custom Data”.

On selection invocation user receives a pre-filled selection with the most popular properties for

[2]multicardinal-table: https://github.com/LUMII-Syslab/multicardinal-table
[3]https://data.nobelprize.org/store/sparql
[4]https://github.com/LUMII-Syslab/viziquer



(a) Focused fragment

(b) Simplified selection

this	this > rdfs:label	this > :laureate	this > :laureate > rdfs:label
http://data.nobelprize.org/resource/laureateaward/Literature/1947/618	• Nobelprisen i litteratur 1947, André Gide • Nobelpriset i litteratur 1947, André Gide • The Nobel Prize in Literature 1947, André Gide	http://data.nobelprize.org/resource/laureate/618	André Paul Guillaume Gide
http://data.nobelprize.org/resource/nobelprize/Literature/1947	• Nobelpriset i litteratur 1947 • Nobelprisen i litteratur 1947 • The Nobel Prize in Literature 1947	http://data.nobelprize.org/resource/laureate/618	André Paul Guillaume Gide
http://data.nobelprize.org/resource/laureateaward/Physics/2018/960	• The Nobel Prize in Physics 2018, Arthur Ashkin • Nobelprisen i fysikk 2018, Arthur Ashkin • Nobelpriset i fysik 2018, Arthur Ashkin	http://data.nobelprize.org/resource/laureate/960	Arthur Ashkin
http://data.nobelprize.org/resource/nobelprize/Physics/2018	• The Nobel Prize in Physics 2018 • Nobelpriset i fysik 2018 • Nobelprisen i fysikk 2018	• http://data.nobelprize.org/resource/laureate/960 • http://data.nobelprize.org/resource/laureate/961 • http://data.nobelprize.org/resource/laureate/962	• Arthur Ashkin • Gérard Mourou • Donna Strickland
http://data.nobelprize.org/resource/laureateaward/Physics/2018/961	• The Nobel Prize in Physics 2018, Gérard Mourou • Nobelprisen i fysikk 2018, Gérard Mourou • Nobelpriset i fysik 2018, Gérard Mourou	http://data.nobelprize.org/resource/laureate/961	Gérard Mourou

(c) Table result fragment

Figure 3: From schema fragment to multicardinal table result fragment in Nobel Prize dataset[3]

the specific diagram element. ViziQuer relies on a data shape server[5] (DSS) that contains extended information about multiple SPARQL endpoints (consider [14] for schema description preparation and storing in DSS). This server is then used to fetch top 5 data and top 2 object properties for the selection.

When selection is initiated through property link, the pre-filling process is done to both connected classes which are then connected through the object property that the clicked property link represents. The top property data is provided by DSS that is integrated within ViziQuer. This default selection serves as a sensible default and a starting point for forming more advanced selections.

The created property selection interface allows introducing new properties into the query selection list, including the properties of the pivot class instances, as well as the instances from the linked classes.

The selection component supports arbitrary nesting depth. This ability lets users chain several classes through object properties, allowing more advanced queries to be built, without sacrificing the selector's

[5]https://github.com/LUMII-Syslab/data-shape-server

ease of use.

In order to aid the discovery and selection of the properties to examine, selector component accepts custom suggestion functions which receive selection context and return a promise of suggestions. While the promise is unresolved a loading spinner is displayed. In ViziQuer, the suggestion functions call DSS.

Once selection is done, its JSON representation is returned which is then converted into a SPARQL query. The JSON representation allows easy manipulation and serialization of the selection. This representation can be used to provide selection templates or history so that the user does not need to make a selection from scratch on every invocation.

Once the query is generated it can be executed as is or it can be passed to multicardinal table component. Since multicardinal table allows wrapping any SPARQL query and the property selector is not directly related to the table component, the concerns are clearly separated.

Example selection flow can be seen in Figure 3. First a desired fragment of the schema is selected. Then the selection is simplified to only contain labeled award and its laureates that are also labeled. As a result, a multicardinal table with four columns is obtained.

4. Conclusions

We have demonstrated the viability of integrating the multicardinal table and property selector within the rich foundation of RDF-related tools within ViziQuer. The clean integration between ViziQuer and implemented functionality provides a seamless experience for navigating from general overview of an RDF graph to obtaining specific data.

Despite the focus on interoperation with ViziQuer, the implemented functionality is independent. Multicardinal table and property selector can easily be integrated in various contexts, and ViziQuer serves as a prime example of integration ability.

Further work on multicardinal table utilities would naturally involve enrichment of the table UI and a more granular row-level selection. A richer table UI would include such elements as hyperlinks, images and charts which can aid users' understanding of data.

Whereas more granular row-level selection is required to handle edge cases in which a single row can contain hundreds or even thousands of values. Not only these edge cases can make UI bloated, the resulting SPARQL queries can quickly hit the query limit. Reaching the limit is unfavorable because the data is incomplete and there is no certainty in which cells the data is missing.

Additionally, query generation and execution must be evaluated in a wide array of situations in order to ensure desirable execution time. Performance characteristics must be evaluated not only on different queries but also on different query engines, since each engine has quirks that could dramatically impact the execution time.

Acknowledgments

This work has been partially supported by a Latvian Science Council Grant lzp-2024/1-0665 "What is in Your Knowledge Graph?".

Declaration on Generative AI

We have not used Generative AI in the preparation of this paper.

References

[1] L. Rietveld, R. Hoekstra, The yasgui family of sparql clients, Semantic Web 8 (2016) 373–383.

[2] H. Rantala, A. Ahola, E. Ikkala, E. Hyvönen, How to create easily a data analytic semantic portal on top of a SPARQL endpoint: introducing the configurable Sampo-UI framework, in: VOILA! 2023

Visualization and Interaction for Ontologies, Linked Data and Knowledge Graphs 2023, volume 3508, CEUR Workshop Proceedinbgs, 2023. URL: https://ceur-ws.org/Vol-3508/paper3.pdf.

[3] N. K. Grislis, K. Čerāns, M. Grasmanis, H. Rantala, E. Hyvönen, How to add a user interface on top of an external SPARQL endpoint: Case Nobel Prize Sampo, in: Semantics 2025: Posters and Demos, CEUR Workshop Proceedings, Vienna, Austria, 2025.

[4] M. Destandau, C. Appert, E. Pietriga, S-paths: Set-based visual exploration of linked data driven by semantic paths, Semantic Web 12 (2020) 99–116.

[5] A. Khalili, P. Van den Besselaar, K. A. de Graaf, Ferasat: A serendipity-fostering faceted browser for linked data, in: European Semantic Web Conference, Springer, 2018, pp. 351–366.

[6] A. Khalili, A. Loizou, F. van Harmelen, Adaptive linked data-driven web components: Building flexible and reusable semantic web interfaces: Building flexible and reusable semantic web interfaces, in: European semantic web conference, Springer, 2016, pp. 677–692.

[7] G. Vega-Gorgojo, M. Giese, S. Heggestoyl, A. Soylu, A. Waaler, PepeSearch: Semantic data for the masses, PLoS ONE 11 (2016) e0151573. doi:10.1371/journal.pone.0151573.

[8] K. Čerāns, A. Šostaks, U. Bojārs, J. Ovčiņņikova, L. Lāce, M. Grasmanis, A. Romāne, A. Sproģis, J. Bārzdiņš, ViziQuer: A web-based tool for visual diagrammatic queries over RDF data, in: ESWC 2018 Satellite Events, volume 11155 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 158–163. doi:10.1007/978-3-319-98192-5_30.

[9] L. Lāce, A. Romāne-Ritmane, M. Grasmanis, A. Sproģis, J. Ovčiņņikova, U. Bojārs, K. Čerāns, Visual presentation and summarization of Linked Data schemas, in: Proc. of KGSWC 2024, volume 15459 of *Lecture Notes in Computer Science*, 2024, pp. 290–305. doi:10.1007/978-3-031-81221-7_20.

[10] A. Soylu, M. Giese, E. Jimenez-Ruiz, G. Vega-Gorgojo, I. Horrocks, Experiencing optiquevqs: a multi-paradigm and ontology-based visual query system for end users, Universal Access in the Information Society 15 (2016) 129–152.

[11] H. Vargas, C. Buil-Aranda, A. Hogan, C. López, Rdf explorer: A visual sparql query builder, in: International Semantic Web Conference, Springer, 2019, pp. 647–663.

[12] S. Ferré, Sparklis: An expressive query builder for sparql endpoints with guidance in natural language, Semantic Web 8 (2016) 405–418.

[13] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying Large Language Models and Knowledge Graphs: A roadmap, arXiv preprint arXiv:2306.08302 (2023). URL: https://doi.org/10.48550/arXiv.2306.08302. doi:10.48550/arXiv.2306.08302.

[14] K. Čerāns, A. Romāne-Ritmane, M. Grasmanis, L. Lāce, OBIS Extractor: Create a schema for your knowledge graph, in: SEMANTiCS 2026 Poster and Demo Proceedings, 2026. Accepted demonstration paper, forthcoming.