Towards UML-style Visual Queries over Wikidata

Kārlis Čerāns[0000−0002−0154−5294], Jūlija Ovčiņņikova[0000−0002−5884−763X], Mikus Grasmanis[0000−0002−0668−0970], and Lelde Lāce[0000−0001−7650−2355]

Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia
{karlis.cerans,julija.ovcinnikova,mikus.grasmanis,
lelde.lace}@lumii.lv

Abstract. We describe and demonstrate the options for visual UML-style presentation and visual-based creation of SPARQL queries over *Wikidata*, a central Linked Data resource employing a custom data classification encoding. We provide visual presentations of public *Wikidata* example queries, this way adding a visual dimension to their comprehensibility. The process of visual query creation within the tool is supported by auto-completion facilities that consider the context of the already created query part, where possible.

Keywords: SPARQL · Wikidata · Visual queries · Query visualization · ViziQuer

1 Introduction

Visual presentation of information artefacts can help their perception. There is a number of tools available for visual creation of SPARQL queries over RDF data endpoints (cf. e.g., [5, 6, 8, 9]). Visual method has been successfully used for SPARQL query formulation by custom domain experts [8]. *ViziQuer* [5] has shown the possibility to use a UML-style notation to visually create [4] and visualize [3] complex SPARQL queries, involving e.g., basic graph patterns, aggregation and subqueries, complex data expressions and filters.

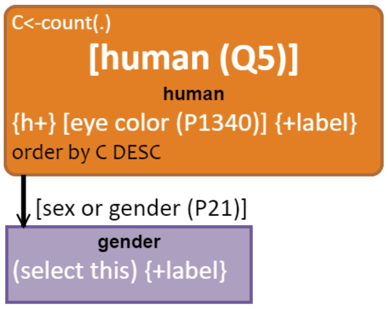
In this paper we demonstrate for the first time the possibility to ***create visual presentations*** for a large set of generic SPARQL queries (we do this for seven sections of *Wikidata* [10] example query set [11]); we provide a gallery of the visually presented queries in a working visual tool environment (each visual query can be translated back into SPARQL and executed over the *Wikidata* data endpoint[1]).

We also demonstrate an ***auto-completion-supported environment*** for visual query creation over *Wikidata* (involving both schema-level and data-level elements); we believe this to be novel regarding creation of UML-style visual queries over large, heterogeneous and custom-encoded data endpoints, as *Wikidata* is.

We discuss the solutions enabling the *Wikidata* SPARQL query visualization and visual query creation over *Wikidata*.

The supporting material for the paper includes links to the live environment and is available at http://viziquer.lumii.lv/examples/wikidata2022.

[1] https://query.wikidata.org/



```
SELECT ?gender (COUNT(?human) AS ?C)
      ?eye_colorLabel ?genderLabel WHERE{
  ?human wdt:P31 wd:Q5.
  ?human wdt:P1340 ?eye_color.
  ?human wdt:P21 ?gender.
  SERVICE wikibase:label {bd:serviceParam
      wikibase:language "[AUTO_LANGUAGE],en" .}
}
GROUP BY ?eye_colorLabel ?gender ?genderLabel
ORDER BY DESC(?C)
```

Fig. 1: Human eye colors by gender

2 Visual Query Notation

A UML-style visual query in *ViziQuer* involves query nodes describing variables or resources; each node can have a possible class name and an attribute specification list. There is a main query node (orange round rectangle) in the query. The edges that connect the nodes usually correspond to links among the node variables or resources (there can be ”same-instance” links, labelled by ’==’, and ”empty” links that do not specify a data connection, labelled by ’++’, as well). Textual condition/filter fields, along with aggregation and query nesting options are available, as well.

We refer the reader to [1] for *ViziQuer* basic constructs and design rationale explanation, [5] for tool description and [4] for its syntax and semantics.

Figure 1 shows a simple query looking for eye color statistics of humans by their gender in both the visual notation and in SPARQL (the prefix definitions are omitted).

For visual queries over *Wikidata* we use a ***presentation by label*** approach for its entities (classes, properties, individuals) to ensure direct readability of the visual query presentation. The entity IRI fragment information is maintained in the entity presentation to ensure that it is unanimous, and to enable easier connecting the visual queries with other approaches for exploring, querying, and analyzing the *Wikidata* data.

The UML-style visual notation allows for a ***single designated classification triple*** to be shown in the prominent UML class position within a query node. Usually, this triple is based on the *rdf:type* property. In *Wikidata*, the most informative classification property is *wdt:P31* (“instance of”), therefore this property is used to link a query node instance corresponding to a query node with the item visualized as UML class name. One also can specify a classification of an instance to be “indirect”, thus making the instance-to-class relation to become *wdt:P31/wdt:P279** (*wdt:P279* is a “subclass of” relation).

The visual notation has also custom means (e.g., {+ *label*}) to support the ***label service extension*** in SPARQL queries over *Wikidata* that instructs finding meaningful entity characterization in query results (this option is used widely in the *Wikidata* SPARQL query examples [11]). The languages of the label service, if not being the default set, can be specified on the level of the entire query, or a subquery.

The visual query notation contains full information about a query (in the context of a given data set schema). It is possible to ***generate the query SPARQL form***, as well as to ***execute the query*** from the visual environment. The ***reverse translation (visualization)*** of SPARQL queries into the visual notation is available, as well (cf. Section 3).

3 SPARQL Query Visualization and Visual Query Library

The principal advantages of a ***visual UML-style query presentation*** of a SPARQL query (presumably used to ***accompany***, not to replace the textual query form) lie within

- presenting the ***classification and attribute selection triples*** in a compact notation (that involves just a class or property name within a query node), and
- splitting the query over ***multiple visual elements*** to reduce the local complexity of any visually separated query part.

An important feature of visual *Wikidata* SPARQL query presentation comprehension is also inclusion of the ***entity labels*** within the query presentation.

The automated visualization of SPARQL queries (cf. [3]) can be possible due to the rich visual query constructs supported by the ViziQuer notation (cf. [1]). It has been used to create a visual query library from the *Wikidata* SPARQL query example set (a manual positioning of the query nodes is done after the automated query visualization). The query visualization module has been expanded from [3] to consider the special query encoding and custom query constructs that are typical for the queries over *Wikidata*.

There are currently 105 queries in the library that have been obtained from 120 considered SPARQL queries from the *Wikidata* example query set [11]. A query is said to be visualized successfully if the obtained visual query can be translated back into a query producing the same results (the notion of mathematical equivalence is adopted if both queries time out or produce empty results). The obtained ratio of 87.5% successfully visualized queries is encouraging[2] as the considered queries are meant to make sense to actual SPARQL query writers and illustrate various aspects of the SPARQL query creation.

The visual *Wikidata* query library is available as a project within *ViziQuer* environment, accessible from the paper's support page. The support page also lists the most recent standings regarding the *Wikidata* example query visualization success rates.

4 Visual Query Environment

The visual query environment provides a visual-centered interactive interface for query creation, starting from the query initialization (query seeding) by a class, a property or

[2] The current successful visualizations by the example set sections are: Simple queries 20 (of 21), Lexeme queries 17 (17), Wikibase predicates 7 (8), Wikimedia projects 13 (13), Entertainment 17 (20), Computer Science and Technology 9 (11), Biology and Medicine 22 (30). We expect to reach 90% threshold in near future.

an individual, followed by query expansion (query growing) that includes adding information to query nodes (class name, instance URI, attributes and aggregations, filters, ordering and slicing), or adding linked nodes to the query.

The principal element for both query seeding and query growing is auto-completion that is expected to offer to the user meaningful options of choosing the elements relevant for a particular context, including a name text search option.

For seeding a query from a class or property name, as well as for choosing a property in a context of a class name, or a class name in a context of a property, a custom relational database of holding the class and property names, as well as their relations is used (a similar solution over *DBPedia* has been described in [2]); the data has been extracted from the public *Wikidata* SPARQL query endpoint by a similar set of queries. Seeding and growing of queries by individual names is currently done via public *Wikidata* entity search API.

5 Discussion and Conclusions

The demonstrated work shows a possibility to adapt rich UML-style visual query solutions to work with *Wikidata* that is a large and heterogeneous Linked data collection and uses custom data encoding conventions. The implementation of the *ViziQuer* tool has demonstrated the necessary flexibility to incorporate the required custom solutions.

The library of query visualizations demonstrates the richness of extended UML-style visual notation in encoding SPARQL queries of varying complexity. The possibility to obtain a visual presentation of a SPARQL query provides a benefit of an extra structural view over it[3]. The extent of this benefit and the scope of the users and queries over which it is substantial is to be determined within a future work.

The choice of UML-style diagrams for SPARQL query visualizations can be rightfully questioned. The current proposal is an expansion of an existing visual query creation and SPARQL query visualization approach (that has clear benefits in the case of smaller and more class-oriented data endpoints) to handle the queries typical for the *Wikidata* environment. Its shortcomings can be overcome by further improvements or alternative SPARQL query visualization and/or visual query creation approaches that are aimed at serving the SPARQL query comprehensibility and exploit the visual query presentation dimension in query creation.

From the visual query creation perspective an important future work would be to refine the query completion services to allow class-specific instance suggestions (e.g., by integrating a FAAS-style component [7]), property-property connections (like in [2] for *DBPedia*) and property-instance connections (what instances can be subjects and what instances can be objects of triples with a certain property). A general observation, however, is that a context-sensitive auto-completion solution is also going to be demanding in computing resources; therefore, a balance of the auto-completion specificity and available resources is to be sought for.

Acknowledgements This work has been partially supported by a Latvian Science Council Grant lzp-2021/1-0389 "Visual Queries in Distributed Knowledge Graphs".

[3] This would apply to any potential user, encountering a SPARQL query.

References

1. Čerāns, K., Bārzdiņš, J., Šostaks, A., Ovčiņņikova, J., Lāce, L., Grasmanis, M., Sproģis, A.: Extended uml class diagram constructs for visual sparql queries in viziquer/web. In: VOILA@ISWC. pp. 87–98. No. 1947 in CEUR Workshop Proceedings (2017), `http://ceur-ws.org/Vol-1947/paper08.pdf`
2. Čerāns, K., Lāce, L., Grasmanis, M., Ovčiņņikova, J.: A uml-style visual query environment over dbpedia. In: Research Conference on Metadata and Semantics Research. pp. 16–27. Springer (2022)
3. Čerāns, K., Ovčiņņikova, J., Lāce, L., Grasmanis, M., Romāne, A.: Visual presentation of sparql queries in viziquer. In: VOILA@ISWC. pp. 29–40. No. 3023 in CEUR Workshop Proceedings (2021), `http://ceur-ws.org/Vol-3023/paper12.pdf`
4. Čerāns, K., Šostaks, A., Bojārs, U., Bārzdiņš, J., Ovčiņņikova, J., Lāce, L., Grasmanis, M., Sproģis, A.: Viziquer: a visual notation for rdf data analysis queries. In: Research Conference on Metadata and Semantics Research. pp. 50–62. No. 846 in CCIS, Springer (2018)
5. Čerāns, K., Šostaks, A., Bojārs, U., Ovčiņņikova, J., Lāce, L., Grasmanis, M., Romāne, A., Sproģis, A., Bārzdiņš, J.: Viziquer: a web-based tool for visual diagrammatic queries over rdf data. In: The Semantic Web: ESWC 2018 Satellite Events. pp. 158–163. No. 11155 in LNCS, Springer (2018)
6. Haag, F., Lohmann, S., Siek, S., Ertl, T.: Queryvowl: Visual composition of sparql queries. In: The Semantic Web: ESWC 2015 Satellite Events. pp. 62–66. No. 9341 in LNCS, Springer (2015)
7. de la Parra, G., Hogan, A.: Fast approximate autocompletion for sparql query builders. In: VOILA@ISWC. pp. 41–55. No. 3023 in CEUR Workshop Proceedings (2021), `http://ceur-ws.org/Vol-3023/paper10.pdf`
8. Soylu, A., Giese, M., Jimenez-Ruiz, E., Vega-Gorgojo, G., Horrocks, I.: Experiencing optiquevqs: a multi-paradigm and ontology-based visual query system for end users. Universal Access in the Information Society **15**(1), 129–152 (2016)
9. Vargas, H., Buil-Aranda, C., Hogan, A., López, C.: Rdf explorer: A visual sparql query builder. In: International Semantic Web Conference. pp. 647–663. Springer (2019)
10. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)
11. Web: Wikidata:sparql query service/queries/examples, `https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries/examples`, accessed on 14.04.2022