Schema-Based Visual Queries over Linked Data Endpoints

Kārlis Čerāns(✉), Lelde Lāce, Aiga Romāne, Jūlija Ovčiņņikova, Mikus Grasmanis, Artūrs Sproģis, and Agris Šostaks

Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia
karlis.cerans@lumii.lv

Abstract. We present the option to use the schema-based visual query tool ViziQuer over realistic Linked Data endpoints. We describe the tool meta-schema structure and the means for the endpoint schema retrieval both from an OWL ontology and from a SPARQL endpoint. We report on a store of the endpoint-specific schemas and the options to support the schema presentation to the end-user both as a class tree within the environment and as external visual diagram.

Keywords: Visual query tool · RDF · SPARQL · Linked data · Data schema

1 Introduction

Visual query composition paradigm (cf. [1–5]) along with facet-based [6, 7] and controlled natural language based [8] approaches is a promising venue enabling end-user involvement in query composition over SPARQL endpoints (cf. [1]).

Since the visual query composition is based on creating query patterns from the data classes and their connecting properties, the *a priori* knowledge of the schema of the data to be queried is helpful to support the query construction. The dynamic on-the-fly data schema extraction from a SPARQL endpoint, as used in the facet-based [7] and language based [8] approaches, could be used also in visual query composition. Although probably appreciated by a casual tool end user and by-passing the size limitations of the schema-based approach, the on-the-fly approach would not allow to provide an overall data structure presentation; it would be expected to have difficulties with context-sensitive query creation guidance and the general performance efficiency, as well.

We focus here on the schema-based approach to the query formulation and answer the question, how to create and present to the end-user the schema describing a given existing Linked Data endpoint. The data schema we are to retrieve includes the class and their connecting property information as well as the data types and the property cardinalities. A principal component of the schema computation is observing the sub-class information among the data classes that enables compact data schema presentation in a tree or graph-like form, important e.g. for reducing the relevant source and target class sets for a given property (cf. [2] for a visual illustration of the problem), or enabling a tree-shaped presentation of the endpoint's class structure within the query tool.

E. Garoufallou et al. (Eds.): MTSR 2019, CCIS 1057, pp. 200–206, 2019.
https://doi.org/10.1007/978-3-030-36599-8_18

We describe here the following novel solutions for Linked data endpoint structure examination and visual query creation over those:

- a pipeline for enabling schema-based visual query creation over a Linked Data endpoint (involving schema generation, storing, adjustment, export as OWL ontologies and visual presentation);
- a store of generated and manually curated schemas for specific Linked data endpoints, ready for loading into the visual query tool *ViziQuer*[1] [9, 10];
- hierarchical class tree component in the *ViziQuer* tool allowing to start query creation from a tree-shaped schema presentation.

We explain also, for the first time, the data meta-schema holding information structure enabling the visual query management in *ViziQuer* and report on a proof of concept experiment with a group of students visually querying a Linked Data endpoint.

2 Visual Tool Data Meta-Schema

Figure 1 provides an overview of the basic data structure used for storing the data schema information in *ViziQuer* [9]. There are entities: classes, properties and datatypes in the data model, identified by their full names (IRIs) and equipped with local names, optionally with prefixes. Every role comes with a list of its applicability contexts ("schema roles"), consisting of source and target classes. For every attribute there is a datatype and a list of source classes. The minimum and maximum cardinalities can be specified both at the property level and at the level of the property within schema; the strongest of the cardinalities must apply. The entities can be annotated.

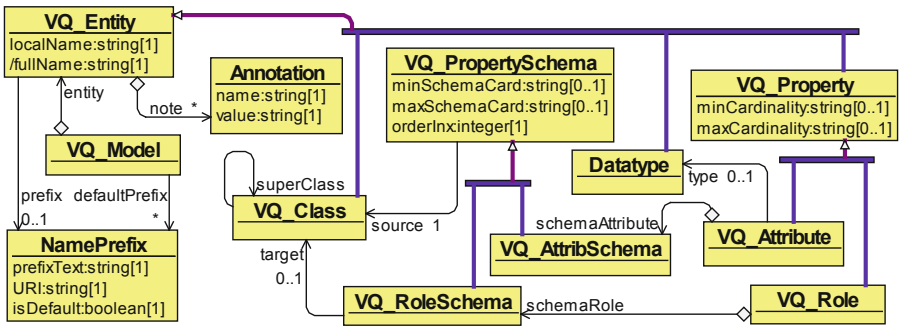

Fig. 1. Visual tool data meta-schema (*ViziQuer*)

We use the knowledge graph with the structure shown in Fig. 1 as the core metadata model in the visual tool and provide its interoperability with RDFS/OWL ontology format via explicit import and export operations. Direct use of RDFS/OWL with

[1] http://viziquer.lumii.lv/.

property domain and range for the property applicability is problematic for properties applicable to multiple classes since a property domain assertion just states that the property cannot be applied outside the class (not that the property can be applied in the class' context).

The mapping of the visual data schema to OWL that we offer can be achieved using two approaches: either the annotation-based one, or mapping the set of property applicability assertions in the data schema to a single OWL property domain assertion with the union of all classes where the property is applicable to (the target classes correspond to the property range class, or to the target class in the all values from restriction). Both the data schema generation from an OWL ontology (ontology "import") and the OWL ontology export from the visual tool data schema can be performed using either of these approaches.

The navigation graphs [11] used recently by *Optique VQs* [12] provide a richer unidirectional way (corresponding to ontology import) of interpreting an OWL ontology as a class and property connectivity graph forming a visual tool schema. The navigation graphs are based on observing any property-based links among the classes found in the ontology (including domain/range, as well as all values from, some values from and cardinality restrictions); it would be useful to extend the OWL ontology import in *ViziQuer* to include the various axiom formats used in navigation graph generation.

Technically, the construction of the *ViziQuer* data schema from an OWL ontology would not include the subclass closure and superclass closure steps of navigation graph construction [11], instead the closures are computed during the tool runtime.

Still, one of the principal data schema acquisition means in *ViziQuer* is by examining the SPARQL endpoint structure itself (cf. Sect. 3), possibly less relevant and therefore not supported within *Optique VQs* approach.

3 Visual Query Enabling Pipeline

The visual queries in *ViziQuer* are created within a context of a project that requires a loaded data schema to enable the query composition (the specification of a SPARQL endpoint is necessary, if the queries are to be directly executed, as well). So, the pipeline of enabling the visual queries starts with obtaining the data schema that is a JSON-structured file, corresponding to the conceptual structure of Fig. 1.

The data schema can be created from an OWL ontology (cf. Sect. 2), or it can be obtained directly from a SPARQL endpoint by a series of schema-level queries.

Several sources, including [2, 13, 14], describe systematic approaches of SPARQL endpoint schema retrieval. Our schema retrieval approach stands out among the others by aiming at full data schema coverage in accordance to Fig. 1 model, including the subclass and cardinality information. The principal (consolidated) steps of the schema extraction involve:

(1) Find all classes and their instance counts and all pairs of intersecting classes;
(2) Based on intersecting classes graph compute the superclass relation;
(3) Find property source and target class information;
(4) Compute minimum and maximum (check, if 1) cardinalities.

The schema extraction process (involving the sub-class and cardinality computation and property assignment to the domain and range classes) can be somewhat lengthy. Having the schema creation process detached from the actual query composition allows to overcome the eventual computational problems of the schema creation.

There is fully viable option of the data schema extraction by the data end user by the schema extractor accompanying the *ViziQuer* tool (the schema extractor can be run on a local computer as JAVA-based web-service).

Furthermore, the created data schemas can be manually tuned, they can be stored and shared; some of the *ViziQuer* schemas for popular SPARQL endpoints are available on its Schema Store[2]. A possible option would be also for a SPARQL endpoint maintainer to create and publish the visual tool schema for the endpoint to foster visual exploration of the endpoint data.

The manual tuning of the automatically generated schemas can be aimed towards improving the visual query creation experience (e.g. by tuning the schema prefix names, or, more radically, by re-grouping and/or excluding classes and their properties that are not considered relevant for the endpoint schema presentation and querying).

The automated data schema generation has been performed for several SPARQL endpoint including, among others, Scholarly data[3], UNESCO[4] (SKOS) and Linked GeoData[5]. Endpoints like *DBPedia* and *wikidata* have not been tried for size considerations.

An important aspect of the visual query enabling is also the presentation of the data schema structure to the end user. The availability of the data schema in the ViziQuer tool format allows exploring it in a tree-like fashion during the query generation (cf. Sect. 4). The option to export the data schema to OWL enables using the OWL ontology support software (e.g. the VOWL ontology visualizer[6] or OWLGrEd[7] ontology editor) to obtain visual presentation of the data schema. The LD_VOWL [11] and LODSight [12] approaches may allow to obtain a visual presentation of the endpoints' data schema, helping the end user orientation, as well.

4 Visual Query Environment

Figure 2 shows a visual query environment fragment in the *ViziQuer* tool with loaded Scholarly data schema. Two important parts are shown: the visual query pane with two queries (each starting at an orange rectangle), and the data schema tree. The generated SPARQL query and its execution results are available within the environment, as well.

The visual queries in Fig. 2 can be described in the natural language, as follows: "*List titles of all papers from the eswc 2017 conference, together with their author*

[2] http://viziquer.lumii.lv/schema-store/index.html.

[3] http://www.scholarlydata.org/sparql.

[4] http://vocabularies.unesco.org/sparql.

[5] http://geo.linkeddata.es/sparql.

[6] http://vowl.visualdataweb.org/.

[7] http://owlgred.lumii.lv/.

count (sort descending)" and "*Find top 10 situations with persons publishing most papers at a single conference*". Both queries rely on the nested query concept (visually introduced using a black circle at an edge end) letting to compute the subquery (possibly including aggregation) in the context of each host query node instance separately. The query notation has been described in detail in [5] and [10].

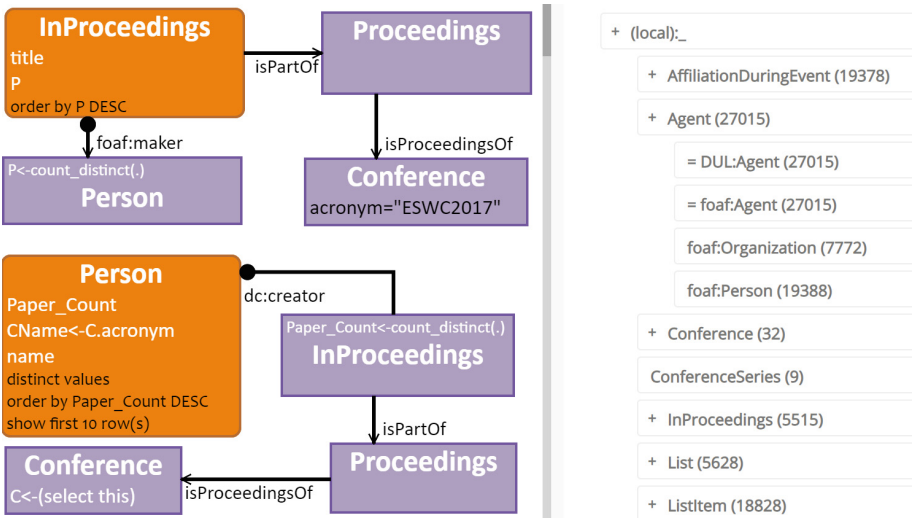

Fig. 2. Visual query environment fragment

The schema tree shows the hierarchy of data schema classes; each class node can be doubly clicked to start a query composition from that class. The current design decision is to make the top-level hierarchy arranged by the namespaces used in the data model; the sub-class structure for each class is shown within its corresponding namespace fragment. The classes and properties from the local namespace are typically shown without the name prefix. The Scholarly schema example can be viewed fully by loading it into the tool from the *ViziQuer* schema store.

A preliminary user study with the query environment has been performed with 17 master's degree students attending the Knowledge Engineering course at University of Latvia, without prior knowledge of the data endpoint structure, with essentially no prior experience with ViziQuer. There has been well over 75% success rate for each of the simple queries (e.g. "*Find all papers published at ESWC2017 conference together with their author count*", or "*Find all conferences together with number of published papers within proceedings of a conference*"). For more complicated queries the result quality varies comparably with result quality variation for query composition success over simple schemas (e.g. the query composition over an in-house hospital data schema, reported in [10]), so a conclusion can be reached that realistic Linked data endpoints may be reachable via the visual query approach, offered by *ViziQuer*.

5 Conclusions

We have shown that schema-based visual query environments, including the *ViziQuer* tool can be used to query SPARQL endpoints, encountered within Linked Data. The full data schema (including sub-class relations) retrieval described here has been successfully applied to a number of data endpoints, still it is work in progress to adjust to handling of different SPARQL endpoints, as they have been made available.

The data schema store at the *ViziQuer* tool home page offers the data schemas ready to be used for visual query composition over the designated Linked Data endpoints. The schema store is expected to be growing both by automatically retrieved and manually curated data schemas. The option to save the data schema as OWL ontology allows to further analyze and visualize it using OWL ontology support software.

The schema retrieval service is open source, as is the *ViziQuer* tool itself, its download and public instance running links are available from the *ViziQuer* tool home page.

References

1. 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. Univ. Access Inf. Soc. **15**(1), 129–152 (2016)
2. Zviedris, M., Barzdins, G.: ViziQuer: a tool to explore and query SPARQL endpoints. In: Antoniou, G., et al. (eds.) ESWC 2011. LNCS, vol. 6644, pp. 441–445. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21064-8_31
3. Kapourani, B., Fotopoulou, E., Papaspyros, D., Zafeiropoulos, A., Mouzakitis, S., Koussouris, S.: Propelling SMEs business intelligence through linked data production and consumption. In: Ciuciu, I., et al. (eds.) OTM 2015. LNCS, vol. 9416, pp. 107–116. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26138-6_14
4. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: visual composition of SPARQL queries. In: Gandon, F., Guéret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9341, pp. 62–66. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25639-9_12
5. Čerāns, K., et al.: Extended UML class diagram constructs for visual SPARQL queries in ViziQuer/web In: Voila!2017, CEUR Workshop Proceedings, vol. 1947, pp. 87–98 (2017)
6. Vega-Gorgojo, G., Giese, M., Heggestoyl, S., Soylu, A., Waaler, A.: PepeSearch: semantic data for the masses. PLoS ONE **11**(3), e0151573 (2016). https://doi.org/10.1371/journal.pone.0151573
7. Khalili, A., Meroño-Peñuela, A.: WYSIWYQ—what you see is what you query. In: Voila! 2017, CEUR, vol. 1947, pp. 123–130 (2017). http://ceur-ws.org/Vol-1947/paper11.pdf
8. Ferré, S.: Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. Semant. Web **8**, 405–418 (2017)
9. Čerāns, K., et al.: ViziQuer: a web-based tool for visual diagrammatic queries over RDF data. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 11155, pp. 158–163. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98192-5_30
10. Čerāns, K., et al.: ViziQuer: a visual notation for RDF data analysis queries. In: Garoufallou, E., Sartori, F., Siatri, R., Zervas, M. (eds.) MTSR 2018. CCIS, vol. 846, pp. 50–62. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14401-2_5

11. Soylu, A., et al.: OptiqueVQS: a visual query system over ontologies for industry. Semant. Web **9**(5), 627–660 (2018)
12. Soylu, A., Kharlamov, E.: Navigating OWL 2 ontologies through graph projection. In: Garoufallou, E., Sartori, F., Siatri, R., Zervas, M. (eds.) MTSR 2018. CCIS, vol. 846, pp. 113–119. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-14401-2_10
13. Weise, M., Lohmann, S., Haag, F.: LD-VOWL: extracting and visualizing schema information for linked data. In: Voila!2016, pp. 120–127, October 2016
14. Dudáš, M., Svátek, V., Mynarz, J.: Dataset summary visualization with LODSight. In: The 12th Extented Semantic Web Conference (ESWC2015). http://lod2-dev.vse.cz/lodsight/lodsight-eswc2015-demopaper.pdf