Visual Query Environment over RDF Data

Kārlis Čerāns[1], Jūlija Ovčiņņikova, Lelde Lāce, Jūlija Hodakovska, Aiga Romāne, Mikus Grasmanis, Elīna Kalniņa, Artūrs Sproģis, Agris Šostaks

Institute of Mathematics and Computer Science, University of Latvia
[1]karlis.cerans@lumii.lv

Abstract. We demonstrate the interactive ViziQuer environment for composing rich visual queries (including aggregation, nesting and data expressions) over SPARQL endpoints. We describe the interactive means for building the visual constructs of the query language and discuss the user study results on the query environment usability. The ViziQuer tool environment is publicly available and open source.

Keywords: Visual queries, query environment, ad-hoc queries, RDF, SPARQL

1 Introduction

Visual query composition paradigm (cf. [1], [2], [3], [4], [5]) along with facet-based ([6], [7]) and controlled natural language based ([8]) approaches offers a promising avenue for involving end-users in query composition over SPARQL endpoints (cf. [1]).

The recent ViziQuer notation ([5], [9]) allows for visual presentation of rich instance level and aggregate queries, involving data expressions, as well as query nesting, with expressive power approaching that of the full SPARQL 1.1 [10]. The corresponding query construction environment reported so far [11] has been rather basic with hierarchic property pane of the visual elements as the primary query building tool.

We shall report in this paper and present in the demonstration:

- Interactive rich visual query building environment with class tree basis and context-based query construction options;
- Context-dependent code completion of attribute and condition expressions and property paths;
- Ready-to-use data schemas over a number of existing public Linked data endpoints (the query examples in this paper are over Scholarly data endpoint [12]).

The user studies identifying the strengths as well as remaining areas of necessary attention within the visual query creation environment are discussed, as well.

There are interactive multi-modal visual query creation environments available in Optique VQs [1] and LinDA [3] visual query systems. The ViziQuer system considered here is meant for much wider visual query range (including aggregation and subqueries, as well as textual language for conditions and attribute expressions).

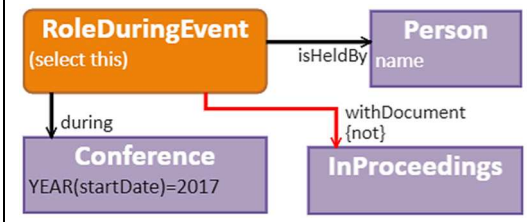
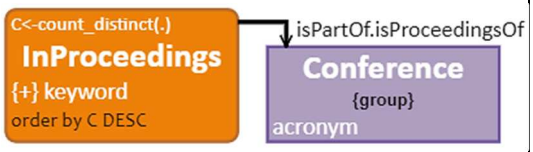
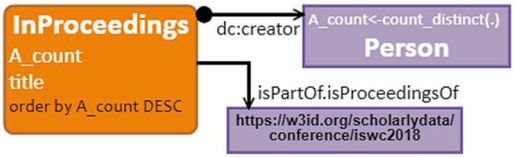
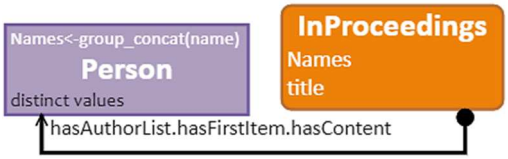
The open source ViziQuer query environment and its supporting resources are available from the home page http://viziquer.lumii.lv/, including the tool local set up options.

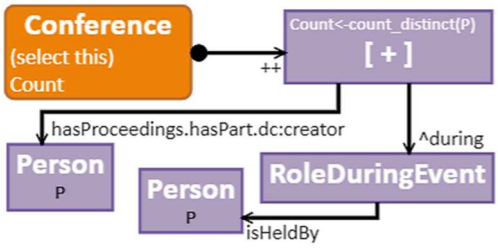
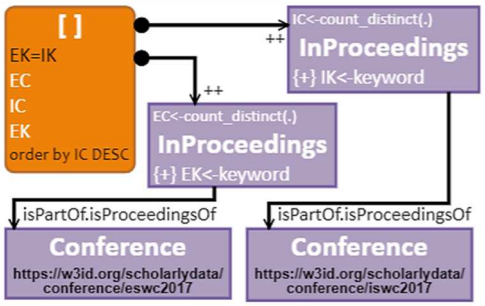
2 Notation Examples

The visual queries are formulated in the context of a data dictionary that is a part of the data schema used by the ViziQuer tool (cf. Section 3), the data dictionary maps the entity short names used in the visual query presentations to their full IRIs.

Each query is a connected graph with one main query node (orange round rectangle). The query examples in Table 1, formulated over Scholarly Data schema [12], show a class-attribute-link-condition option (1), queries with aggregation and grouping (2 and 3), queries with nested aggregation (4 and 5) and queries with advanced structure (6 and 7). The ViziQuer notation is explained in detail in [5] and [9] and the ViziQuer tool home page. The new notations here, if compared to [9], involve explicit grouping (3), instance IRI references (4 and 7) and *distinct values* concrete syntax (5).

Table 1. Visual Query examples

1. *List all non-paper-related roles during a conference starting in 2017, together with the role holder names.* Each query node is a data instance pattern, listing its class name and conditions, and the selection items; the links show the instance connections.	
2. *Find the number of conferences and the number of conferences by the year of their start date (two queries).* The grouping is automatic by all non-aggregated selection fields.	
3. *For every keyword find its usage count (sort descending) in papers within each conference.* The keyword presence is mandatory, denoted by {+}. The *Conference* instance is explicitly added to the grouping set.	
4. *List titles of all papers from the iswc 2018 conference, together with their author count (sort descending).* Nested query construct (*dc:creator* link) is used for calculating author count for each paper separately.	
5. *List titles of all papers together with the papers' first author name(s).* The data may contain duplicate relation information. So, the distinct person name values are selected for each paper, then concatenated to include all name forms in the output.	

6. *For every conference find the number of persons that are either authors of its proceeding papers or have some role during the conference.* The visual links from the union node [+] describe the data links from the *Conference* instances.	
7. *Find the keywords used in proceeding papers of both iswc and eswc conferences in 2017, with respective usage counts, order descending by the usage count in iswc 2017.* The unit node [] is a wrapper around the two subquery results (there can be combining, filtering, or even further aggregation within wrapper nodes).	

3 Query Environment

The query environment serves the creation of queries (such as the ones listed in Table 1) over a knowledge graph schema describing the structure of the data to be analyzed.

The query environment setup requires loading the data schema, listing the available entity names and connections. The data schema can be extracted either from an OWL ontology or directly from the SPARQL endpoint. There is a list of pre-built schemas, as well as the reference to the schema extraction service available at the ViziQuer Schema Store[1]. The Scholarly data schema used in the Section 2 examples has been retrieved into the Schema Store from the Scholarly Data SPARQL endpoint[2]. There is also an example project with the Scholarly data queries available in the Schema Store.

The reference to a SPARQL endpoint is to be specified within a project (in the project parameter window), if the queries are to be directly executed over the endpoint.

The query creation process typically starts by double clicking a node in the class tree (Fig. 1, left) thus adding a query node (orange round rectangle) with this class into the query pane. Figure 1, center, shows the context menu available for nodes within the visual diagram. The attributes, ascribed to the class (or to its superclass, or a subclass) in the data schema, can be checked to be added to the query output via the *Add Attributes* context menu item (the attribute choice dialogue example is in Figure 1, right; the *(select this)* option adds the class instance URI itself; the [+] button expands the choice list to offer object property selection options, as well).

At this point the query SPARQL form can be generated or the query can be executed over the provided SPARQL endpoint using the respective context menu commands.

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

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

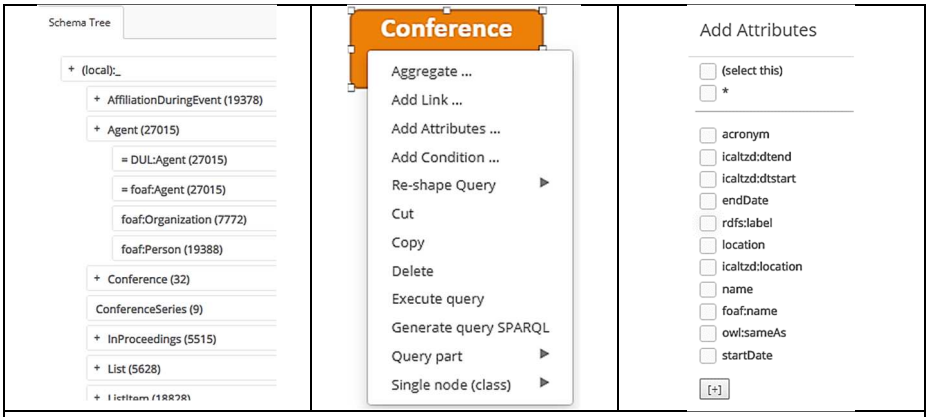


Figure 1. Query environment elements: clickable class tree (left), context menu (center) and attribute addition dialogue example (right).

There are also separate context menu items for aggregation (e.g. count) option and condition introduction into the query nodes. The full editing facilities of the visual query element properties are available in the designated property pane that can be used for structural element introduction, fine-tuning, editing and deleting alike.

There are two options of linked node introduction into a query: either by introducing both nodes (typically, classes) into the query pane and then connecting the nodes by a link from the diagram symbol palette, or the *Add Link* option from the class context menu that suggests adding the links with schema-defined properties for the class, together with the property target classes. Both link creation options distinguish the class join links from the nested query links (links with black bullets, as e.g. in Table 1 examples 4, 5, 6 and 7). The *Add Link* dialogue further on offers a wizard for fast introducing of aggregation into the nested query, together with its result handling within the host query, as e.g. in query 4 in Table 1.

The textual expression information entry within attribute expression and condition fields and in line property paths is supported by a code completion option that is aware of the expression focus placement with respect to the data schema and the field position within the query. So, entering the property paths e.g. in queries 3, 5 and 6 in Table 1, each property name in the path is gradually offered to the end user after the entry of the previous property name, followed by the dot that indicates the continued navigation.

The query re-shape options involve a service for changing the query main class, and a more fundamental *Add Outer Query* option that adds a non-data node '[ ]' introducing the outer query level around the current base query, as e.g. in query 7 in Table 1.

4 Discussion and Conclusions

An earlier user study with Computer Science master's degree students has shown [9] that for persons literate in computing the visual query composition is easier than corresponding query writing directly in SPARQL. Still, the study indicated not very high productivity in visual query creation (on average just 5.27 successful queries within 70

minutes [9]). To test the query composition environment improvements from [9] and [11], we repeated the test (the visual notation part) with 28 similarly prepared participants (the students of the same study course the next year), and the average correctly completed query score has risen to 7.5 under similar conditions. Meanwhile, the most difficult query task with aggregating over attribute under the condition of existential nested query that has reached only 25% (2 of 8) correct score among those attempted the query in [9], did receive the rather unconvincing 45% (9 of 20) here. The analysis of the study results has allowed to identify issues in query composition, resolved in the current tool release, leading to a hope that the ViziQuer tool can ease the query composition task over RDF data stores at least for technically literate persons.

The ViziQuer tool is open source. To ease the local ViziQuer server usage, a prebuilt Docker environment with the tool is offered. The further usability improvements including the integrated and more efficient data schema retrieval is work in progress.

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. Universal Access in the Information Society, March 2016, Volume 15, Issue 1, pp 129–152.
2. Zviedris, M., Barzdins, G.: ViziQuer: A Tool to Explore and Query SPARQL Endpoints. In: The Semantic Web: Research and Applications, LNCS, Volume 6644, pp. 441-445, (2011)
3. B.Kapourani, E. Fotopoulou, D. Papaspyros, A. Zafeiropoulos, S. Mouzakitis, S.Koussouris., Propelling SMEs Business Intelligence Through Linked Data Production and Consumption, In OTM OTM 2015 Workshops pp 107-116.
4. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: Visual Composition of SPARQL Queries. In: The Semantic Web: ESWC 2015 Satellite Events. LNCS, Vol.9341, pp. 62-66. Springer, (2015), http://vowl.visualdataweb.org/queryvowl/
5. K.Cerans, J.Barzdins, A.Sostaks, J.Ovcinnikova, L.Lace, M.Grasmanis and A.Sprogis. Extended UML Class Diagram Constructs for Visual SPARQL Queries in ViziQuer/web In Voila!2017, CEUR Workshop Proceedings, Vol.1947, (2017) pp.87-98.
6. G. Vega-Gorgojo, M. Giese, S. Heggestoyl, A. Soylu, A. Waaler. PepeSearch: Semantic Data for the Masses. In: PLoS ONE 11(3): e0151573. doi: 10.1371/journal.pone.0151573, 2016. http://dx.doi.org/10.1371/journal.pone.0151573
7. A.Khalili, A.Meroño-Peñuela. WYSIWYQ --- What You See Is What You Query. In Voila!2017, CEUR, Vol.1947, (2017) pp.123-130. http://ceur-ws.org/Vol-1947/paper11.pdf
8. S.Ferré: Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language, Semantic Web, 2017, Vol 8, pp 405-418
9. Čerāns, K., Šostaks, A., Bojārs, U., Bārzdiņš, J., Ovčiņņikova, J., Lāce, L., Grasmanis, M. and Sproģis, A., ViziQuer: A Visual Notation for RDF Data Analysis Queries. In *Research Conference on Metadata and Semantics Research.* Springer CCIS, Vol.846, pp.50-62, 2018
10. SPARQL 1.1 Query Language. W3C Recommendation 21 March 2013, http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
11. 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: ESWC 2018 Satellite Events. LNCS, vol 11155, pp. 158-163, 2018.
12. A. L. Gentile and A. G. Nuzzolese. cLODg - Conference Linked Open Data Generator. In *ISWC 2015 Posters & Demonstrations Track,* CEUR-WS.org, Vol. 1486, 2015.