SPARQL Query Builders: Overview and Comparison

Pavel Grafkin[1], Mikhail Mironov[1], Michael Fellmann[2], Birger Lantow[2], Kurt Sandkuhl[2] and Alexander Smirnov[3]

[1] ITMO University, Saint Petersburg, Russia
[2] University of Rostock, Rostock, Germany
[3] SPIIRAS, Saint Petersburg, Russia

Abstract. The SPARQL query language has been proposed as a simple language for querying graph-structured data on the Semantic Web. However, users have to write queries that must conform to the SPARQL syntax. This requirement might be alleviated using a SPARQL query builder that suggests relevant parts of the query. But, up to now, only a few basic comparisons of such tools exist. The goal of this paper is to develop such a comparison as the result of a structured literature analysis.

Key words: Visual query builder, SPARQL, comparison criteria, usability, systematic literature review.

1 Introduction

Semantic technologies are a promising means to reduce the information overflow in organizations since they provide for much more advanced queries and precise results in comparison with traditional approaches. With this, information supply can become "intelligent". Towards this goal, the SPARQL query language has been proposed as a simple language for querying graph-structured data. Originally proposed in the context of the Semantic Web, this language can be used for a large spectrum of use cases. Examples are querying Linked Data on the Web, retrieving information represented with the Simple Knowledge Organisation System (SKOS) from public organizations, querying traditional enterprise-specific relational data viewed as RDF via the RDB to RDF Mapping Language (R2RML) and querying statistical data represented with the W3C Data Cube vocabulary. However, users still have to write queries that must conform to the SPARQL syntax. This is a challenging task especially for novice or casual users. In this regard, visual support and suggestions which are ubiquitous in modern software applications such as mail clients, text editors and operating systems might provide a remedy. Increasingly, such mechanisms are also implemented in SPARQL query builders that may increase the usability of querying. They do so in offering a graphical metaphor for the query and moreover by suggesting relevant parts of the query. Up to now, only a few basic comparisons of such

tools exist. The goal of this paper is to develop such a comparison as the result of a structured literature analysis.

2 Research Method

The research approach used in this paper is a methodologically rigorous review of research results. It aims to be transparent and repeatable as well as to present evidence for all conclusions that were drawn.

2.1 Systematic Literature Review

We conducted a systematic literature review (SLR) according to the guidelines defined by Kitchenham et al [1] in order to provide an overview of what has been published on SPARQL query builders. Kitchenham recommends six steps when conducting a SLR [1]:

1. Formulating the problem at hand - research questions
2. Identification of papers - search process
3. Paper selection - inclusion and exclusion of papers
4. Data collection - data extraction from selected papers
5. Data analysis - presentation of results
6. Interpretation of results

These steps were used in our work and are reflected in the structure of this paper.

2.2 Research Questions

To guide our research activity, we defined the following research questions (RQ) based on the problem statement given in the introduction:

1. Which SPARQL query builders exist and which design goals were followed?
2. What are suitable criteria to compare them?
3. Have these query builders already been used outside the Semantic Web/Linked Data community?
4. Is there an empirical evaluation available?
5. How do the query builders scale when data becomes large or the schema (expressed as an ontology) is expressive (or both)?

3 Paper Selection and data Extraction

3.1 Presearch

The exploration of the topic started with a pre-search: We examined websites of query builders that we knew in advance prior to executing our search. Specifically, we looked for keywords that the authors assigned to their papers to build

our initial query used within the search process (cf. Section 3.3). The list of inspected query builders is given below.

1. Konduit VQB
   http://ceur-ws.org/Vol-565/paper4.pdf
   Keywords: Visual Query Builder, SPARQL, Nepomuk.
2. ViziQuer
   http://viziquer.lumii.lv/
   Keywords: Visual query creation, SPARQL, RDF databases, Semantic technologies.
3. QueryVOWL
   http://vowl.visualdataweb.org/queryvowl/
   Keywords: Visual querying, QueryVOWL, VOWL, SPARQL, RDF, OWL, Visualization, Linked Data,
4. Semantic Web.
   OptiqueVQS http://ceur-ws.org/Vol-1456/paper10.pdf
   Keywords: Visual Query Formulation, Ontology, Usability, SPARQL
5. Visual SPARQL Query Builder.
   No website or articles found. Closest one is http://ceur-ws.org/Vol-658/paper518.pdf.

3.2 Literature Sources

Following sources were selected for systematic literature analysis:

– http://link.springer.com (Discipline: Computer Science)
– http://www.sciencedirect.com

We reviewed papers published from 2006 to 2015.

3.3 Search Process

Based on the results of our pre-search, the following keywords were selected and grouped by similarity:

– Visual query builder, visual query creation, visual querying, visual query formulation, visualization;
– SPARQL;
– RDF, RDF databases;
– Semantic technologies, Linked data, Semantic web;
– Usability;
– Ontology.

The initial search term was obtained by connecting all keywords. Keywords inside groups were joined with *OR* operator and surrounded with brackets:

– (“visual query builder” OR “visual query creation” OR “visual querying” OR “visual query formulation” OR “visualization”) AND “SPARQL” AND (“RDF” OR “RDF databases”) AND (“semantic technologies” OR “linked data” OR “semantic web”) AND “usability” AND “ontology”

This search term gave us 254 results at Springer and 69 results at ScienceDirect. The exploration of search results showed that the keyword *Visualization* is very widely used in papers not related to the research topic. After removing it from the term we got:

– ("visual query builder" OR "visual query creation" OR "visual querying" OR "visual query formulation") AND "SPARQL" AND ("RDF" OR "RDF databases") AND ("semantic technologies" OR "linked data" OR "semantic web") AND "usability" AND "ontology"

By this term, 29 results at Springer and 3 results at ScienceDirect were found. To widen search results we left only necessary keywords related to research topic:

– ("visual query builder" OR "visual query creation" OR "visual querying" OR "visual query formulation") AND "SPARQL" AND "usability"

Finally, we got 37 results from Springer and 7 results from ScienceDirect.

3.4 Paper Selection and Data Collection

The target of the process is to leave only relevant articles in the paper list. A Relevant paper is a paper which helps us to answer at least one question of our research. The decision on paper-relevance was made after reading the article's abstract or looking through the article's content (in case something was not clear in the abstract). Due to practical reasons, we also had to sort out articles that we were unable to access, e.g. due to fees imposed by the publisher that were not covered by our library's subscription. After paper selection, 10 articles remained for further exploration. Subsequently, the data required to answer the research question has been extracted from these articles.

4 Data Analysis

The analysis of data is structured based on the research questions. Therefore, the next section describes the found query builders (RQ1). This is followed by a construction of criteria for comparison (RQ2). Section 4.3 discusses usage outside the Semantic Web community (RQ3), Section 4.4 discusses the evaluation of tools and approaches (RQ4). Finally, Section 4.5 addresses scalability issues (RQ5).

4.1 Query Builders

Here we briefly describe SPARQL query builders mentioned in the selected papers.

QUaTRO2 [2] offers a graphical user interface and domain-expert orientation. The system has positioned itself as domain-independent with the possibility to formulate complex queries despite a high-level visual query language.

Developers of **OptiqueVQS** [3] primarily wanted to make a product for end users who have no or very limited technical skills and knowledge. They tried

to simplify the interface for easier ways to address the basic tasks. Also, for the achievement of this core idea, the authors do not use a formal notation and syntax for query representation (but their syntax still conforms to the underlying formalism). They however employ a formal approach projecting the underlying ontology into a graph for navigation, which constitutes the backbone of the query formulation process.

NITELIGHT [4] has a GUI-support for creating SPARQL queries using a set of graphical notations and built-in editing operations. Also, the developers name features like ontology alignment, information integration, rule creation.

QueryVOWL [5] is a tool for visual querying which implements a graph-based approach. QueryVOWL relates to open web standards and does not use proprietary languages.

Smeagol [6] implements the "specific-to-general" paradigm which means an interface that explicitly supports starting with an example and generalizing it to find other similar examples. The authors suggest that it is friendly for novice users allowing them to effectively pose complex queries against a Semantic Web data set.

For the easy way of SPARQL query construction, developers of **SPARQL Assist** [7] offer to use context-sensitive type-ahead completion with prioritization of the most likely suggestions which are using their multi-lingual labels and descriptions. In addition to an assistance feature covering the basic syntax, ontological terms are indexed by their labels, using the xml:lang attribute to record the language of each label for each term.

The authors of **XSPARQL-Viz** [8] are positioning the project as a tool implementing a mashup-based approach for auto-generation of XSPARQL queries. Its visual query editor facilitates mapping between XML and RDF data sets. Query results can be transformed into any desired output format or as input for another query. One of its specific features is the capability of auto-generating an XSD-schema for XML-data sets and RDFS-schema for RDF-data sets.

According to the developers of **Ontology-Based Graphical Query Language** [9], the main purpose of the system is to enable efficient querying on ontologies even by novice users who do not have an in-depth knowledge of internal query structures. This should allow the users to construct query graphs by interacting with the ontology in a user-friendly manner. The system also supports graphical recursive queries and methods to interpret recursive programs from these visual query graphs.

The main feature of **NL-Graphs** [10] is combining two query approaches (graph-based and natural language) as a hybrid query approach.

4.2 Comparison of Query Builders

In order to derive criteria for the comparison of query builders, major design goals have been identified in the selected papers. Their design goals can be split into two parts: *common goals* - goals which were mentioned in several articles, and *others* - goals which are touched only in one article.

Common Goals Number in brackets is a count of articles mentioning the goal.

- *Usability* (8)
  Availability of user-friendly interface, GUI query builder with drag-n-drop support and other features which make using the tool easier for a casual user.
- *Expressivity* (4)
  So that formulation of advanced queries is possible despite a high-level visual query language.
- *For end users* (4)
  So that query formulation does not require a significant IT-expertise.
- *Querying approach* (4)
  Form-based, graph-based, natural language, etc.
- *Ontology exploration* (3)
  To give users a possibility to become familiar with the domain before querying, e.g. ontology browsing.
- *Domain independence* (2)
  To avoid any assumptions as to the contents or structure of the data and thus potentially support any RDF data set.

Other Goals

- Adaptivity & adaptability
- Built-in query validity checker
- Internationalization
- Interoperability
- Modularity
- Open standards
- Recursive queries
- Reusability
- Scalability
- Support of different output formats
- Type-ahead completion

Comparison Criteria In most cases, researchers declare in their works such goals that were not reached by previous authors and that exist in respect to state of the art query builders. So they compare related work to desired results. And the desired results in the situation is what researches are going to build - new query builders. Therefore we can say that authors compare existing query builders to new ones whereby the comparison criteria are declared goals. That's why we feel free to use the goals derived in previous section as comparison criteria for query builders. By this approach, we hence inductively gather a set of comparison criteria that are both relevant and common. Regarding the former, relevance is ensured since a criterion already served comparisons in previous publications. Regarding the latter, we record the number of mentions and use only such criteria that have been mentioned by at least two papers.

The only exception is the goal "usability" which is not a functional and not formal goal, so it is hard to use this criteria objectively. However, as we

mentioned, in most cases, usability equated to the possibility of building queries with a GUI. So we can use "GUI query building" as criteria instead of the "usability" goal.

Also, we use the criteria "expressivity" to denote whether all (or nearly all) expressions and operators of SPARQL are supported by a query builder. If this is not the case, then it is up to the reader interested in a specific query builder to check whether a given construct of the language is covered or not. A detailed analysis of SPARQL coverage is left open for future work.

Builders Comparison Table Number in brackets is a number of section in paper describing the tool where confirmation of the given grade could be found.

Builder/Criteria	**Querying approach**	**Domain independence**	**Expressiveness**	**GUI query building**	**For end users**	**Ontology exploration**
QUaTRO2	graph-based (3)	+ (1)	- (5)	+ (4.2)	+/- * (5)	+ (4.2)
OptiqueVQS	graph-based (2.3)	+ (2.3)	- (5)	+ (2.1)	+/- * (3)	+ (2.3)
NITELIGHT	graph-based (3.4)	+ (5.2)	+ (3.7)	+ (4.1)	+/- * (6)	+ (4.2)
QueryVOWL	graph-based (2)	+ (4)	- (3.4)	+ (3.3)	+ (3)	- (5.1)
Smeagol	graph-based (2)	+ (4)	- (4)	+ (3.1)	+ (8)	+ (3)
SPARQL Assist language-neutral query composer	raw query (3)	+ (2)	+ (3)	- (3)	- (3)	- (3)
XSPARQL-Viz	form-based (1)	+ (3)	+ (2)	+ (3)	+ (1)	+ (2)
Ontology Based Graphical Query Language	graph-based + recursion support (1.2)	+ (2.1)	- (4)	+ (2.2)	+/- * (2.2)	+ (2.1)
NL-Graphs	graph-based + natural language (2)	+ (3.1)	+ (2)	+ (2)	+ (3.3)	- (2)

* - Paper describing QueryVOWL shows that visual notation used by this tool is still very close to the RDF and SPARQL syntax which is a problem for lay users who are not familiar with the low-level semantics of RDF graphs [5][11].

4.3 Usage Outside the Semantic Web-/Linked Data-Community

The Semantic Web- and Linked Data-community is very established worldwide. One of the most large-scale projects in the realm of these communities is DBPedia. The DBpedia Ontology is a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia.

But often, all Visual Query Builder (VQB) systems do not go beyond the community area and exist only at the stage of research projects, not commercial products.

The first was described by McCarthy et al. in "SPARQL Assist language-neutral query composer" [7]. The main application idea was to let bioinformaticians quickly generate a request with a help of context-sensitive type-ahead completion. Here, words are predicted by previously declared variables or known individuals.

The second example is OptiqueVQS [3] which was evaluated by experiments conducted with Statoil ASA and Siemens AG. The first one was carried out on an oil & gas ontology, which in total includes 253 concepts, 208 relationships, and 233 attributes. For the Siemens experiment, a diagnostic ontology was provided which includes 5 concepts and relationships, and 9 attributes. All of the ontologies and datasets in both cases are not public. The authors did not report of usage of OptiqueVQS in the companies on a permanent basis. However, they emphasized on the fact that the experiments results indicated a high effectiveness and efficiency thus suggesting that the system is a viable tool for users without any technical background to construct considerably complex queries.

4.4 Empirical Evaluation

Due to the lack of industrial experiments, developers mostly evaluated their systems on small groups of people. Therefore, most tools are not evaluated sufficiently. In general, tests should evaluate usability and learnability. However, it is hard to isolate any empirical results of such tests.

Sadamandan et al. try in [9] to assess the quality of the system parameters through Effectiveness, Efficiency and Satisfaction of users. The first and the second of the criteria were calculated by the following formula: "*(Yes + (Partial x 0.5)) / Total x 100%*". In more detail, users had to perform a set of tasks regarding SPARQL querying in an experiment. "Yes" is the number of successfully completed tasks by the users. "Partial" is the number of partially completed tasks. User satisfaction was calculated using post questionnaires with the participants rating satisfaction on a Likert scale. All three scores were mapped to a percent scale with a maximum of 100% . Usability in this case is calculated as the average percentage of the three scores.

Clemmer and Davies [6] conducted a classical experiment for evaluation. The experiments goal was to compare results of testing group (control) which uses existing application with results of another group (experimental) which works with the new solution. Identical tasks performed by the groups ensure the relevance of the results. In this approach, empirical results for each task are

the percentage of the number of correct answers to the number of people in the group.

Elbedweihy et al. [10] used a standardized usability questionnaire in order to assess their NL-Graphs approach and provide an detailed description of the results. However, a lot of articles provide just a description of an experiment, number of participants and the final results of the experiment without presentation of the experiments process and without any numbers, calculations or tables. In such works, developers use users comments or other kind of feedback for evaluation.

4.5 Scaling with the Large Data or Expressive Ontology

Usability of VQB systems should appear not only in ease of operations for non-technical users, but also in the possibility to provide convenient ways to work with large data and expressive ontologies. So the question addresses scalability of query builders. In this list of articles we found three systems which take this problem into account.

One of the solutions is to provide on-demand access to a relevant part of the ontology in order to avoid working with the whole data. At the same time, a ranking approach offers auto-completion at every step of the query definition. A combination of these features at the same time ensures the absence of a high computational load and better user orientation in the ontology.

Mostly, VQB systems use the approach of adding criteria to narrow the results of a query. The developers of Smeagol call this approach general-to-specific. In contrast, they suggest an approach called specific-to-general [6]. Their approach is based on expansion: a generalization of a specific example's parameters allows to find other results similar to this template. Authors believe that it is more convenient for users to express questions by means of an example.

The last paper mentioning this problem described the method of recursive query building [9]. The recursive method helps to avoid the need of drawing up a complex query to retrieve information from a graphs of unknown depth. Also, as the authors say, the system is designed in such way that it can handle large datasets using fast indexing mechanisms.

Returning to the problem of the specificity of ontologies and linked data, it is worth to emphasize that the absence of interfaces of the query builders to data providers such as SPARQL endpoints or triplestores does not allow to empirically evaluate scalability issues.

5 Conclusion

We conducted a systematic review of literature on the topic of SPARQL query builders. Design goals for query builders were summarized and reviewed. We derived suitable criteria for comparing SPARQL query builders and presented a table with a comparison of the builders mentioned in the selected papers.

Also, a discussion concerning usage of the builders outside of the Semantic Web community was conducted. We described methods for empirical evaluation of query builders and reviewed possibilities of query builders scaling when data becomes large or the ontology is expressive.

Regarding the comparison of query builders it can be concluded, that all of the found tools are domain- independent. Also, most of them are end user-oriented and provide a GUI for query construction. However, just half of them support the full expressiveness of SPARQL. This could be a starting point for future research - is the full expressiveness required for practical usage scenarios? Regarding practical usage and evaluation, no industry-scale or business application has been noted among the identified approaches except for OptiqueVQS.

Acknowledgment. The work has been partially supported by the Government of Russian Federation, Grant 074-U01.

References

1. Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic Literature Reviews in Software Engineering - A Systematic Literature Review. Inf. Softw. Technol. **51**(1) (January 2009) 7–15
2. Balis, B., Grabiec, T., Bubak, M.: Domain-Driven Visual Query Formulation over RDF Data Sets. In Wyrzykowski, R., Dongarra, J., Karczewski, K., Waniewski, J., eds.: Parallel Processing and Applied Mathematics. Number 8384 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (September 2013) 293–301 DOI: 10.1007/978-3-642-55224-3_28.
3. Soylu, A., Kharlamov, E., Zheleznyakov, D., Jimenez-Ruiz, E., Giese, M., Horrocks, I.: Ontology-Based Visual Query Formulation: An Industry Experience. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., Pavlidis, I., Feris, R., McGraw, T., Elendt, M., Kopper, R., Ragan, E., Ye, Z., Weber, G., eds.: Advances in Visual Computing. Number 9474 in Lecture Notes in Computer Science. Springer International Publishing (December 2015) 842–854 DOI: 10.1007/978-3-319-27857-5_75.
4. Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.R.: A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In Gangemi, A., Euzenat, J., eds.: Knowledge Engineering: Practice and Patterns. Number 5268 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (September 2008) 275–291 DOI: 10.1007/978-3-540-87696-0_25.
5. Haag, F., Lohmann, S., Siek, S., Ertl, T.: QueryVOWL: A Visual Query Notation for Linked Data. In Gandon, F., Guret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A., eds.: The Semantic Web: ESWC 2015 Satellite Events. Number 9341 in Lecture Notes in Computer Science. Springer International Publishing (May 2015) 387–402 DOI: 10.1007/978-3-319-25639-9_51.
6. Clemmer, A., Davies, S.: Smeagol: A Specific-to-General Semantic Web Query Interface Paradigm for Novices. In Hameurlain, A., Liddle, S.W., Schewe, K.D., Zhou, X., eds.: Database and Expert Systems Applications. Number 6860 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (August 2011) 288–302 DOI: 10.1007/978-3-642-23088-2_21.
7. McCarthy, L., Vandervalk, B., Wilkinson, M.: SPARQL Assist language-neutral query composer. BMC Bioinformatics **13**(S1) (January 2012) 1–9

8. Gillani, S.Z.H., Ali, M.I., Mileo, A.: XSPARQL-Viz: A Mashup-Based Visual Query Editor for XSPARQL. In Cimiano, P., Fernndez, M., Lopez, V., Schlobach, S., Vlker, J., eds.: The Semantic Web: ESWC 2013 Satellite Events. Number 7955 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (May 2013) 219–224 DOI: 10.1007/978-3-642-41242-4_28.
9. Sadanandan, A.A., Onn, K.W., Lukose, D.: Ontology Based Graphical Query Language Supporting Recursion. In Setchi, R., Jordanov, I., Howlett, R.J., Jain, L.C., eds.: Knowledge-Based and Intelligent Information and Engineering Systems. Number 6276 in Lecture Notes in Computer Science. Springer Berlin Heidelberg (September 2010) 627–638 DOI: 10.1007/978-3-642-15387-7_66.
10. Elbedweihy, K., Mazumdar, S., Wrigley, S.N., Ciravegna, F.: NL-Graphs: A Hybrid Approach toward Interactively Querying Semantic Data. In Presutti, V., dAmato, C., Gandon, F., dAquin, M., Staab, S., Tordai, A., eds.: The Semantic Web: Trends and Challenges. Number 8465 in Lecture Notes in Computer Science. Springer International Publishing (May 2014) 565–579 DOI: 10.1007/978-3-319-07443-6_38.
11. Dadzie, A.S., Rowe, M.: Approaches to Visualising Linked Data: A Survey. Semant. web **2**(2) (April 2011) 89–124