QueryVOWL: A Visual Query Notation for Linked Data

Florian Haag[(✉)], Steffen Lohmann, Stephan Siek,
and Thomas Ertl

Institute for Visualization and Interactive Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{florian.haag,steffen.lohmann,thomas.ertl}@vis.uni-stuttgart.de

Abstract. In order to enable users without any knowledge of RDF and SPARQL to query Linked Data, visual approaches can be helpful by providing graphical support for query building. We present QueryVOWL, a visual query language that is based upon the ontology visualization VOWL and defines mappings to SPARQL. We aim for a language that is intuitive and easy to use, while remaining flexible and preserving most of the expressiveness of SPARQL. In contrast to related work, the queries can be created entirely with visual elements, taking into account RDFS and OWL concepts often used to structure Linked Data. This paper is a revised version of a workshop paper where we first introduced QueryVOWL. We present the query notation, some example queries, and two prototypical implementations of QueryVOWL. Also, we report on a qualitative user study that indicates lay users are able to construct and interpret QueryVOWL graphs.

Keywords: Visual querying · VOWL · QueryVOWL · Visualization · Linked Data · SPARQL · RDF · OWL · Semantic Web

1 Introduction

An increasing amount of Linked Data is being published and ready for consumption [4,12]. The data is not only of interest to the Semantic Web community but first and foremost to lay users and domain experts from different areas [17]. A large portion of Linked Data is available in RDF format and can be queried using the standardized query language SPARQL [6,12]. However, writing SPARQL queries is not an easy task and requires technical knowledge on RDF, HTTP, and IRIs, among others. Lay users cannot be expected to have this knowledge, but visual interfaces can provide graphical support for querying Linked Data. The interfaces must enable the flexible creation of queries without any knowledge of RDF, SPARQL, and related Semantic Web technologies.

Experience from relational databases and SQL querying can only partly be reused, as the data is organized in fixed table structures in those databases. Linked Data, by contrast, is often represented as an RDF graph, which is more

related to the representation of data in graph databases, and SPARQL is used to retrieve information from this graph-based data. Appropriate solutions must therefore address the unique specifics of SPARQL and Linked Data, such as the schema-independent description of resources and the use of IRIs for global identification.

This paper presents QueryVOWL, a novel approach for visual querying that reuses graphical elements from the Visual Notation for OWL Ontologies (VOWL) [28] and defines SPARQL mappings for them. The paper is a revised and extended version of our paper for the HSWI workshop [21], where we first introduced QueryVOWL.

2   Related Work

Several approaches to support the querying of Linked Data have been proposed in the last couple of years. A popular paradigm is form-based querying, where the queries are composed by entering variables, identifiers, and other query components using form elements, such as text boxes with auto-completion features, drop-down lists, and radio buttons. Examples of form-based querying include SPARQLViz [14], Konduit VQB [7], PepeSearch [22], or DBpedia's Graph Pattern Builder [8]. While form-based querying can be very usable, it offers a rather linear way of query building that is less flexible than other querying paradigms. Furthermore, most of the available approaches are not designed for lay users but for people who have at least some knowledge of RDF and SPARQL and are familiar with the triple representation.

Graph-based querying usually provides more flexibility than the form-based paradigm by using node-link diagrams to create arbitrary SPARQL query patterns. Examples for such approaches include NITELIGHT [31], iSPARQL [5], RDF-GL [25], and LUPOSDATE [18]. However, the visual query languages used in these tools are still very close to the RDF and SPARQL syntax: Although the triples are visually combined to node-link diagrams, they strictly follow the subject-predicate-object notation from RDF instead of providing a higher degree of abstraction. While this is fine for expert users, lay users are known to have problems with the low-level semantics of RDF graphs [17].

The same holds true for many works that visualize queries on a slightly higher degree of abstraction. One such approach supports the composition of SPARQL queries with UML-based diagrams [9]. These diagrams can further reduce the challenges of querying Linked Data, but they are still comparatively difficult to use for lay users [29].

Other approaches completely depart from the SPARQL syntax. For instance, SparqlFilterFlow [19] supports the visual composition of SPARQL queries by letting users create filters connected by flows. However, edges in SparqlFilterFlow represent logical connections between filter criteria rather than property links between classes or individuals. Thus, the focus is on the logical combination of filter criteria, whereas object relations made explicit in QueryVOWL are not directly displayed.

Furthermore, Linked Data can be queried as part of the browsing process by generating and sending SPARQL queries in the background. Examples of such Linked Data browsers include Tabulator [11], Disco [2], and gFacet [24], among others [17]. These browsers are comparatively easy to use, but rely on particular patterns of queries and are therefore limited in their flexibility and expressiveness. Similar constraints apply to visual approaches that query Linked Data for specific purposes, such as relationship discovery [23] or to explore context information about locations [10].

QueryVOWL is related to visual querying approaches for graph databases, such as qGraph [13], or a visual graph-based system for genomics data [16]. In contrast to those attempts, QueryVOWL specifically addresses RDF and SPARQL that Linked Data is usually based on, and defines reusable mappings for the visual language. It is therefore related to open web standards and the well-specified VOWL notation. This is different from visual querying approaches in the context of graph databases, which often use underspecified or proprietary languages supported only by specific graph databases.
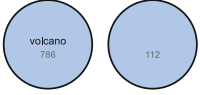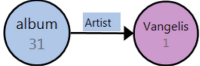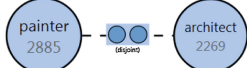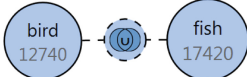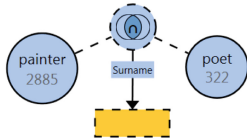
3 QueryVOWL

We decided to base the visual query language on the VOWL notation, which has proven to be comparatively intuitive and understandable, also and especially to lay users [28,29]. Furthermore, it provides the degree of abstraction we consider helpful to ease the query building, as VOWL has been designed for RDFS and OWL, and concepts from these vocabularies are often used to structure Linked Data.

3.1 VOWL

VOWL defines mappings of OWL language constructs to graphical elements that are combined to node-link diagrams. Figure 1 shows the VOWL visualization of a small ontology created with WebVOWL 0.4 [27]. Classes are represented by circles that contain the class name, whereas datatypes are displayed as rectangles with a border. Property names are shown inside borderless rectangles that are complemented by arrow lines indicating the direction of the properties. Some language constructs are expressed in a different way, such as subclass relations or special OWL classes.

In addition, VOWL comes with a set of colors that are defined in an abstract way according to their function in order to allow for custom color schemes. This leaves the freedom to use custom color schemes beside the default scheme recommended by the specification. For each visual element, the applicable colors are specified in abstract terms. For instance, classes can have the "general", "deprecated", or "external" color, datatypes and resources are always shown in their respective fixed color, and a "highlight" color is used to dynamically display certain features of elements in interactive contexts. Shapes and textual labels in VOWL have, however, been chosen in a way so no essential information

Table 1. Visual elements of QueryVOWL and their translation into SPARQL.

| QueryVOWL Element	SPARQL Mapping
(volcano 786) (112) — The VOWL class notation is used to represent sets of individuals. The label indicates the class the individuals are restricted to. If no label is set, the class of the individuals is not restricted. Similar to VOWL, the size of the circle may be used to roughly indicate the number of matching individuals. The exact number is additionally displayed.	Each set of individuals is represented by a unique variable. If the class of the individuals is restricted, that restriction is added as a triple: `?x a dbpo:Volcano .` The number of individuals is retrieved with the `count` function.
(Aracar 1) — The VOWL notation for RDFS classes is used to represent a single individual.	Single individuals are represented by their IRI: `dbpr:Aracar`
(album 31 → Artist → Vangelis 1) — The VOWL property notation is used to represent properties that connect sets of individuals and/or single individuals. Like in VOWL, the arrow line indicates the direction of the property.	Properties are represented as predicates in the triples: `?x a dbpo:Album .` `?x dbpp:artist dbpr:Vangelis .`
(child, politician 1075, 1388) — If the VOWL property notation is used without an arrowhead, matching properties can point in both directions.	`?y a dbpo:Politician .` `{ ?x dbpo:child ?y . }` `UNION` `{ ?y dbpo:child ?x . }`
(astronaut 97 — Surname — ▯ — Surname — cyclist 154) — The VOWL literal notation is also used in QueryVOWL. It can be restricted either by using specific values or by applying filters, such as range or pattern restrictions. Literals can be connected to more than one property to define that they shall have the same value as range.	Each literal is represented by a unique variable used in all triples and filters that refer to that literal: `?x a dbpo:Astronaut .` `?y a dbpo:Cyclist .` `?x foaf:surname ?v .` `?y foaf:surname ?v .`
(painter 2885 — (disjoint) — architect 2269) — The VOWL notation for disjointedness is used to define that the individuals in different sets are disjoint.	`?x a dbpo:Painter .` `?y a dbpo:Architect .` `FILTER(?x != ?y).`
(bird 12740 — (U) — fish 17420) — The VOWL union notation is used to represent the set of individuals that are contained in *at least one* of the connected nodes.	`?x a dbpo:Bird .` `?y a dbpo:Fish .` `{ ?s a dbpo:Bird . }` `UNION` `{ ?s a dbpo:Fish . }`
(painter 2885, (∩), poet 322, Surname) — The VOWL intersection notation is used to represent the set of individuals that are contained in *all* connected nodes.	`?x a dbpo:Painter .` `?y a dbpo:Poet .` `?s a dbpo:Painter ;` `   a dbpo:Poet ;` `   foaf:surname ?n .`
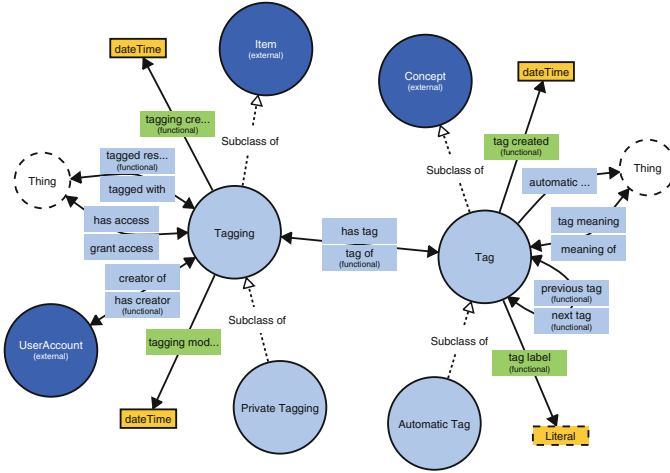
Fig. 1. Small ontology (MUTO [26]) visualized with WebVOWL 0.4 [27].

is lost if the colors are absent [28]. All elements and visual attributes of VOWL are precisely defined in a specification document [30].

3.2 Visual Elements

In contrast to VOWL, which has been designed to visualize complete ontologies, the purpose of QueryVOWL is to express user-defined filter criteria for searching for specific RDF graphs in Linked Data. The basic idea is to visually model a partial graph that is presumed to exist in a dataset. The graph defines certain restrictions, with some of its elements being placeholders. This mimics SPARQL, which allows to define graph patterns where some elements are variables.

When a QueryVOWL graph is applied to a given RDF dataset, all subgraphs from the dataset that match the query are retrieved, as with a SPARQL query. One difference is that the SPARQL query explicitly specifies the format and selection of results (for instance, as a list of table columns), and so do visual queries in related query visualizations [18,31]. In contrast, QueryVOWL enables users to dynamically explore the matches for parts of the graph: QueryVOWL users can select any of the visual query elements to retrieve the set of matching resources. Visualization approaches outside the scope of QueryVOWL can then be used to display the results in a user-friendly way, for instance, on a map or timeline as in NITELIGHT [31] and similar tools.

We started out from the VOWL specification and reused elements and definitions as appropriate for building query graphs. In contrast to VOWL, where each visual element represents a particular conceptual element from the TBox of an ontology [28], visual elements in QueryVOWL can also act as placeholders that are not fully specified on a TBox level, and for which restrictions can be added by the user. Therefore, some VOWL elements had to be adapted to indicate the

variability of the IRIs or values they represent, and to provide for the interaction options that users require to specify their query. Unlike other notations [25,31], users do not get in touch with variable names.

The VOWL property notation is used to represent properties that connect specific individuals or sets of individuals, analogously to related work [5,24,25]. Different from VOWL, QueryVOWL allows to add properties without specifying the direction. In these cases, matching properties can point in either direction. It also permits empty property labels, in which case all matching properties are considered. This is related to the idea of the RelFinder [23], in particular, if properties and classes are combined to chains.

Literal nodes can be connected to several datatype properties (also of different objects) to enforce that only individuals with the same value for these properties are found, like in other approaches [18].

Table 1 (above) outlines the visual elements that QueryVOWL consists of, as well as their mappings to SPARQL query fragments. Figure 2 shows a small QueryVOWL graph assembled from the visual elements, along with the SPARQL query that results from the graph based on the selected element.


```
 1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
 2 PREFIX dbpp: <http://dbpedia.org/property/>
 3 PREFIX dbpr: <http://dbpedia.org/resource/>
 4 PREFIX dbpo: <http://dbpedia.org/ontology/>
 5
 6 SELECT ?Node1
 7 WHERE {
 8   ?Node1 a foaf:Person .
 9   ?Node1 dbpp:almaMater ?Node2 .
10   dbpr:Barack_Obama dbpo:almaMater ?Node2 .
11   ?Node2 a dbpo:University .
12   { ?Node1 ?Link5 dbpr:Barack_Obama . }
13    UNION { dbpr:Barack_Obama ?Link5 ?Node1 . }
14   ?Node1 dbpp:party ?Node3 .
15   ?Node3 a dbpo:PoliticalParty .
16 }
```
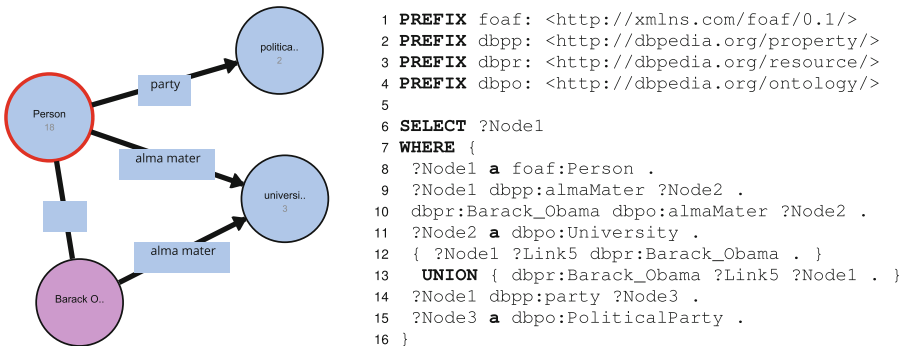
Fig. 2. Example of a QueryVOWL graph, along with the SPARQL query resulting from that graph (when class *Person* is focused, as indicated by the red border) (Color figure online).

3.3   Interactive Editing

WYSIWYG editing of the query graph can be allowed by adding interactive features to the aforementioned elements. The following four functions are required:

Delete: All visual elements contain a delete button so that they can be removed.
Connect: Properties can be added as links, either as unspecified properties or by choosing from a list of available properties.
Substitute: Any class, individual, and property can be replaced by another class, individual, or property, based on a list of available choices.

Edit: Restrictions on classes, properties, and literal values can be set or removed.

These interactive features may remain hidden unless the elements are pointed at (Fig. 3). In that case, additional information (e.g., IRIs or long labels) or interactive elements that would otherwise not have enough display space may also be shown.
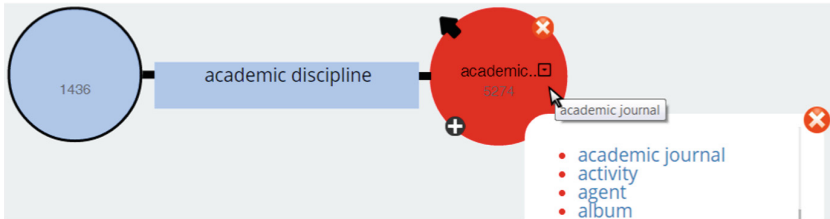
Fig. 3. Interaction elements are usually hidden and only appear on demand.

Whenever the graph structure or restrictions are modified, any connected class nodes will dynamically update their counts. This helps users immediately recognize the effects of their changes and provides them with a way to estimate whether further extensions or restrictions are required to retrieve a meaningful result. Some nodes can be excluded from this update process to reduce server load: Subgraphs that are *exclusively* connected via nodes restricted to specific individuals are only included in the SPARQL query if they contain the focused element. To retrieve all that information, as well as the final result set, the internally generated SPARQL queries are sent to a SPARQL endpoint that can be chosen as a backend in the visualization.

Generation of the SPARQL queries requires only a negligible amount of time, as this merely requires an iteration over all elements found in the query graph, while the statements expressed by these graphical elements are added to the resulting SPARQL query step-by-step. For a QueryVOWL graph that consists only of n class and/or individual nodes and m edges, time complexity of this SPARQL query generation remains within $\mathcal{O}(n \cdot m)$. Depending on the SPARQL engine and the triple store running on the server, the processing time for the query may vary significantly, though.

3.4 Language Limitations

QueryVOWL covers a part of the SPARQL query language, but, to date, also omits some elements. Literal nodes can be restricted based on constant values, and they can be used to express that several individuals are connected to the same property value. A visual representation for other types of relationships, such as inequality or asymmetric relationships (greater than, less than, etc.),

has not yet been defined. Furthermore, we focused on a straightforward setup
where a query is sent to the default graph of a dedicated endpoint. Federated
queries or named graphs are currently not included in QueryVOWL, although it
should be noted that implementations might support such features as a part of
their backend configuration, without any explicit indication in the QueryVOWL
visualization.

There are also some OWL concepts represented by graphical elements in
VOWL, for which we did not define related QueryVOWL elements yet. While
it might be desirable to create a query where something is connected to the
complement of a set restricted by filters, we have not yet devised a SPARQL
mapping for such an element. Likewise, cardinalities might be added to the visual
notation—for instance, to search for all individuals of a given type that have at
most two values for a given property—, but we deemed the SPARQL represen-
tation of such a restriction too problematic at the current state of development.

4 Exemplary Queries

The following examples illustrate how the visual elements of QueryVOWL can
be assembled to query graphs. As QueryVOWL is independent of any particular
dataset, we are using different datasets in the examples, all accessed by their
SPARQL endpoints.
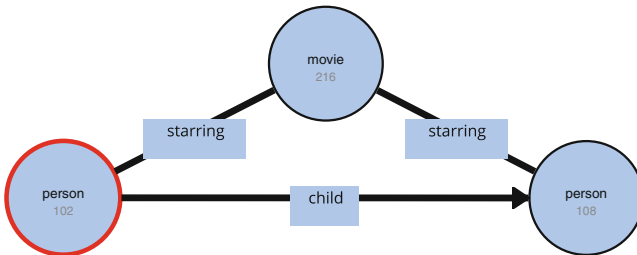
Fig. 4. DBpedia knows about 102 persons who starred in movies together with at least
one of their children.

Who *starred* in a *movie* together with his or her *child*? Figure 4 shows
a QueryVOWL graph based on DBpedia for retrieving any movies along with
two of their actors, one of whom must be the child of the other. The latter actor
is focused, as the graph is used to identify the elder actor of the two (according
to the direction of the property *child*).

**Which *authors* published on both conferences *ESWC* and *WWW*
in the same *year*?** The QueryVOWL graph created for the Faceted DBLP
dataset [3] is depicted in Fig. 5a. It asked for authors of two works, which are
linked via the year of issue to indicate that they were published in the same year

(any same year). One of the works should belong to the series *ESWC*, the other one to the series *WWW*.

Which *countries* have at least two different *industries* and participate in the *World Health Organization*? This query is shown in Fig. 5b, based on the CIA World Factbook [1]. A *disjoint* edge is used to indicate that the two *Industry* nodes are supposed to map to different individuals in each result. WHO is represented by an individual node.
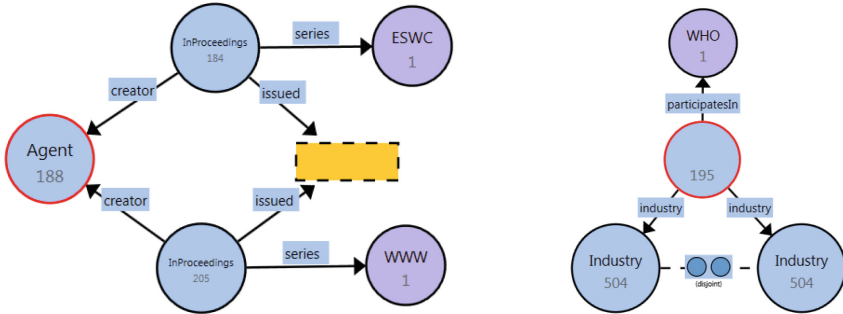
(a) There are 188 authors who published at ESWC and WWW in the same year according to Faceted DBLP.

(b) The CIA World Factbook contains data on 195 countries that have at least two different industries and participate in the World Health Organization.

Fig. 5. Examplary QueryVOWL graphs.

5   Evaluation

We have evaluated the applicability and usability of the approach by implementing it in two interactive prototypes and by conducting a qualitative user study.

5.1   Implementations

The two prototypes are based on different technologies to verify various aspects of the approach and to get an idea of how well it can be implemented with different frameworks and development techniques.

Web-Based Implementation. The web-based prototype (Fig. 6a) implements the main elements of the visual query language and provides an opportunity to try the look and feel of an interactive QueryVOWL implementation.[1] It is based

---

[1] The web-based prototype is available at http://queryvowl.visualdataweb.org.
  A demo of that prototype has been presented at ESWC 2015 [20].

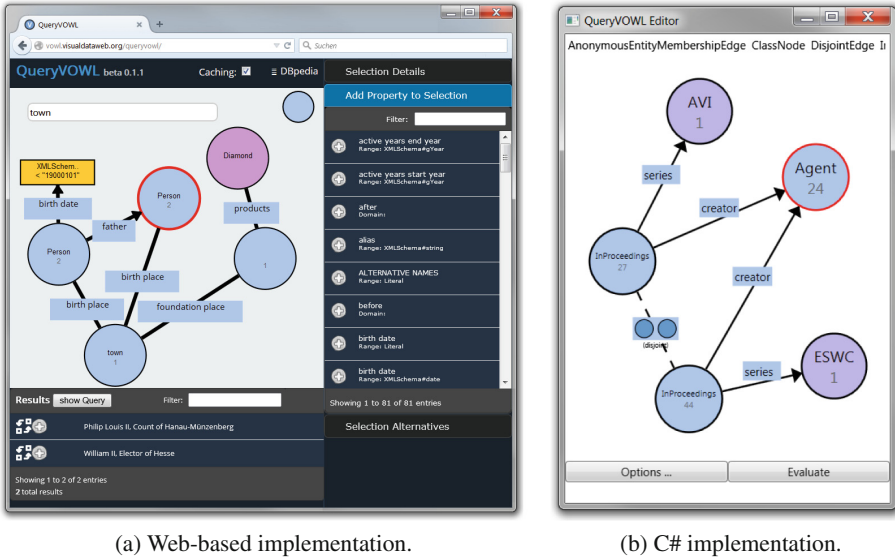(a) Web-based implementation.   (b) C# implementation.

Fig. 6. Screenshots of two prototypical QueryVOWL implementations.

on open web standards (HTML, JavaScript, CSS, SVG) and integrates some JavaScript libraries, most importantly D3 [15] for the visualization of the query graph.

Users can create and modify QueryVOWL graphs by adding and removing visual elements as well as positioning nodes with drag-and-drop. Restricted and unrestricted class nodes, properties (both directed and undirected), individuals, and literal nodes with filters for some ordinal types are supported. The union, intersection, and disjointedness operators, as well as the mapping of result set sizes to class node radii, have not yet been included. Query building is supported by automatic updates upon changes to the graph, asynchronous loading of lists of resources compliant with the current selection, and configuration options that are displayed upon hovering over elements.

A sidebar provides information about the selected element, as well as options to modify its filter restrictions and to add linked elements. A result list at the bottom shows individuals that are valid replacements for the selected node.

Stand-Alone Desktop Application. The desktop application (Fig. 6b) runs on the Microsoft .NET Framework and was created in C# with the Windows Presentation Foundation (WPF) user interface toolkit. It is intended as a showcase for the object-oriented implementation of the QueryVOWL elements that uses polymorphism for the generation of SPARQL query strings based on the rules outlined in Table 1.

All elements listed in the table are implemented in this prototype, but interactivity is limited. The prototype supports drag-and-drop, dynamic node scaling,

and the insertion of IRIs from the system clipboard. As in the web implementation, SPARQL queries are automatically generated and sent to a given SPARQL endpoint. Once the requests are answered, the retrieved result counts are displayed and nodes are scaled accordingly.

5.2   User Study

We have conducted a qualitative user study to gather further insight into the comprehensibility of QueryVOWL, the usability of our interactive implementation, and some general comments on the visual query language.

Tasks. We prepared a total of eight tasks based on data from the DBpedia dataset. While the study was conducted in German, much of the structural information in the DBpedia dataset uses English. Therefore, all tasks were provided bilingually, to help participants bridge any possible gaps in their English knowledge.
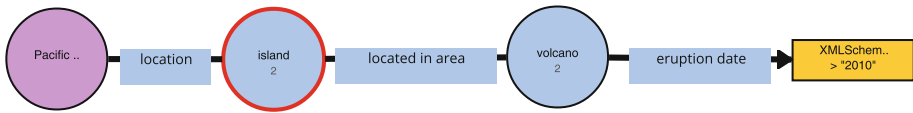
Fig. 7. One of the query graphs that participants of the user study had to construct. It can be used to answer the question "How many islands contain a volcano and are located in the Pacific Ocean?"

Seven tasks consisted of a natural language question, and possibly some more specific sub-questions. Users were asked to construct a QueryVOWL graph that represents the question with our web-based prototype (like the one in Fig. 7) and to select the appropriate element in the graph to find an answer to the question. Answering the question meant showing the graph and explaining briefly where on the screen the response to the question can be found. The full set of construction questions is listed in Table 2.

The eighth task was a comprehension task, in which a QueryVOWL graph with a selected node was shown (Fig. 8). Users were asked to express the query represented by the graph as a natural language question.

In all, the tasks in the study made use of the QueryVOWL features available in the web-based implementation. They made use of labeled and unlabeled class nodes, individuals, directed and undirected property edges, as well as literal values.

Material. A MacBook Air with a 13.3 in. display, a screen resolution of 1440×900 pixels, and an external mouse was used during the study. The QueryVOWL implementation was executed in a Mozilla Firefox 31 browser in full-screen mode.

Table 2. English text of the construction tasks from the user study. For tasks split up by forward slashes (/) in this table, participants had to incrementally assemble the query in a stepwise manner.

| #	Question
1	How many films did Bruce Willis star in? / Does anyone appear together with his or her child in any of these films? / What film and which child?
2	How many persons / wrote a song / and are connected to a band in some way / that Freddie Mercury used to belong to? / How many bands are there?
3	How many persons / are spouses of a senator / and have any children born in or after 1935? / How many children are there?
4	How many mountains are there in Madagascar? / How many rivers / originate from one of those mountains / and flow into the Indian Ocean?
5	How many bridges are there in Pittsburgh? / How many of them span a river? / How many of them span the Ohio River?
6	How many islands / contain a volcano / and are located in the Pacific Ocean? / How many of the volcanos erupted after 2010?
7	How many politicians / have a spouse / and have the same birthdate as their spouse?

All on-screen activity was captured by a screen recording software to ease the analysis.

The tasks, as well as a questionnaire on demographic data and the participants' impression of QueryVOWL, were printed on paper. An introductory video with a runtime of approximately 4.5 min was prepared. It explained QueryVOWL by constructing an exemplary query step by step.

Participants. Six participants (3 female, 3 male) between the age of 22 and 43 (median: 26) took part in the user study. All of them had different professions, none of them from the field of information technology. None of the participants had any prior experience with ontologies or the Semantic Web. Therefore, we could ensure that participants did not bring any prior knowledge on querying Linked Data, which might bias the results.

Procedure. The study was conducted in a closed room, with one participant at a time. Participants were first shown the introductory video and were asked to complete a training task to get to know the visualization and the user interface. Subsequently, the sheet with the questions was handed out, and screen recording was started.

After reading each of the tasks, participants were given an opportunity to ask questions in the case of doubts about the tasks. Participants would then start solving the tasks, while the interaction steps were noted down.
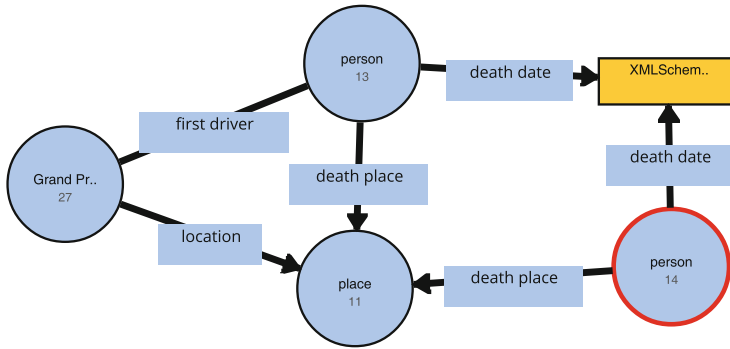
Fig. 8. The query graph shown to the participants of the user study. Participants had to recognize that this graph can be used to find people who passed away on the same date and at the same place as the first driver of a Grand Prix.

Finally, participants were asked to complete the questionnaire to gather information on which parts of the visualization caused confusion and which elements were helpful for understanding the queries.

Results. Participants could solve most of the tasks. Some adapted their initial query to reach a correct solution. There was a noticeable preference for elements and features that had been presented during the introductory video. Moreover, when constructing QueryVOWL graphs, participants followed the provided questions very closely and used exactly the words and the order of words found in the questions.

In general, the use of class nodes and properties was clear. Participants could understand the basic graph structure and correctly identify which graph element represented the entity searched for. Likewise, five of the six participants could easily read the visualized query in the comprehension task. The only difficulty seemed to be the distinction between class nodes and individual nodes, whose difference in color was either not understood or not even consciously noticed by participants.

In a few cases, participants got confused during the composition tasks over the distinction between classes and properties. While they correctly identified *spouse* as a relationship between two persons, they expected a class *child* rather than a *child* property. Moreover, when participants were aware they had to use a property, participants sometimes were unsure about its direction, for example, whether the *child* property points from the parent to the child ("has the child") or vice-versa ("is child of").

Participants could flawlessly understand and use the literal node for single property values, even though it had not been shown in the introductory video. The only difficulty arose when two persons with the same birth date had to be found. Almost all participants expected to make the comparison explicit by two

connected literal nodes, rather than by simply linking the *birthDate* property of the two *Person* nodes to the same literal node.

All participants stated that they could imagine using the approach in everyday situations. Two of them stated the technique could be used in cases where conventional search engines are not sufficient, and two more participants could also imagine browsing data in QueryVOWL without having a specific goal in mind, as the information about possible extensions to the query is accessible in the interactive graph.

6   Conclusions and Future Work

We have built upon the ontology visualization VOWL to create QueryVOWL, a visual query language for Linked Data. Visual elements of VOWL were reused and adapted, and we have defined how the resulting graphs map to SPARQL queries. By using our web-based prototype, we have conducted a qualitative user study where we found that lay users could handle the basic query structure well, except for some more specific aspects of the visualization that were not immediately clear to the study participants.

Based on the user study, we believe that a brief but complete introduction to the visual notation is an efficient way to teach previously untrained users how to use QueryVOWL. Furthermore, user comments suggest that dynamically displayed explanations, and possibly a natural language representation of the query or parts thereof, may further support comprehension. We consider including these features in future versions.

Other suggestions referred to interaction features of the web-based implementation. Some interactive elements, such as the property direction toggle button, might be placed so as to avoid accidental clicking. Also, literal nodes could signal in a more obvious way that they can be connected to more than one class or individual node at a time.

Overall, QueryVOWL covers many concepts found in Linked Data. As the general approach appears to be usable, we would like to consider more advanced features, such as functions to process or transform property values for filtering. Likewise, enforcing comparison relationships between property values of two or more individuals beside equality could be desirable. Introducing existential or universal quantifiers as well as disjunctions between alternative filter restrictions could make QueryVOWL even more powerful, if appropriate ways of visualizing the concepts and mapping them to SPARQL can be found. Finally, the selection of new query elements currently happens primarily through lists of identifiers found in the SPARQL endpoint. Integrating an ontology or dataset overview visualization such as VOWL to select elements from might render the creation of QueryVOWL queries on unknown datasets more intuitive.

References

1. CIA world fact book in DAML. http://www.daml.org/2001/12/factbook/
2. Disco. http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/
3. Faceted DBLP. http://dblp.l3s.de
4. Linked data. http://linkeddata.org
5. OpenLink iSPARQL. http://oat.openlinksw.com/isparql/
6. SPARQL endpoints status. http://sparqles.okfn.org
7. Ambrus, O., Möller, K., Handschuh, S.: Konduit VQB: a visual query builder for SPARQL on the social semantic desktop. In: VISSW 2010, vol. 565. CEUR-WS (2010)
8. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
9. Bārzdiņš, G., Rikačovs, S., Zviedris, M.: Graphical query language as SPARQL frontend. In: ABDIS 2009, Workshops and DC, pp. 93–107. Riga Technical University (2009)
10. Becker, C., Bizer, C.: Exploring the geospatial semantic web with DBpedia Mobile. Web Semant. **7**(4), 278–286 (2009)
11. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: exploring and analyzing linked data on the semantic web. In: SWUI 2006 (2006)
12. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semant. Web Inf. Syst. **5**(3), 1–22 (2009)
13. Blau, H., Immerman, N., Jensen, D.: A visual query language for relational knowledge discovery. Computer Science Department Faculty Publication Series 105, University of Massachusetts-Amherst (2001)
14. Borsje, J., Embregts, H.: Graphical query composition and natural language processing in an RDF visualization interface. Bachelor's thesis, Erasmus University Rotterdam (2006)
15. Bostock, M., Ogievetsky, V., Heer, J.: D3 data-driven documents. IEEE Trans. Visual Comput. Graphics **17**(12), 2301–2309 (2011)
16. Bulter, G., Wang, G., Wang, Y., Zou, L.: A graph database with visual queries for genomics. In: Proceedings of the APBC 2005, pp. 31–40. Imperial College Press (2005)
17. Dadzie, A.S., Rowe, M.: Approaches to visualising linked data: a survey. Semant. Web **2**(2), 89–124 (2011)
18. Groppe, J., Groppe, S., Schleifer, A.: Visual query system for analyzing social semantic web. In: WWW 2011, pp. 217–220. ACM (2011)
19. Haag, F., Lohmann, S., Ertl, T.: SparqlFilterFlow: SPARQL query composition for everyone. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC Satellite Events 2014. LNCS, vol. 8798, pp. 362–367. Springer, Heidelberg (2014)
20. 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 Satellite Events 2015. LNCS, vol. 9341, pp. 62–66. Springer (2015)
21. Haag, F., Lohmann, S., Siek, S., Ertl, T.: Visual querying of linked data with QueryVOWL. In: Joint Proceedings of SumPre 2015 and HSWI 2014–15. CEUR-WS (to appear)

22. Heggestøyl, S., Vega-Gorgojo, G., Giese, M.: Visual query formulation for linked open data: the norwegian entity registry case. In: 27th Norsk Informatikkonferanse (NIK 2014). Bibsys Open Journal Systems (2014)
23. Heim, P., Lohmann, S., Stegemann, T.: Interactive relationship discovery via the semantic web. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part I. LNCS, vol. 6088, pp. 303–317. Springer, Heidelberg (2010)
24. Heim, P., Ziegler, J., Lohmann, S.: gFacet: a browser for the web of data. In: IMC-SSW 2008, vol. 417, pp. 49–58. CEUR-WS (2008)
25. Hogenboom, F., Milea, V., Frasincar, F., Kaymak, U.: RDF-GL: a SPARQL-based graphical query language for RDF. In: Chbeir, R., Badr, Y., Abraham, A., Hassanien, A.-E. (eds.) Emergent Web Intelligence: Advanced Information Retrieval. Advanced Information and Knowledge Processing, pp. 87–116. Springer, London (2010)
26. Lohmann, S., Díaz, P., Aedo, I.: MUTO: the modular unified tagging ontology. In: I-SEMANTICS 2011, pp. 95–104. ACM (2011)
27. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: web-based visualization of ontologies. In: Lambrix, P., Hyvönen, E., Blomqvist, E., Presutti, V., Qi, G., Sattler, U., Ding, Y., Ghidini, C. (eds.) EKAW 2014 Satellite Events. LNCS, vol. 8982, pp. 154–158. Springer, Heidelberg (2015)
28. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: user-oriented visualization of ontologies. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 266–281. Springer, Heidelberg (2014)
29. Negru, S., Haag, F., Lohmann, S.: Towards a unified visual notation for OWL ontologies: insights from a comparative user study. In: Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS 2013, pp. 73–80. ACM (2013)
30. Negru, S., Lohmann, S., Haag, F.: VOWL: visual notation for OWL ontologies (2014). http://purl.org/vowl/
31. Russell, A., Smart, P., Braines, D., Shadbolt, N.: NITELIGHT: a graphical tool for semantic query construction. In: SWUI 2008, vol. 543. CEUR-WS (2008)