User study thesis

There is a range of data analysis queries (involving aggregation and subqueries) over RDF data endpoints that for a certain group of IT-trained users (without specific background in SPARQL) is easier to compose in the visual notation and tool than write in the textual SPARQL notation.

The ease of query composition is measured in the terms of the number of queries the test participants are able to complete correctly within a given time limit.

User study design

The design idea of the user study is, as follows:

1) to have two groups of similar size of generally IT-trained persons without specific knowledge of RDF data querying;
2) to provide them with a fair introduction into RDF data querying languages SPARQL and ViziQuer, a broad range of supplementary materials, and a set of example queries (including machine readable ones);
3) to explain to both user groups the schema (vocabulary) of the data source to be queried (the vocabulary is to be made available in a machine readable form);
4) to assign to both groups of participants the same set of queries over the data source formulated in a natural language (with necessary disambiguation added and references to the data schema explicated); the query tasks shall correspond to different query patterns/language constructs needed to define the queries;
5) to ask both user groups to write the queries in a formal notation and execute them, the participants of one group are asked to write the queries in SPARQL, the participants of the other group are asked to write the queries in the ViziQuer visual notation; there is equal limited time given for all participants to complete the task;
6) to provide environments for both participant groups for query creation and execution and observing the results (basic support for name suggestion and syntactically correct query creation is to be provided for both user groups);
7) to compare the results quantitatively in terms of correctly defined queries by the participants, as well as to perform additional analysis (e.g. individual task execution time, the success rate for every task, most typical error analysis, etc.).

User study execution

The participants

The user study was offered on a voluntary basis to 30 students studying computing towards master degree within the frame of a 'Knowledge Engineering' course at the University of Latvia; the user study was performed by 28 of them.

The students have completed the undergraduate bachelor degree in IT or a similar discipline and most of them have some industry experience in an IT-related field. The students generally have some (at least academic) experience in SQL and relational databases, however, none of them have had any specific experience with SPARQL or RDF data querying. The students have been taught RDF principles as the subject of the previous lecture in the 'Knowledge Engineering' course.

The query environment

All user test participants were admitted to the ViziQuer tool environment (pointed to from http://viziquer.lumii.lv), each with his/her own login name and password (the user environments have been created and configured beforehand to access the hospital data ontology and a corresponding SPARQL endpoint, served by an OpenLink Virtuoso engine).

There were two projects HospitalA and HospitalB within the environment, each of them containing an empty diagram 'WorkingArea', further on there was a diagram 'ExampleQueries' in the HospitalB project (this diagram, although potentially very informative, was practically unusable due to system performance reasons, as explained below).

The HospitalA project was used to demonstrate the SPARQL-only window showing simultaneously both the SPARQL query and its execution results over the provided SPARQL endpoint.

The SPARQL query window contains a basic integration of YASQE platform providing SPARQL syntax coloring and a basic text completion for class and property names that was demonstrated to the query participants.

Then during the visual notation presentation stage all students were invited to the HospitalB project, where the attempt to open the ExampleQueries diagram by all users simultaneously was not successful (the system became non-responsive), so the visual query options (query node and link construction, property pane usage) were demonstrated within the empty WorkingArea.

The instructions for query environment reconstruction, as well as for entering the SPARQL only mode in a diagram can be found towards the end of this document.

The execution steps

During the study the students were given a 90 minute introduction to SPARQL, half of it serving as general presentation of SPARQL and the other half being devoted to the presentation of the SPARQL constructs necessary for the execution of the following user study queries. The presentation included brief RDF principle recap, the SPARQL query types, SPARQL select query constructs, including basic graph patterns, filtering, aggregation and subqueries.

There was also a visual presentation of the data source schema distributed to the students (as a hand-out, available also electronically), as well as electronically available text file with the data ontology prefixes and identifiers that could become necessary during the query composition over the data source.

The basic name completion services (available on hitting Ctrl-Space in the context of class or property name filling) have been demonstrated to the students. The students were also explained, how to use the available ontology identifier file for query creation.

The users were provided also an option to formulate and execute simple SPARQL queries over the hospital data endpoint within the HospitalA project, used further in the study.

The SPARQL presentation material is included on the user study supplementary material site, it has been made available also to the participants during the study.

The study followed by a 25 minute introduction to the ViziQuer visual notation and query tool. The principal elements of the presentation included:

- a hands-on demonstration of the creation of visual query elements (nodes, attributes, conditions, links) in the query tool, including the visual element property setting in the element property pane;
- explanation (on the basis of an example) of basic visual query patterns (class-attribute-link-condition, aggregation and statistics, subqueries).

Before the testing phase all students were given an extensive set of query examples both in a visual notation (as a hand-out, available also electronically in colored image format) and in SPARQL (a MS Word document with a two-column table, with the visual query image in Column 1 and machine-readable and copy-able SPARQL text in Column 2).

There was also a version of executable example visual queries provided to the students, however, this version turned out to be unusable for performance reasons: at the moment when all students opened the example diagrams, the query environment became non-responsive; there was the possibility for all students to leave the large example diagrams and enter empty diagrams, where the hands-on examples were demonstrated.

The users were split in two groups by distributing task sheets that were marked by the type of the environment (SPARQL or visual) the users have to use.

Each user was given a sheet with 10 basic query tasks (natural language description of the query) and asking to provide the time the user started to work with the query, the obtained result in terms of resulting row count, or numeric result (if there was a single-cell result to

the query), as well as a subjective indication of how easy it was to complete the query. There were 6 additional query tasks distributed to the students, as well, however, none of the students has considered these given the total time limit of the user study.

The total time allocated for the test was 70 minutes, plus there were 5 minutes for post-study questionnaire.

The task sheets and the questionnaires were collected from the students.

The participants performing textual SPARQL query writing were asked to submit the created queries by e-mail. The work of the participants doing the visual queries was checked by downloading and inspecting the projects created by the participants within the visual tool.

The logs of SPARQL queries (the ones that have been submitted to the SPARQL endpoint Virtuoso server) have been collected and have been used later to gather supplementary information about the query participant performance.

There were two users who have been assigned the SPARQL query writing did not follow the test conditions, explaining that they are not ready to write the queries in the SPARQL notation. These students attempted to do queries in the visual notation during the test time; so it the results of these students were not counted towards the test results.

Results

There 10 basic tasks given to the users were designed to correspond to different query patterns:

1. Class – attribute – condition
2. Class – attribute – link – conditions
3. Count + condition
4. Count + link + conditions
5. Statistics by attribute (including link and condition)
6. Subquery (condition on related object count)
7. Count over condition over subquery results
8. Existential link and sum aggregate function
9. Nested subqueries
10. Negated links

Table 1 summarizes provides the success of the participants on the queries:

- +: success,
- /: notable partial success (not counted in statistics),
- -: not solved,
- a: attempted query with no submitted solution (based on students indications on their task sheets).

The columns U indicate the user ID's, the columns 1-10 correspond to the tasks; the table left part shows SPARQL text writers and the right part – the visual notation users.

U	1	2	3	4	5	6	7	8	9	10	U	1	2	3	4	5	6	7	8	9	10
2	+	a	/	+							1	+	/	+	+	+	-	-	-		
3	+	/	+	+	+	/	/	a			4	+	+	+	+	+	+	+	/	-	
8	+	+									5	+	+	+	+	+	+	+	-		
9	+	-	+	-	a						6	+	+	+	+	+	+	+	a		

11	+		+	+	-	-	a				7	+	a	+	+	a				
12	+	+	+	+	/	a					10	+	+	+	+	+	a			
14	+										13	+	+	+	+	+	a			
17	+	+									15	+	+	+	+	/	+	+	+	+
19	+	+	+	+	a						16	+	+	+	+	+	a			
22	+	+	+	+	a	/					18	+	+	+	+	+	-	-	-	
23	-	+	+	a							20	+	+	+		-	+			
24	+	+	+	+	+	/	+	+			21	+	+	+	+	a	+	a		
25	+										26	+	+	+	+	/	+	+	+	a
											29	+	/	+	/	/	/			
											30	+	+	+	+	+	a		-		-

Table 1. Raw test results

Result interpretation

For every participant we count the number of fully completed tasks and then create the ranking groups depending on the fully completed task count.

Rank group	Mean rank	Completed tasks	Count	A count (SPARQL)	B count (ViziQuer)	A rank (13)	B rank (15)
1	1	9	1		1		1
2-5	3.5	7	4	1	3	3.5	10.5
6	6	6	1		1		6
7-12	9.5	5	6		6		57
13-18	15.5	4	6	4	2	62	31
19-20	19.5	3	2	1	1	19.5	19.5
21-26	23.5	2	6	5	1	117.5	23.5
27-28	27.5	1	2	2		55	
Rank sum:						257.5	148.5
Mean rank:						19.8	9.9

We use the following null hypothesis: for the selected set of queries and the selected user type the users will be able to complete at least as many queries in SPARQL, as they will be able to do in the visual ViziQuer notation.

Using standard statistical formulas for random rank assignment probability estimation [1], [2], we obtain that the test statistic (Z) for the null hypothesis is 3.1785 and the probability (p-value) is 0.0007 (7E-04) that is considerably below any commonly used threshold values (e.g. 0.05 for 95% confidence and 0.01 for 99% confidence).

$$Z = \frac{W_a - n_a(n_a + n_b + 1)/2}{\sqrt{n_a n_b (n_a + n_b + 1)/12}}$$

where:
W_a is the sum of the ranks for the first group
n_a is the sample size for the first group
n_b is the sample size for the second group
Z is the test statistic

[1] http://onlinestatbook.com/2/distribution_free_tests/rank_two.html
[2] http://onlinestatbook.com/2/calculators/normal_dist.html

For the calculation the following values are used: $W_a = 257.5$, $n_a = 13$, $n_b = 15$.

The following remarks provide further qualitative analysis of the test results.

The percentage of correct queries among the submitted ones for the users working in both notations did not show a significant difference, although it was slightly higher for the users working in the visual notation: the correct queries (among the submitted ones) were 76% (38 out of 50) for participants writing queries in SPARQL (24% error rate) and 79% (79 out of 100) for participants writing queries in the visual notation (21% error rate).

It would be possible to signal to the users about 4 out of 21 errors admitted within the visual environment by the test participants, where the query graph did not have a spanning tree consisting of non-condition links, by an appropriate error messaging mechanism (such a mechanism shall be included into the upcoming versions of the visual query environment).

The ratio of the successful queries to the attempted ones shows a slightly higher difference between the SPARQL and visual query environments: 66% for SPARQL query writers and 73% for visual query writers.

It may be instructive also to observe the ratio of the correctly completed queries with respect to the attempted ones on the per task basis.

Table 3 shows the “success rate” for each of the tasks in both SPARQL and ViziQuer notations (with the total number of attempting users in parenthesis).

	1	2	3	4	5	6	7	8	9	10
SPARQL	0.92(13)	0.7(10)	0.89(9)	0.78(9)	0.29(7)	0.00(5)	0.33(3)	0.50(2)		
visual	1.00(15)	0.8(15)	1.00(15)	0.86(14)	0.53(15)	0.50(14)	0.63(8)	0.25(8)	0.33(3)	0.50(2)

The table shows a slight edge of the visual notation over the textual one on queries 1-4 (class-attribute-link-condition queries and simple aggregate queries), a more considerable edge of the visual notation over the textual one on queries 5-7 (statistics by attribute and simple subqueries (the results of Task 7 in the SPARQL notation, although very low in the number of participants, can be considered together with the results of Task 6 due to the similarity of Task 7 to Task 6)), however, a concern for the visual notation is indicated by the query 8 and existential condition link therein that were handled correctly just by 2 of 8 test participants attempting the query. An explicit existential query pattern introduction into the query notation presentation can be an important part of the solution, design of a well-established user environment shall also account for the easy user options to create the existential queries.

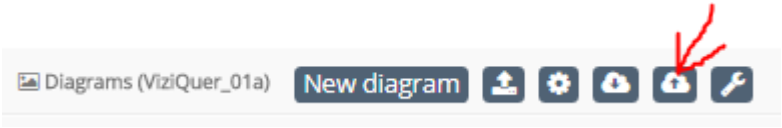
The first Although the problem can be at least partially remedied by introducing an explicit existential condition pattern during the notation presentation, the notation extension options (e.g. such as offering explicit {exists} stereotype possibility over edges) shall be considered along with the future user-friendly query environment development. The visual query environment development and user instruction creation shall target also the composition of queries of Task 5 (statistics by attribute) and Tasks 6 and 7 (simple subqueries) that have been relatively hard for direct query writing in SPARQL.

The results for queries 9 and 10, as well as for the query 8 in the textual SPARQL query writing in Table 3 are just illustrative due to the low number of participants attempting the concrete query.

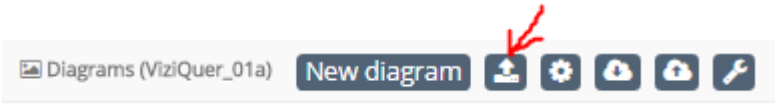
Query environment re-construction

To re-construct the query environment, as used in the user study:

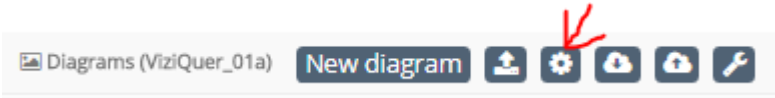
- Sign up to the ViziQuer server (follow instructions on viziquer.lumii.lv);
- Create two empty projects HospitalA and HospitalB;
- Import the diagrams from the HospitalA.json file into the HospitalA project and the diagrams from the HospitalB project (use the 'Upload project' button in the project toolbar):



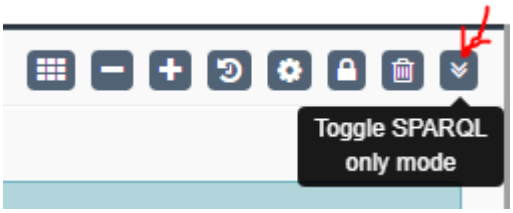
- Import the data ontology into both HospitalA and HospitalB projects (use the 'Import ontology' button from the project toolbar):



- Check that in the project settings pane the setting SPARQL Endpoint has the value http://192.168.100.12:8833/sparql and the Named Graph is set to MiniBkusEN_1 (the settings should be there after the project import; if they are not, they can be adjusted manually):



- To enter the SPARQL-only mode of a diagram, press 'Toggle SPARQL only mode' on the diagram toolbar:



(the diagram has to be opened beforehand).

It is recommended to work with SPARQL only mode over an empty diagram so as not to delete occasionally the contents of the diagram due to not fully polished tool behavior (e.g. keyboard shortcut effects).