ViziQuer
Structured Semantic Data Search Tool

Short user manual

Overview
The ViziQuer tool allows visual/diagrammatic creation of SPARQL queries over data corresponding to a given OWL ontology (the ontology has to be loaded into the tool/project).

The ontology can be loaded either from a file, or directly from a SPARQL endpoint. In the case of loading ontology from SPARQL endpoint the actual endpoint data structure is analyzed what may take time especially in the case of large and/or remote SPARQL endpoints.

The user creates visual queries from which the tool generates corresponding textual SPARQL queries that can be saved into file, or copied into the query window of a SPARQL endpoint browser.

Requirements
Windows XP SP2 or later

Installation and starting
Download the packed executable archive ViziQuer.zip, unzip it to a freely chosen folder on your computer (call it the ViziQuer root folder, it has subfolders Bin, Tools, Projects, etc., as well as executable tda.exe).

Start the executable tda.exe. The first time the executable is started, the user is asked to confirm registration of two components used by the tool.

Managing projects
A project consists of a set of visual query diagrams, typically based on a single data ontology.

From the main tool window ‘Project’ menu select ‘New Project’ to start a new project. Select ‘Open project’ to open an already existing project.

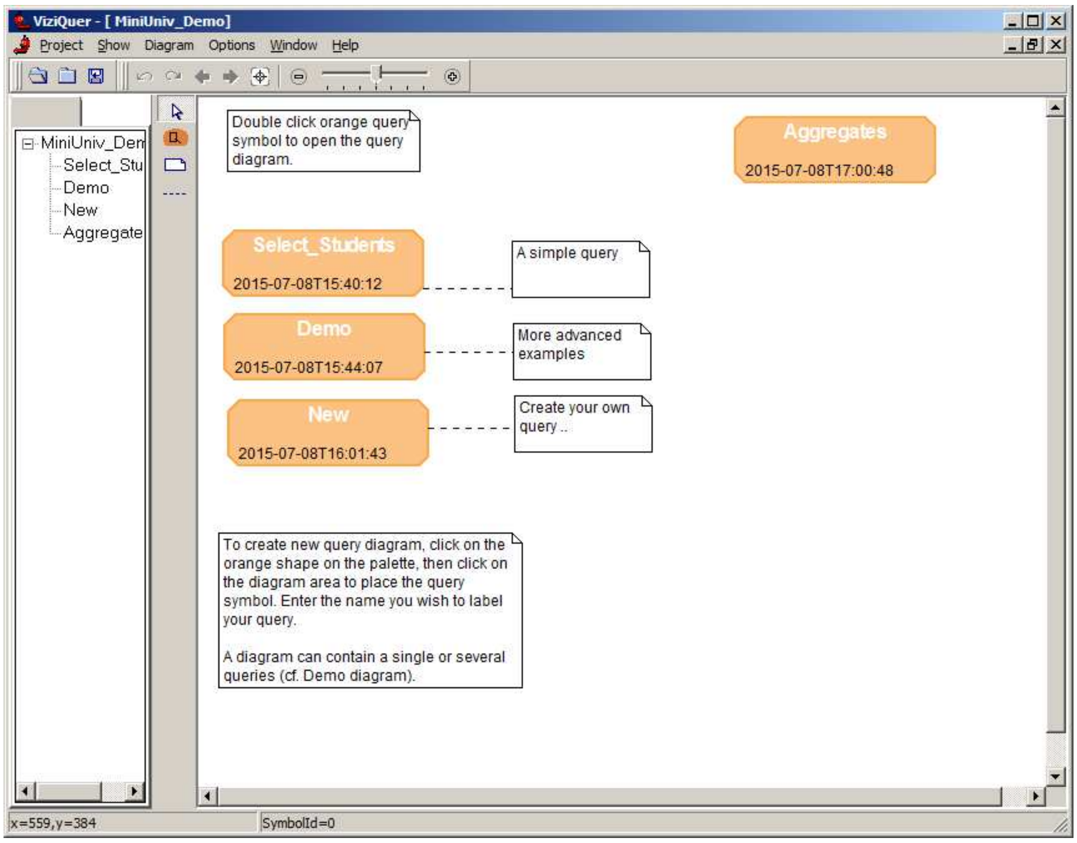
The default directory for projects is folder ‘Projects’ under the ViziQuer root folder. The user may start with trying some of demo projects, if they are pre-loaded into the ‘Projects’ folder.

When starting a new project, a dialogue appears to select the tool (choose ‘ViziQuer_nn’) and the workspace (it is recommended to choose the ‘Projects’ folder or some its subfolder), as well as give the project name (up to user’s choice).

For opening a new project there is a splash screen asking to load a data ontology either from a file, or from a SPARQL endpoint (the splash screen can be dismissed, if the user wants to load the ontology later).

Project diagram

The project diagram lists the query diagrams present in the project, together with optional comments. Figure 1 contains a project diagram example with four query diagrams.



Double click orange query symbol to open the query diagram.

A simple query

More advanced examples

Create your own query ..

To create new query diagram, click on the orange shape on the palette, then click on the diagram area to place the query symbol. Enter the name you wish to label your query.

A diagram can contain a single or several queries (cf. Demo diagram).

Fig. 1. A project diagram example

The diagram symbol palette besides the mouse pointer symbol contains symbols for new query diagram (orange shape), comment (white folded rectangle) and link between comment and query.

To create a new query, click on the orange shape in the palette, then click on the diagram to place the query symbol. Enter the name you wish to label your query, as well as other information you may want to set.

Click on the orange query symbol to open the query diagram.

The context menu for the diagram area (right mouse click on the diagram outside any symbol) offers the ‘Reload ontology’ command for changing the ontology behind the project, as well as ‘Update prefixes’ that is used for tuning the presentation of generated SPARQL queries.

Query diagram - overview

A query diagram corresponds to a single or multiple visual queries. Figure 2 shows an example of diagram containing a query with a single class *Student*, for which the student instance URI (denoted by placing its instance reference name *S* also in the query class attribute list) and its *studentName* property value are listed.

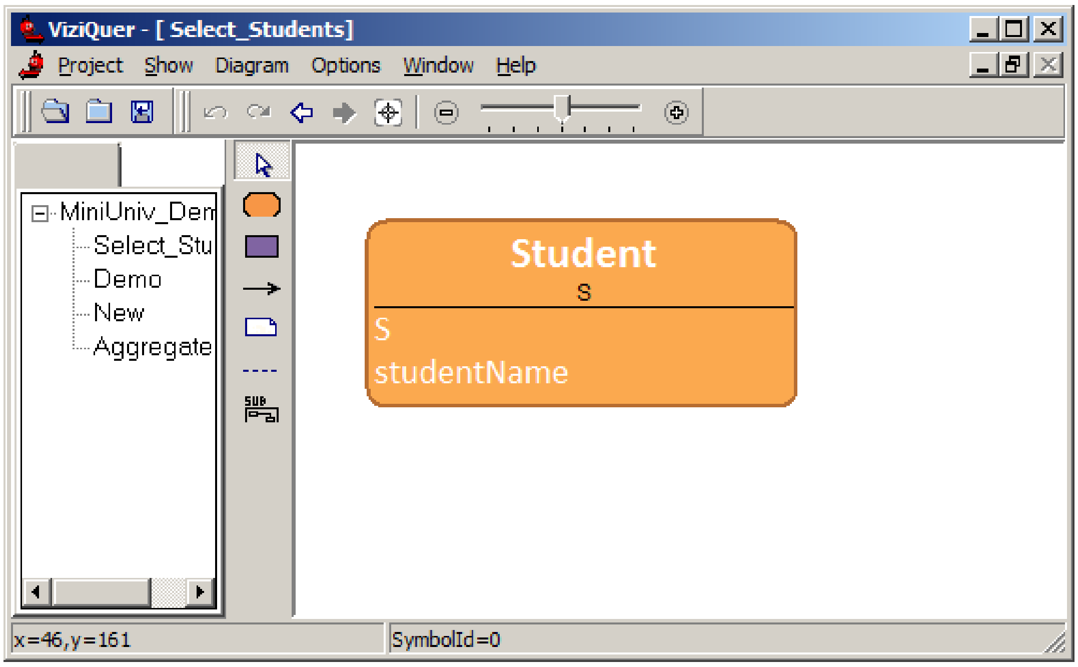
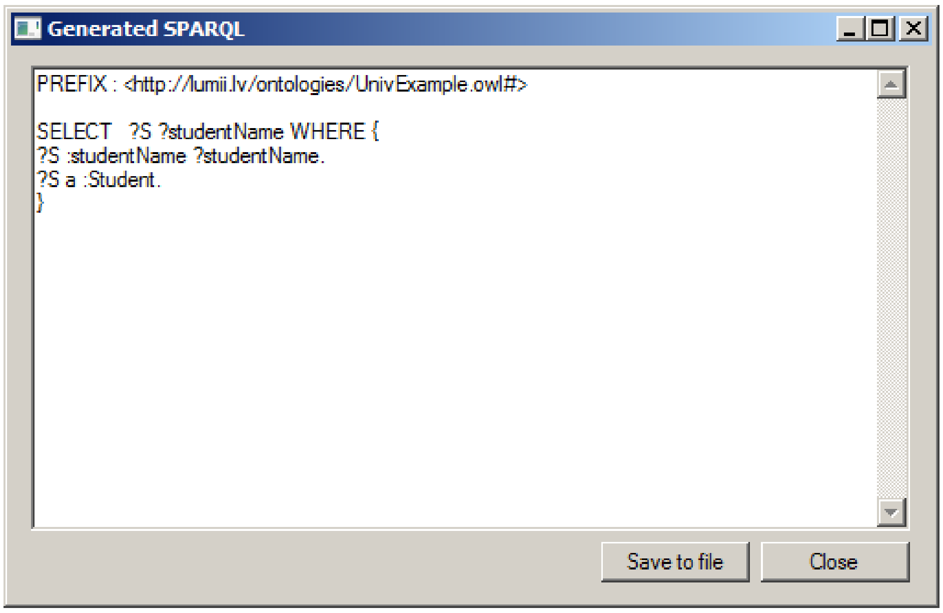


Fig. 2. A simple query diagram

The GenerateSPARQL command that is available within the diagram-level context menu creates the following result (the ontology URI is recorded during ontology loading, at the same time the prefix ‘:’ definition is created).



```
PREFIX : <http://lumii.lv/ontologies/UnivExample.owl#>

SELECT   ?S ?studentName WHERE {
?S :studentName ?studentName.
?S a :Student.
}
```

Fig. 3. A generated SPARQL window

Figure 4 shows a larger query diagram consisting of five different queries. Select a connected query fragment and choose ‘Generate SPARQL from selection’ from the context menu to translate the selected diagram fragment into SPARQL.

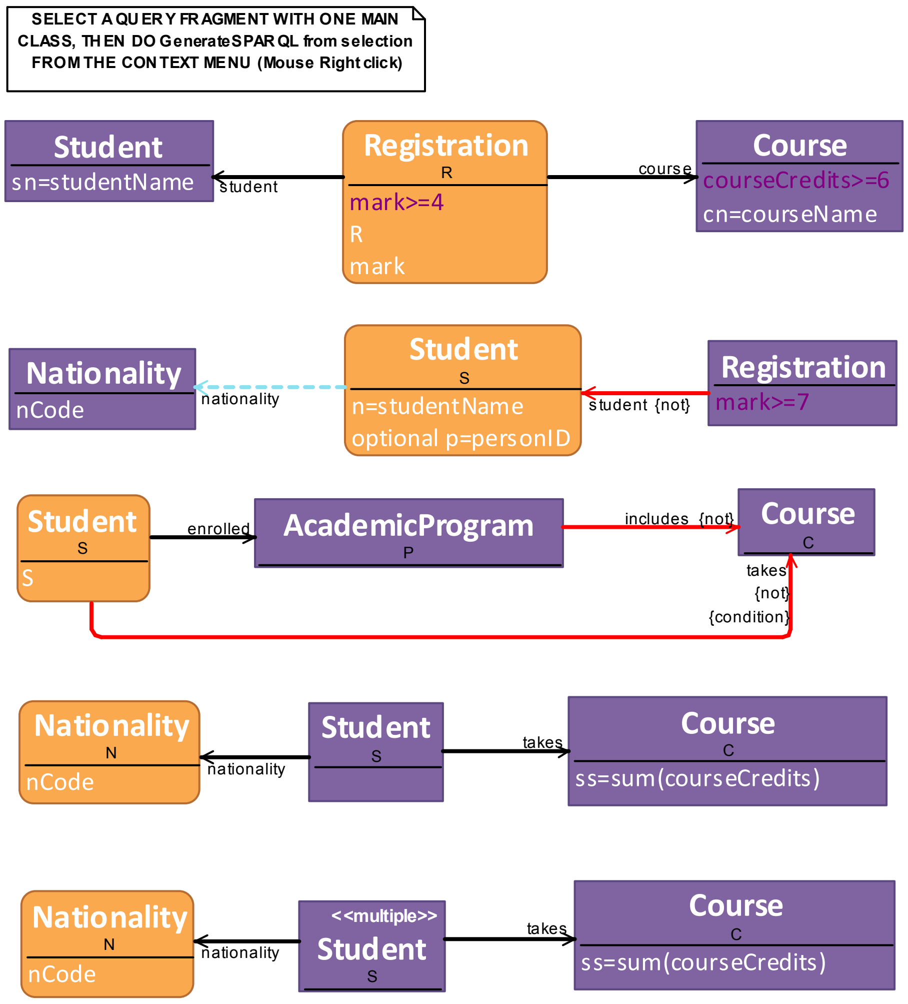


Fig. 4. A larger query diagram example

Figure 5 shows two ways of explicitly introducing subqueries into the queries: via inline grouping links, and via explicit subquery groups (named subqueries). The example query is: select all courses taken by at least 10 students whose mean marks (across all courses) is at least 7.

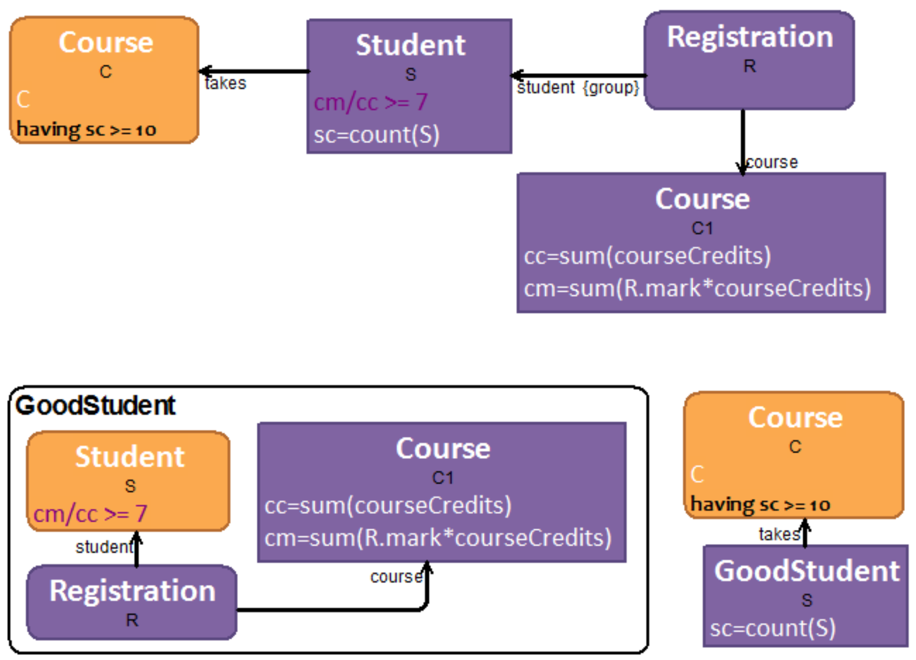


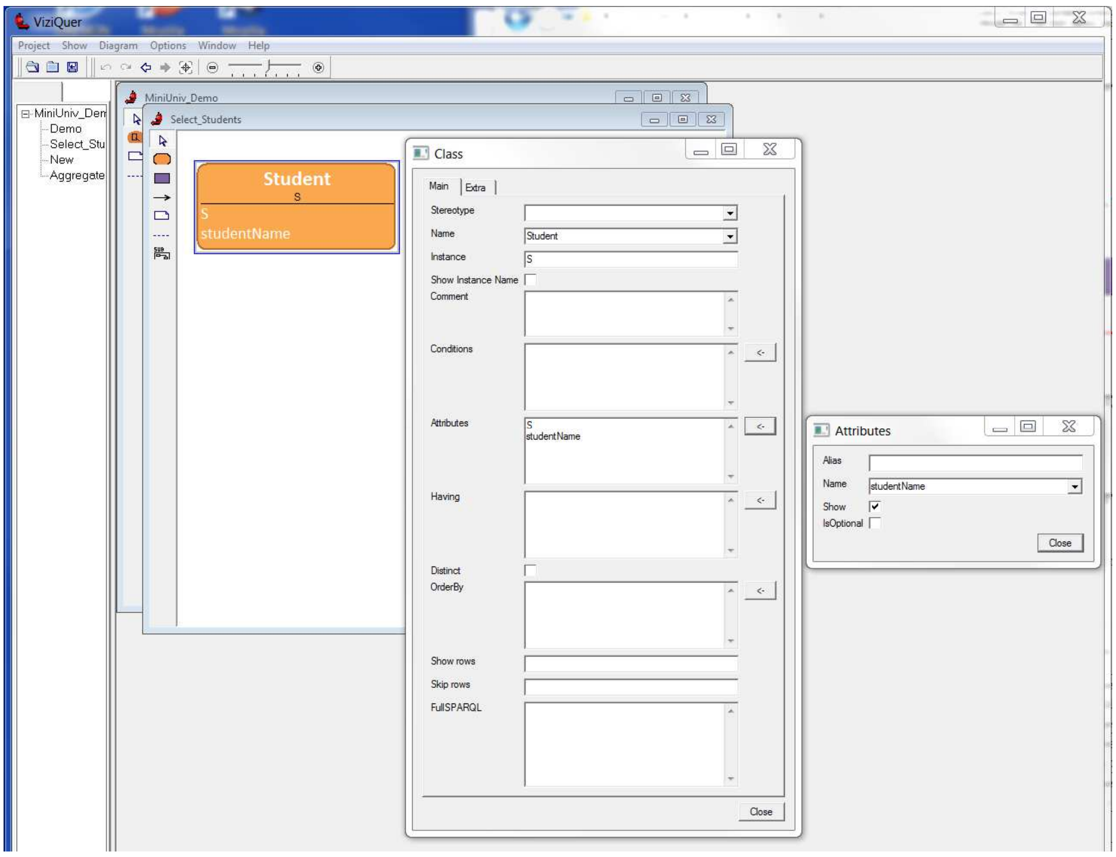
Fig. 5. A class and attribute property dialogue example

Working with query diagrams

There are the following items in the symbol palette of the query diagram:

🟧	Main query class. Every query (more precisely – every query diagram fragment for which the SPARQL query is generated) has to be a connected graph with exactly one main query class (not counting the explicit subqueries).
🟪	A condition class. There can be several condition classes within a query, linked either directly or via other condition classes to the main query class.
→	A link between classes, usually to be labelled with a property name for triples connecting link end class instances.
▭	A comment symbol.
----	A connector for connecting comments to class boxes in the query.
SUB	An explicit subquery group. Place the query and condition classes within the subquery area, name the subquery and then refer to it the same way as to a class in a query.

Figure 6 shows a property dialogue example for a class symbol, as well as its sub-dialogue for attribute information entry; a similar dialogue shape is available both for main query class and condition class property entry.



Stereotype	
Name	Student
Instance	S
Show Instance Name	☐
Comment	
Conditions	
Attributes	S studentName
Having	
Distinct	☐
OrderBy	
Show rows	
Skip rows	
FullSPARQL	

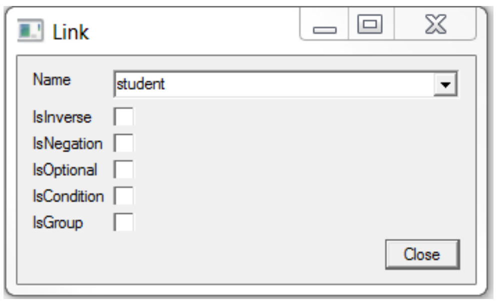
Alias	
Name	studentName
Show	✓
IsOptional	☐

Fig. 6. A class and attribute property dialogue example

Fill in the class name (choose from the drop-down list, filled by the classes in the ontology loaded into the project), optionally edit the class instance name (to be used for variable generation in SPARQL query). Place the filtering expression in the *Conditions* field and selection attributes (or attribute expressions) in the *Attributes* field. A line in the *Attributes* field corresponds to an attribute or attribute expression to be computed and included into the query select list (if the check-box *Show* is not un-selected). There can be simple (i.e. non-aggregate) expressions specified, or aggregate expressions where a SPARQL aggregate function is applied to a simple expression. The filters in the Conditions field are applied before aggregate expression computation (i.e. the aggregation is performed over the filtered selection list).

The Having, Distinct, Order By, Show rows and Skip rows fields are effective for the main query class only. The Having field allows to add conditions on the computed attributes; these conditions are calculated after the attribute expression (including the aggregated attribute computation).

The FullSPARQL field is experimental; it is possible to add SPARQL graph fragments referring to the variables, as they appear in the generated SPARQL query.



Name	student
IsInverse	☐
IsNegation	☐
IsOptional	☐
IsCondition	☐
IsGroup	☐

Fig. 7. A link property dialogue example

Figure 7 shows an example of the link symbol property dialogue. The link name can be chosen from the dropdown list filled from the ontology in the context of link source and target classes, or it can be entered in a textual form. The check-boxes have the following meaning.

isInverse	The query is labelled with the inverse of the property specified in the name field, as well as the subject and object in the property triples in the generated SPARQL query are included in reversed way.
isNegation	The negated link. If isCondition is not checked, everything behind the link (from the main query class viewpoint) become placed into a negation group. If isCondition is checked, the link is translated into the filter asserting non-existence of the corresponding triple between the link end instances.
isOptional	Optional link. If isCondition is not checked, everything behind the link (from the main query class viewpoint) become placed into an optional group. If isCondition is checked, as well, the link semantics becomes void.
isCondition	Marks the link as condition link. There is requirement for the entire query graph (either the whole query diagram, or its part, from which the SPARQL code is generated) to have its non-condition links form a tree covering all graph nodes (except for explicit subqueries). The condition links (either affirmative, or negated) are not included in the tree shape structure.
isGroup	A link, creating inline a subquery group from all query part placed behind it in the query graph (the link itself also is added to the subquery).