



GSoC Midterm Evaluation v. 2

Adding support for Neo4j in Gephi

Student:
Martin Škurla

Mentors:
Tobias Ivarsson, Neo4j project
Mathieu Bastian, Gephi project

Introduction

My name is Martin Škurla and this summer I am working on GSoC project called “Adding support for Neo4j in Gephi”. In this article we will look at implemented features including these under the hood, pictures of dialogs, common use cases and future plans.

Gephi project

At first I want to make quick introduction into Gephi project. Gephi is Open Source Visualization Platform build on top of the NetBeans platform. It is written in Java so you can run it on various Operating Systems including Windows, Linux and Mac OS. It supports many interesting graph analysis capabilities including [2]:

- Real-time visualization
- Layout
- Metrics
- Dynamic network analysis
- Cartography clustering and hierarchical graphs
- Dynamic filtering

The story so far

The main idea of my project was to add support for Neo4j in Gephi. This means the ability to transform the Neo4j graph into Gephi graph. In fact, both graph models are different so the first task was to make mapping between Neo4j graph items and Gephi graph items and vice versa.

There was also a mismatch between types supported in Neo4j and these supported by Gephi. This mismatch was solved by adding new “List” types into Gephi, so now every type in Neo4j has its appropriate type in Gephi.

There were also some changes under the hood which are not visible to end user, but must be defined and implemented. The most interesting thing is adding “Delegating mechanism”. This mechanism is responsible for getting values from storing engine (Neo4j) as well as manipulation with data. In fact during the importing process, graph representation of Neo4j graph is created in Gephi, but all values are not stored directly in memory, but they are queried using the delegating mechanism.

Another minor tasks were to customize the open dialogs used for importing local Neo4j database and debugging the imported database. The open dialog for importing accepts only valid Neo4j database directories. I defined valid Neo4j database directory structure and every valid directory now includes picture of Neo4j in the open dialog. User is able to open only valid Neo4j directories in the process of importing. The open dialog for debugging accepts only Java class files that can be used for debugging process. This simply means they have to implement required interface and have public nonparam constructor. Every valid class file will have Neo4j picture and after selecting a valid debug file, Target and Visualization options will be automatically filled based on data from selected class file.

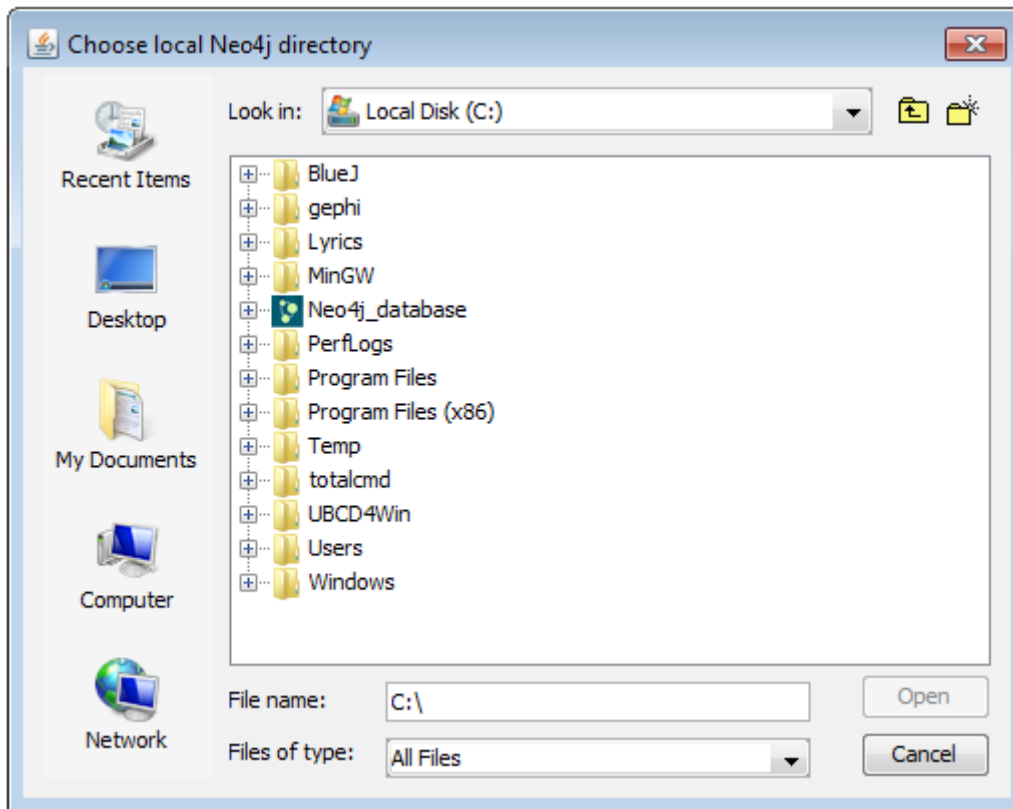


Figure 1. Open Neo4j directory dialog customization

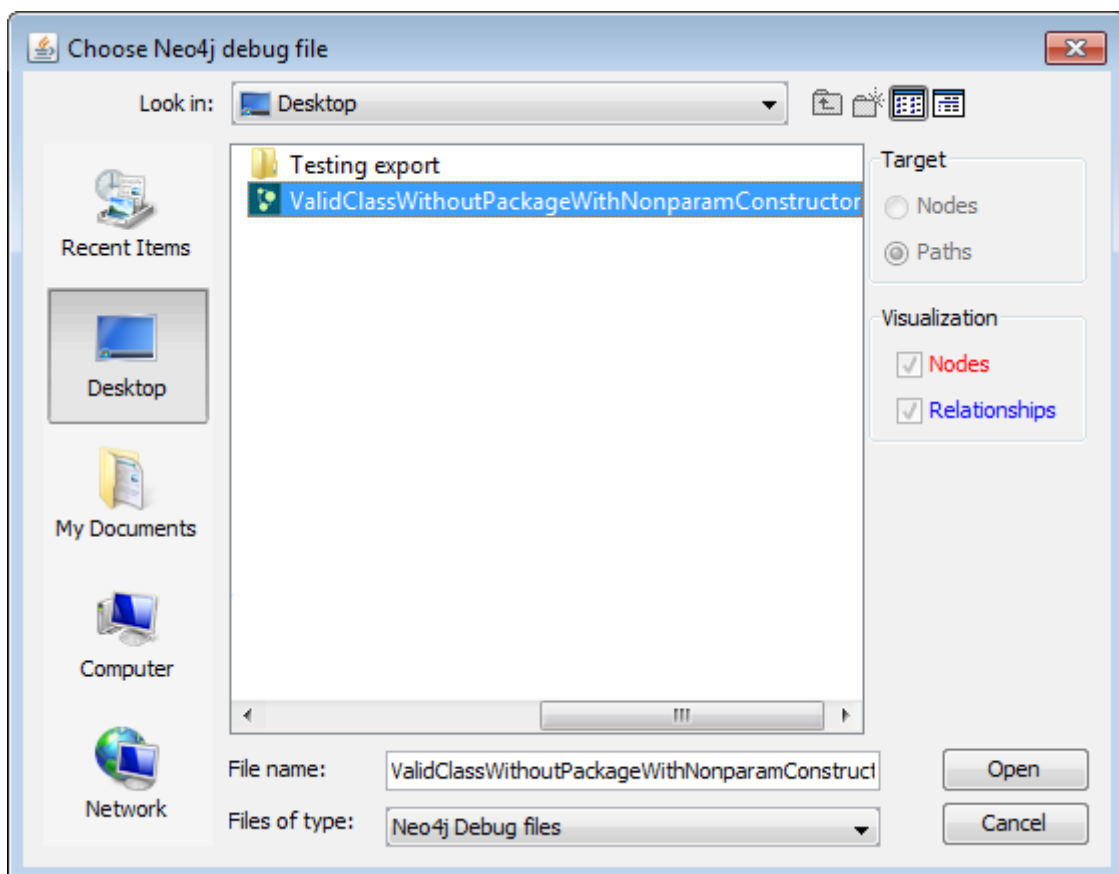


Figure 2. Open Neo4j debug file dialog customization

Neo4j integration

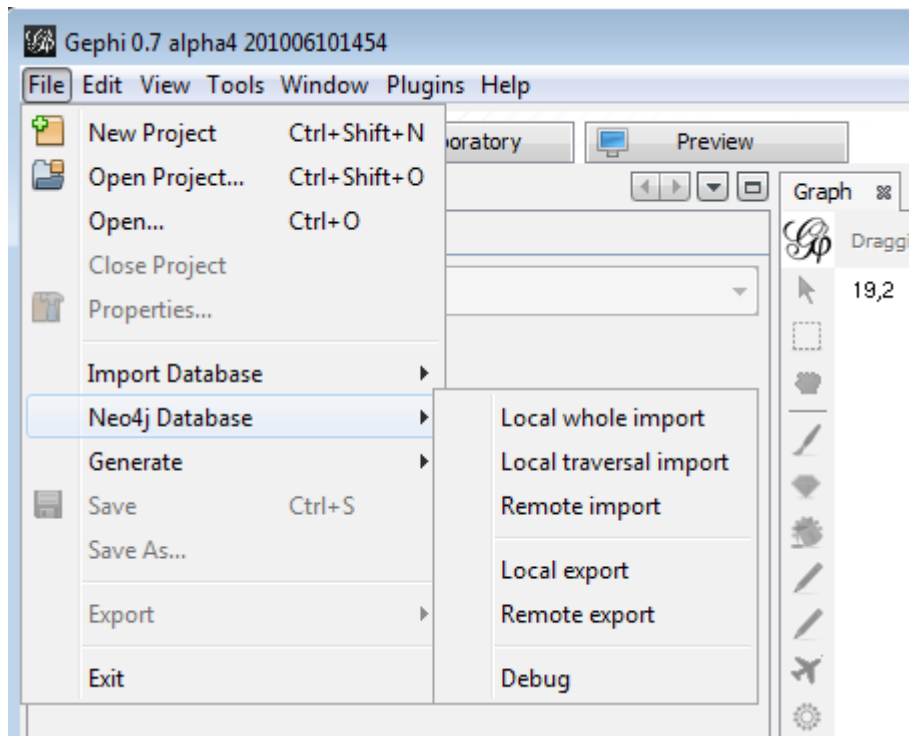


Figure 3. Menu integration

All possible actions started in menu. As we can see, this is the entry point to import from, export to and debug the Neo4j graph. Both importing and exporting support local as well as remote Neo4j databases.

Importing

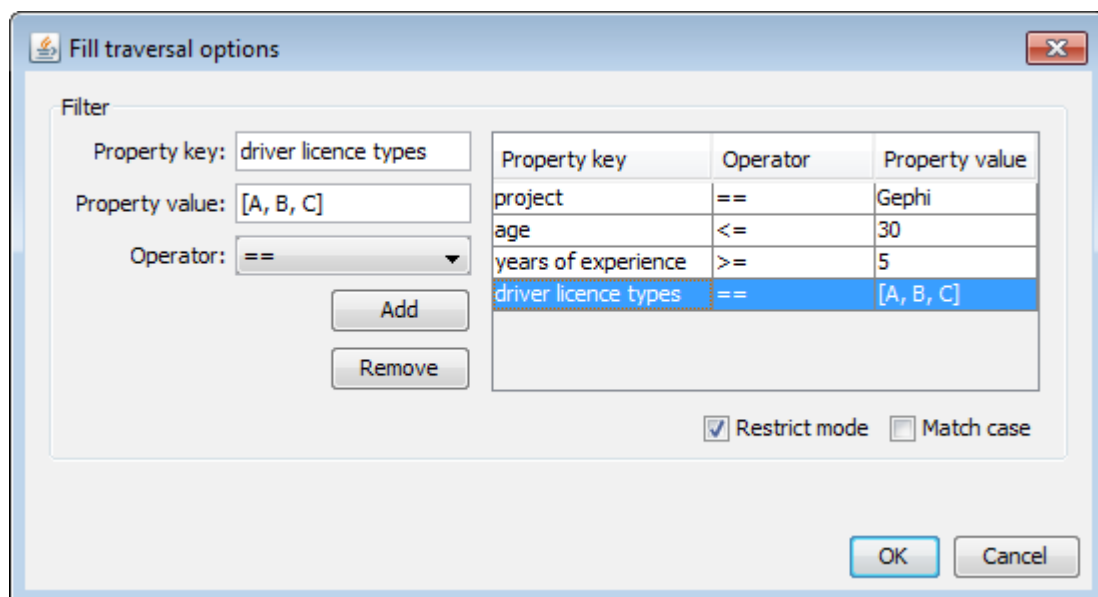


Figure 4. Whole graph import dialog

Importing process consist of 2 approaches:

- whole import
- traversal import

Whole graph import dialog is designed for importing whole graph. We can customize the rules responsible for returning nodes by defining filtering expressions. For example previous dialog can be used when we want to find all people working on project Gephi with maximum age 30 years. Only people with at least 5 years of experience and those which have driver licence types A, B and C will be included.

Let's have a deeper look at the dialog:

- **Property key** is the name of property we want to filter
- **Property value** is the value which will be compared to actual Node property value using chosen operator. Values will be automatically converted into appropriate types and if the value cannot be converted, the node will not be included into graph. All types supported in Neo4j are supported in this dialog. We can also see the support for array types in the last filter expression.
- **Operator** will be applied on the final expression and if the expression is evaluated to true, node will be included
- **Match case** means the ability to compare String, char, String[] and char[] types with respect of the same case
- **Restrict mode** is used to restrict some nodes. Imagine we have people stored in database which have only subset of required property names used in filtering expressions. If the Restrict mode is on, only nodes which have all property names and all filtering expressions evaluated to true will be included. If the Restrict mode is off, every node which has any subset of required property names (even empty subset) will be included if all the filtering expressions applicable to the subset will be evaluated to true.

All the filtering expressions are combined together using AND and the list of current supported operators consist of:

- ==
- !=
- <
- <=
- >
- >=

In fact, usefulness of adding new operators as well as including OR and other useful import options is the main idea behind Questionnaire which is part of this article.

Traversal graph import dialog is designed for importing any subgraph using traversal capabilities of Neo4j v 1.1. Traversal import adds additional options:

- **Start node** can be set in two ways, either by its id or by its indexing key and value pair
- **Order** can be set to depth or breadth first algorithms
- **Max depth** can be set to concrete number or to end of graph
- Relationships can be restricted too. We can set any combination of Relationship types and directions which should traversal include. The list of Relationship types is dynamically filled from database with existing values.

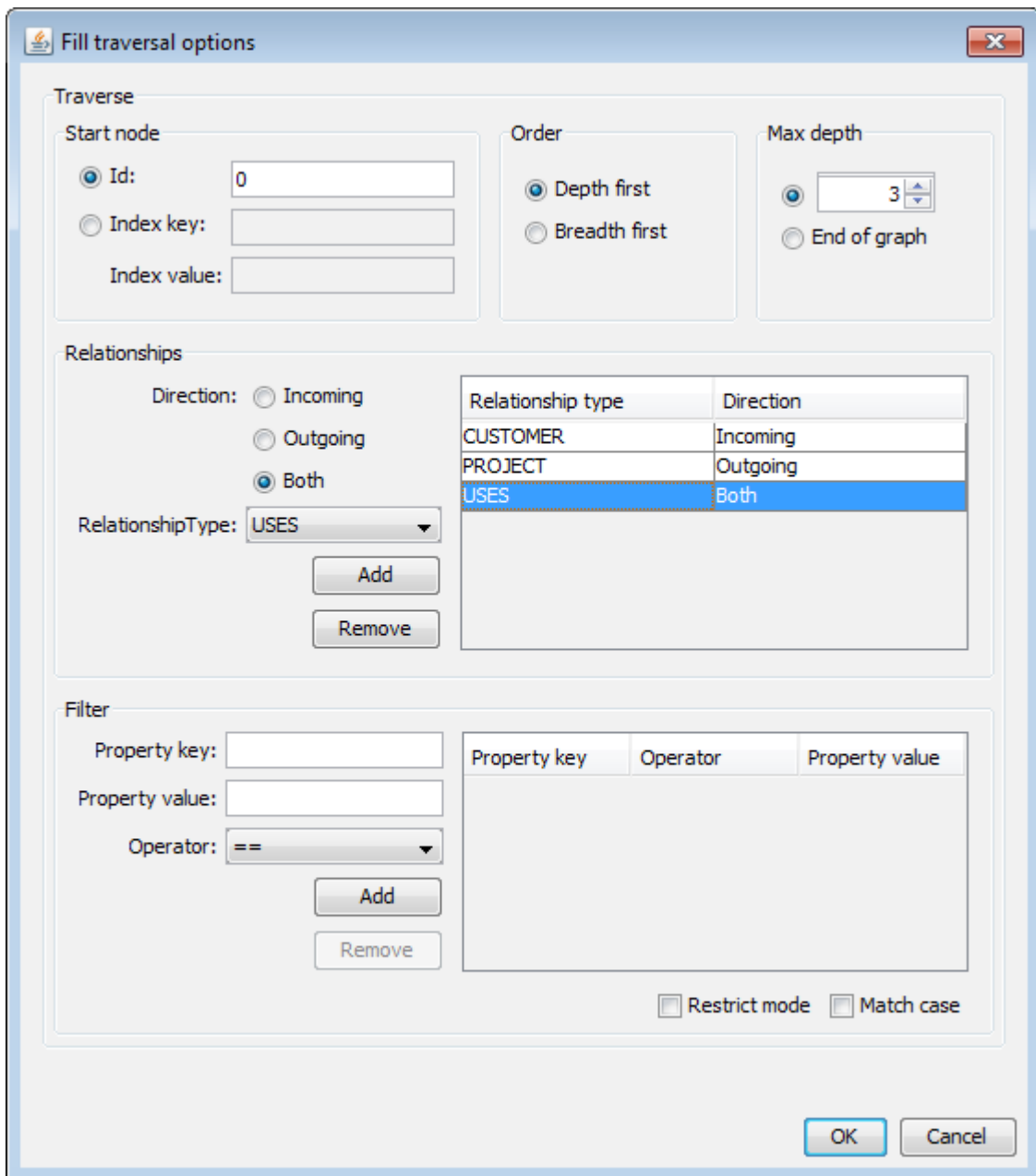


Figure 5. Traversal graph import dialog

This was the quick summary of Gephi Neo4j importing capabilities implemented in the project. We focused on more features and one of them is the support for exporting. We can export any loaded graph into local or remote Neo4j database. The exporting process can be customized in similar way as importing.

Exporting

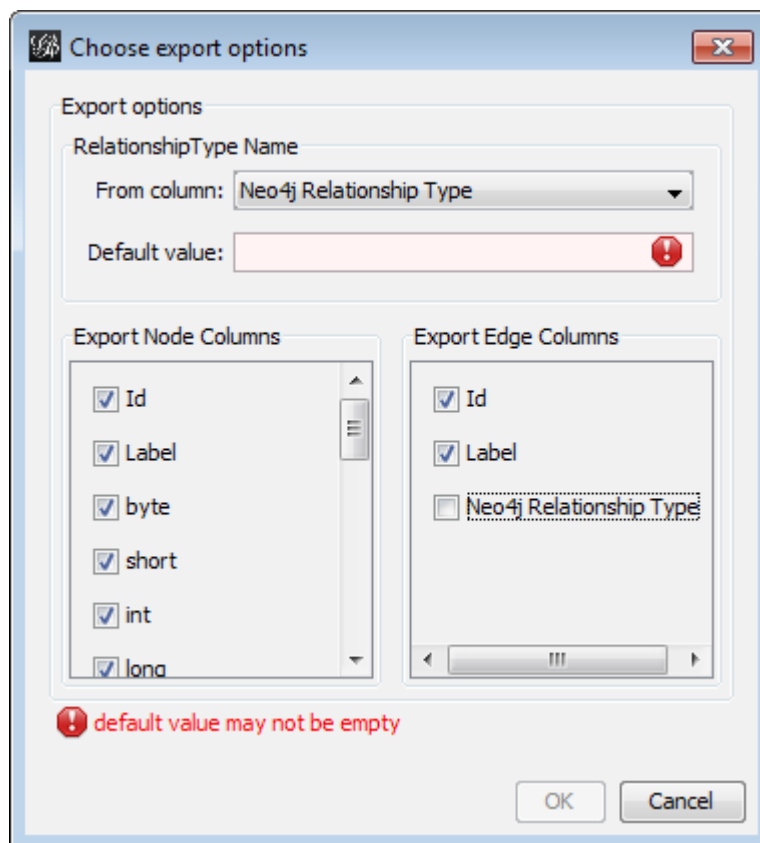


Figure 6. Export dialog

Exporting means opposite process to importing. Previous dialog shows exporting options as well as validation. We can customize exporting process by setting:

- **From column** is used to set the RelationshipType to appropriate values from any of Gephi edge columns. During importing Neo4j graph, column with name “Neo4j Relationship Type” is automatically created.
- **Default value** is used in the case when processed Gephi edge does not have value in selected From column
- **Export Node columns** is the set of Gephi columns in node table which will be exported
- **Export Edge columns** is the set of Gephi columns in edge table which will be exported

Remote importing/exporting

The only difference between local and remote importing/exporting is the existence of Remote dialog, where we need to set following connection information:

- **Remote database URL**
- **Login**
- **Password**

All of these options must be filled in order to successfully import/export remote graph.

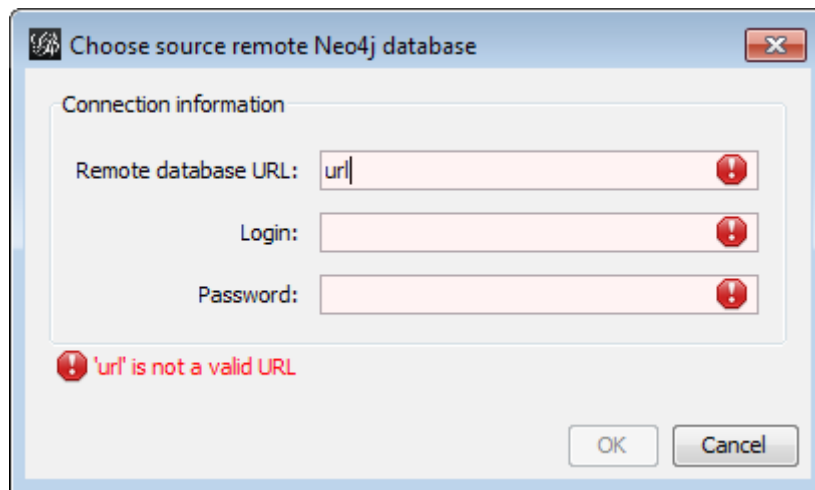


Figure 7. Remoter import/export dialog

Delegation process

Nodes	Label	byte	short	int	long	float	double	boolean	char	string	array byte	array short	array int	array long	array float	array double	array char	array boolean	array string	name
0	0	1	2	3	4	5	6	<input checked="" type="checkbox"/>	A	LALALA	1,2,3	1,2,3	1,2,3	1,2,3	1.0,2.0,3.0	1.0,2.0,3.0	a,b,c	true,false,true	a,b,c	middle project node
1	1							<input type="checkbox"/>												middle customer node
10	10							<input type="checkbox"/>												Apple
11	11							<input type="checkbox"/>												AIRPLANES
12	12							<input type="checkbox"/>												SAP
13	13							<input type="checkbox"/>												AIRPLANES
14	14							<input type="checkbox"/>												IBM
15	15							<input type="checkbox"/>												BUSES
16	16							<input type="checkbox"/>												Orade
17	17							<input type="checkbox"/>												CARS
18	18							<input type="checkbox"/>												Microsoft
19	19							<input type="checkbox"/>												Inkscape
2	2							<input type="checkbox"/>												AIRPLANES
20	20							<input type="checkbox"/>												Google
21	21							<input type="checkbox"/>												CARS
22	22							<input type="checkbox"/>												OpenJDK
3	3							<input type="checkbox"/>												Gephi
4	4							<input type="checkbox"/>												Chromium
5	5							<input type="checkbox"/>												Mozilla
6	6							<input type="checkbox"/>												Gnome
7	7							<input type="checkbox"/>												KDE
8	8							<input type="checkbox"/>												GCC
9	9							<input type="checkbox"/>												

Figure 8. Nodes values exploration

As we can see from previous picture, we can very simply explore all the node and edge values. This is exactly the place where delegating mechanism is used. All values are in fact not stored directly in memory in some kind of Gephi data structure, but the storing engine (Neo4j) is requested for actual values every time we need them.

Debugging

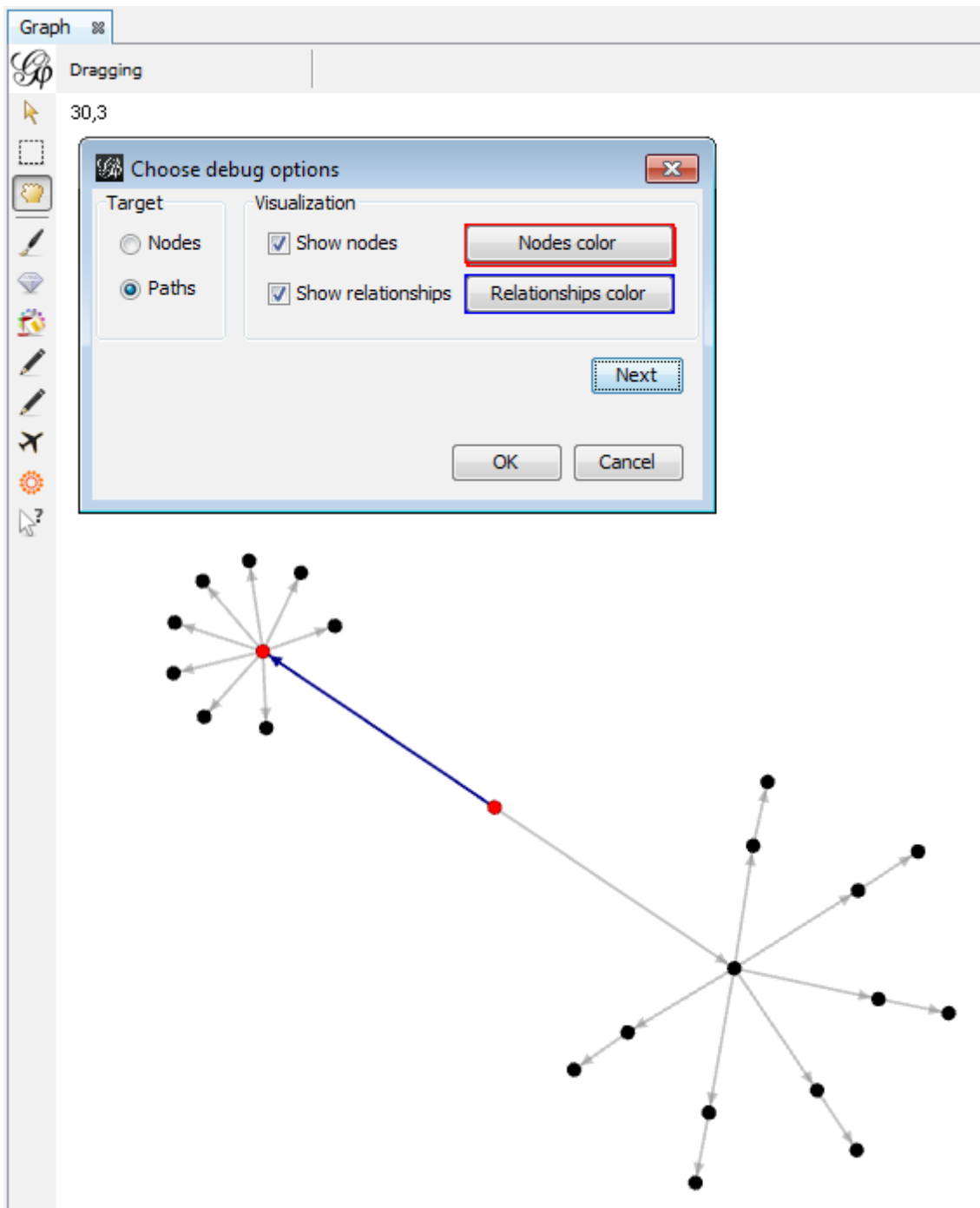


Figure 9. Debugging in action

We can see debugging in action in previous picture. The dialog is initialized with data from chosen debug class file, but we can change all of them at the runtime too. Any change in options will automatically update graph visualization. We can change visibility of nodes and edges as well as colors for both nodes and edges. User proceeds to next step of debugging/traversal by clicking on the Next button.

Use cases

That was the quick summary of all implemented features and now we can summarize common use cases every user can be interested in.

Visualizing Neo4j graphs

One of the main ideas of my project was to implement the ability to visualize Neo4j graphs, even big ones. As we saw from the dialog pictures, we have many options how to customize the importing process including filtering. After the import we can use all the rich graph analysis features Gephi provides.

Analyzing only part of the whole graphs

Quite common use case is to analyze only part of the graph, which is possible in Gephi too. We can take advantage of traversing where we can set starting node and other traversal options. After that we can visualize and analyze only part of the graph.

Export graph stored in text files/databases into Neo4j

Another use case could be exporting graphs stored in graph text files or relational databases into Neo4j. In fact, every graph loaded into Gephi can be easily exported to Neo4j database. Importing formats depends on Gephi abilities themselves, currently following formats are supported:

- text formats** [1]: GEXF, GDF, GML, GraphML, Pajek NET, GraphViz DOT, CSV, UCINET DL, Tulip TPL, XGMML
- relational databases**: MySQL, PostgreSQL, SQL Server

Future plans

There are more things which we want to implement, including:

- support for Gephi Toolkit, which is in general set of Gephi core libraries which you can use in your own Java projects for graph visualization and manipulation
- implementing proof of concept Web application using both Gephi Toolkit & Neo4j to manipulate with Neo4j database & show results (probably using GWT)
- more features, bug fixing, performance optimizations

Questionnaire

One of the big advantages of Gephi is the fact that it is developed as Open Source project. We want to add additional features according to user requests and their opinions. That's why we created questionnaire focusing on usefulness of proposed additions. We will be very happy if you fill the questionnaire because it is very valuable source of information and we can focus on features Neo4j users think useful. The questionnaire can be found [HERE](#).

Conclusion

I am very happy that I can be part of the Gephi developer community and introduce integration with Neo4j. During this summer I learned a lot and I am proud that I was chosen as GSoC student. The fact is that none of these features can be done without great help of my mentors, so big thank to both of them: Mathieu Bastian & Tobias Ivarsson.

If you are interested in and want to test the code, you can download source codes from my branch using:

```
bzr branch lp:~bujacik/gephi/support-for-neo4j
```

All the pictures were made on data stored in testing Neo4j database can be created using Java SE project and you can download it using:

```
bzr branch lp:~bujacik/+junk/testing-new-neo4j-traversal-api
```

Links

[1] Gephi supported graph formats <http://gephi.org/users/supported-graph-formats/>

[2] Gephi features <http://gephi.org/features/>