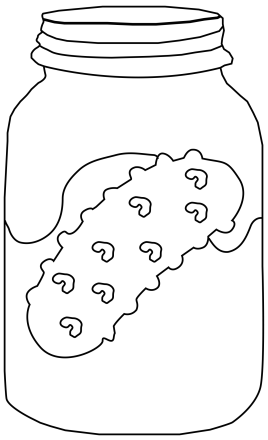
Technische Universität Braunschweig

ifis Institut für Informationssysteme Technische Universität Braunschweig

Algorithm Framework for Neo4j

SeaCucumber-Framework



Bachelor - Teamprojekt
Wintersemester 2017-2018

User instructions

Betreuer: Jan-Christoph Kalo, Stephan Mennicke

Name	E-Mail-Adresse
Christian Goldapp	c.goldapp@tu-braunschweig.de
Fabian Kirchner	fabian.kirchner@tu-bs.de
Maximilian Jahn	maximilian.jahn@tu-bs.de
Felix Scholz	felix.scholz@tu-bs.de
Maximilian von Unwerth	maximilian.von-unwerth@tu-bs.de
Tim Witschel	tim.witschel@tu-bs.de

Braunschweig, 19. Februar 2018

1 Instructions

Dear user,
we are delighted that you use our Framework for Neo4j. For everything to run accordingly, you need Neo4j, Maven and Java for our functions and it runs on every normal operating system. If you have any questions, please do not hesitate to contact us on our „Git Repository“.

1.1 Required-Software

In order to use our Framework you need:

- The Neo4J Client[1] is very important; it has to be installed otherwise you can't run the framework.
- The latest version of Java[2] has to be installed and the needed root-settings have to be set.
- Also your device needs Maven[3], this is needed to build the project and import alle dependencies.

For a better programming experience we recommend to use an IDE. During the instructive steps we use IntelliJ IDEA[4]. If you really need this instruction, you should download IntelliJ IDEA and you can install the Framework step by step (See point 1.3 Step by Step Manual).

[1] https://neo4j.com/download/other-releases/#releases
[2] https://java.com/de/download/
[3] https://maven.apache.org/download.cgi
[4] https://www.jetbrains.com/idea/download

1.2 First use guidance for experienced users

- The required Software should be installed on your computer so you can ensure that everything works right.
- Please start the Neo4j - Software and create a new database.
- The plugin folder has to be created in the following directory(neo4j/default.graphdb/plugins). You need this for your „Neo4j - Procedures“.
- Now you need our Framework. You can clone our „Git Repository[5] “. If you don't now how to clone the „Git Repository“ please read the Step by Step Manual.
- Furthermore you need to open the project with Maven. You can check your individual IDE for further instructions.
- In the package „matcher“ you can create a new Java class where you implement your algorithm.
- Things that need to be changed in this class, so that „Neo4j - procedures“ can work properly, will be explained in chapter 1.4 „Start coding“.

1.3 Step by step guide for beginners

This step by step guide is created for using the software „ItelliJ IDEA“.

- The requested Software should be installed on your computer.
- Please start the Neo4j - Software and create a new database. Neo4j will guide you through the creation process. One important thing is to know where the databse is stored. (By default the folder is called neo4j/default.graphdb).
  (Nice to know for login: by default the username and password is neo4j)
- The plugin folder has to be created in the following directory(neo4j/default.graphdb/plugins). You need this for your „Neo4j - Procedures“.

[5] `https://github.com/vonunwerth/Seacucumber.git`

- After downloading and installing the Software „IntelliJ IDEA“ opens and is ready for use. The following window opens.

- You click on „Check out from Version Control - GitHub“ and a window with the name „Clone Repository“ will open.
  Git Repository URL: https://github.com/vonunwerth/Seacucumber.git
  Parent Directory: The location where you want to save the project.
  Directory Name: For example Neo4j (You can choose whatever you want.)

  Click on Clone and IntelliJ will start to download the repository.

- You get a at the right corner a notification to add the project as a Maven-project. If the notification doesn't appear look up the trouble shoot section 1.7.

- All folders should be visible now. Open the src/main/java folder.

- In the „package matcher“ you can create a new Java file to create your algorithm.

1.4 Start coding

After checking out the repository you can start coding. Ensure you use Java JDK1.8 or higher to compile the project. Otherwise some features of the given classes cannot be used.

- Create a new class for your new Matching-Algorithm in the matcher package and let your new class extends the abstract class *matcher*.

- Implement the *matchingAlgorithm()-method* and import *org.neo4j.graphdb* in order to use the Nodes of Neo4J. Also import the java.util.List instead of the suggested scale list. Last import java.util.Map to get a result map of your keys and lists of nodes in the end.

```
@Override
public Map<Integer, List<Node>> matchingAlgorithm() {
    //Your own matcher
}
```

- You have to write a constructor in your class. The constructor's name has to be the same as the classname. The following structure can be used:

```
public [AlgorithmsName] (org.neo4j.graphdb.GraphDatabaseService db,
    graph.Graph graph) {
    this.db = db; //Describes your database
    this.graph = graph; //Describes your graph
}
```

  A lot of prewritten methods can be found in the abstract „Matcher“ class. Check our javadoc for more information.

- Now you have to create a now procedure to access your matcher on your Neo4J database. Go to the *procedure.GraphProcedures* class and e.g. copy one of the example procedures for Dual Simulation or Failures.

```
@Procedure(value = "graph.[NAME]", mode = Mode.READ)
@Description("[DESCRIPTION]")
@SuppressWarnings("unused")
public Stream<NodeResult> [NAME](@Name("query") String  query) {
Graph graph = prepareQuery(db, query);
[MATCHER] matcher = new [MATCHER](db, graph);
Set<Node> simulated = matcher.simulate();
return simulated.stream().map(NodeResult::new);
}
```

  Replace [NAME] with the name of your new procedure and [MATCHER] with the name of your new matcher class.

1.5 After coding

After you write your algorithm in your new class in the Java package matcher, you have to create the „jar“ for the database.

- Before you create the jar, you can test the code. Please use the „ProcedureTest“ class in the test package. For testing start the main method in the class „ProcedureTest.java“.

- Open the Maven tab in „ItelliJ “ and open the point „Neo4JMatchAlgFramework“. The next folder you need is the „Lifecycle“ folder, here you click „clean“ then you click on „package“.

- After finishing you start your explorer and search for the folder where you cloned the „Git Repository“ to. Here is a new folder named target. Open this folder and copy the „original-Neo4JMatchAlgFramework-1.0.jar“.
  (The other one is for testing and is not important. It won't work.)
- Please go to your Neo4j database from the first steps (see step by step manual).You must paste the „original-Neo4JMatchAlgFramework-1.0.jar“ to the previously created „plugins“ Folder.
- After doing this you can call your procedure in Neo4j. If you are ready, open the database with the Neo4j software.

1.6 Work with Neo4j

To test if your procedure works in Neo4j you can use the following example statements.

- You want to use your procedures? Then use the CALL-Statement.
  For example:

```
CALL graph.exampleQuery(
"MATCH (m:Movie)<-[:DIRECTED]-(p:Person)
RETURN m,p")
```

- You want to take your procedures and would like to search with another query on this result? Then take the CALL statement and the YIELD statement.
  For example:

```
CALL graph.exampleQuery(
"MATCH (m:Movie)<-[:DIRECTED]-(p:Person)
RETURN m,p")
YIELD node MATCH (m:Movie)<-[:DIRECTED]-(p:Person)
RETURN m,p
```

At this point you know everything we know - So have fun and develop new good algorithms.

1.7 Trouble shooting

- The Folder with our „Git Repository“ is displayed at the „IntelliJ“ window, but you only see „readme“and other data but not the Java files.
  If you want see the Java files, open the Maven Tab on the right side. Click on the tab and on the circle on the left corner. You will be ask, if you want to „import Maven Projects“ - click „yes“. The „Maven Dependencies“ are now downloaded, this needs a while.