UV Distributed Information Management
Summer term 2021

Assignment 03

Summary:

Deadline: June 14, 2021, 11:55 pm (aka 23:55) CET.

Extended Deadline: June 21, 2021, 11:55 pm (aka 23:55) CET.

Submission: A compressed archive (e.g., a `zip`-file) of your Python code and the answers to the questionnaire.

Grading: 55% Python code, 45% answers (incl. meeting).

---

1 General Remarks

In the course of this assignment, we will learn how to interact with *Neo4j*, a NoSQL database system that is based on the graph data model. Just like PostgreSQL and MongoDB, Neo4j is open-source and rather easy to install. Since the graph data model is very intuitive, we will also make use of the graphical user interface of Neo4j to interact with the database. However, also Neo4j provides an application programming interface (API) for the Python programming language (i.e., through the `neo4j-driver` [1] module). Similar to the previous assignments, commands (for command-line tools) and Python code are written in TrueType font, e.g., `MATCH (p:Person) RETURN p`.

Please submit your final Python code and the answers to the questionnaire until June 14, 2021, 11:55 pm (aka 23:55) CET via Blackboard [2] (late submission until June 21, 2021, 11:55 pm CET). Furthermore, please keep in mind that the exams contribute 46% to your final grade, hence you need to submit at least one assignment (partially) to pass the course.

1.1 Support

If you have troubles understanding the assignment, please use one of the following communication channels to get help (in this order):

1. **Lecture:** mondays 08:00 - 10:00 am CET (exception: lecture-free periods).
2. **Slack:** `https://dbteaching.slack.com/archives/C01QQHPLFB2`.
3. **Email:** dkocher@cs.sbg.ac.at (as a last resort).

As usual, we recommend to start the assignment early in case you run into problems. The earlier you identify a problem the easier it is for the instructor and other students to provide help in time. **Remark:** Please notify the instructor as soon as possible if the assignment description is (partially) unclear or if there is another problem with the submission.

[1] `https://neo4j.com/docs/api/python-driver/current/` and `https://pypi.org/project/neo4j-driver/`
[2] `https://elearn.sbg.ac.at`

2 Assignment Description

Similar to the previous assignments, we split this assignment into 5 parts:

1. Installation of Neo4j, cf. Section 2.2.
2. Fill the database with some data, cf. Section 2.3.
3. Familiarize yourself with Neo4j and the Cypher query language, cf. Section 2.1.
4. Write an example application in Python that queries the database, cf. Section 2.5.
5. Answer the questionnaire, cf. Section 2.6.

Only part 4 and 5 contribute to the overall grade of this assignment. We recommend to follow these steps and refer to the corresponding sections for details. In addition, we briefly introduce Neo4j in Section 2.1.

2.1 Introduction to Graph Databases & Neo4j

Neo4j is a NoSQL database system that is based on the graph data model. Graphs are a rather intuitive concept in computer science that is almost a perfect fit for certain application scenarios. Nowadays, for example, most people are familiar with social networks like Facebook, Twitter, or Instagram. From a (very) simplified point of view, social networks resemble specific parts of the real world like the social connections/network and the preferences of a person (e.g., through a like system in Facebook). A graph consists of nodes that represent the actual data, and (directed) edges that represent the interrelations in the data. In our social network scenario, the persons are the nodes and the connections between the persons are the edges. Furthermore, different types of edges and nodes (with numerous different attributes) may be supported by a graph data model. This is referred to as *property graph* model. Figure 1 shows an example property graph that consists of four nodes (three persons; one movie) and four (directed) edges that encode the relationships between the nodes. For example, Mark and Christian both know Scarlett, but Scarlett does not know any person in this network. Additionally, Mark and Christian both like the movie "American Psycho" (resembled as edges of type "like"). For better understanding, the node/edge types are color-coded.

The property graph model of Neo4j also supports node labels, which represent the "role" of the node (and can also be used to store metadata). Edges are typically directed (i.e., have a direction) but support the navigation in both directions. For more details on the property graph model of Neo4j and how to model your data as graph, we refer to the Neo4j documentation [3] [4].

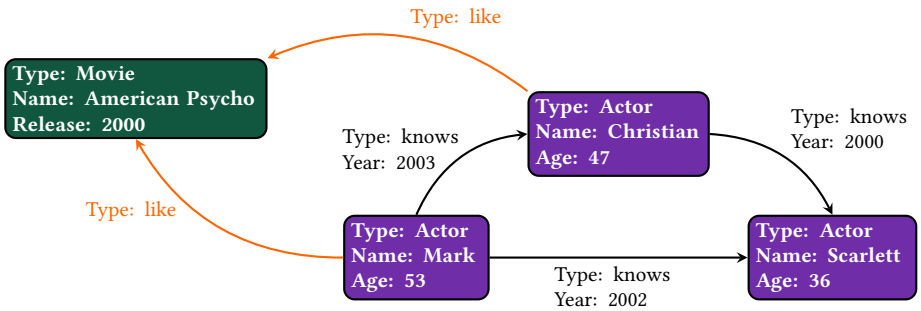

Figure 1: Example graph of a small social network.

[3] https://neo4j.com/developer/graph-database/#property-graph
[4] https://neo4j.com/developer/guide-data-modeling/

2.2 Neo4j Installation

Step 1 is to install Neo4j (Community Server edition) [5] locally on your machine. This should be possible on all popular operating systems, but the specific steps may diverge. All the details on how to install Neo4j on different operating systems can be found in the Neo4j documentation (cf. links given in Section 4). After installing Neo4j, you can create a database and fill it with some data. Although there are multiple ways to import data into Neo4j (e.g., via CSV file [6] or from a relational DBMS [7]), we provide you with a so-called *dump* of a graph that represents popular movies and actors. A dump (of a database) is a snapshot of the data and can be directly imported into the corresponding DBMS (in our case Neo4j).

Like many other systems, Neo4j has a web-based graphical user interface (GUI; that can be access with your browser) and a command-line tool to interact with the database. In this assignment, we encourage you to use both. Since the web-based GUI provides a way to query the data, everything but the data import and the Python code can be done using the web-based GUI. Nonetheless, we encourage you to try out the Neo4j command-line tools. The `cypher` command-line tool (aka `cypher` *terminal* or *shell*) is the most basic way to interact with this particular graph database. To import the data into the database, the `neo4j-admin` command-line tool comes in handy.

Remarks: Please use the command-line tool of your operating system to call Neo4j's command-line tools. On Windows systems, for example, use the `cmd.exe` (and navigate into the folder that contains Neo4j's command-line tools if it does not work from outside).

For the remainder of this assignment description, we assume that the commands/queries are executed using one of the following command-line tools:

- A system command-line tool (Windows: `cmd.exe`, Linux/MacOS: `terminal`).
- The Neo4j command-line tool (on all systems: `cypher-shell`) or the web-based GUI terminal.
- The Neo4j import command-line tool (on all systems: `neo4j-admin`).

Commands and queries are wrapped in a framed listing environment which also specifies the used command-line tool at the beginning of the title (separated by a dash –). The following listing shows an example that executes the command `dir` in Windows' `cmd.exe`:

`cmd.exe` – Show directories.

```
dir
```

The next listing shows an example that executes the command `SHOW DATABASES` in the Neo4j command-line tool:

`cypher-shell` – Show all databases.

```
SHOW DATABASES
```

After the installation, the first step is to validate that the Neo4j server is running on your system. This is most likely the case if you installed Neo4j as a daemon (or service on Windows) and can be checked using the following command:

[5] `https://neo4j.com/download-center/#community`
[6] `https://neo4j.com/developer/guide-import-csv/`
[7] `https://neo4j.com/developer/relational-to-graph-import/`

`terminal` – Check if Neo4j daemon is running.

```
ps aux | grep -v grep | grep neo4j
```

On Windows systems, one way to check if this service is running is to look for the `neo4j` process in the Task Manager. **Remarks:** Neo4j requires Java and is tested with Java version 11. In the best case, it also works with newer versions of Java. Nonetheless, both older and newer versions of Java may cause troubles (in particular, on Windows systems) when you try to install and start the Neo4j server. If you do not manage to install and run the Neo4j server, please contact the instructor as soon as possible to resolve this issue.

Once the Neo4j server runs, we can connect to our local database as follows:

`cmd.exe` / `terminal` – Connect to the local Neo4j server using the Cypher command-line tool.

```
cypher-shell
```

On the first connection, the Neo4j server will ask you for credentials to log into the system. The default credentials are user "neo4j" with password "neo4j". After the first login, you will be asked to set a new password which is to be used for subsequent logins.

The connection can also be established using the web-based graphical user interface by opening a browser and connecting to the following address:

`browser` – Connect to the web-based graphical user interface of your local Neo4j server (assuming the default port 7474).

```
localhost:7474
```

For both methods, we can start to interact with the database once the connection is established. As highlighted in red in Figure 2, the web-based GUI provides a terminal on top where queries and commands can be executed.

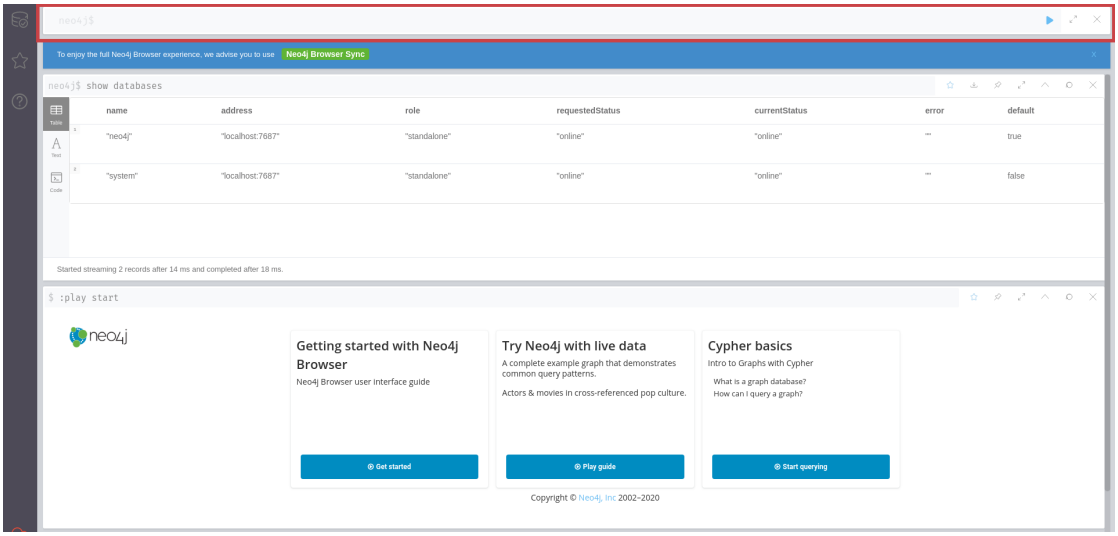

Figure 2: Terminal in the web-based graphical user interface of Neo4j (red).

Similar to PostgreSQL and MongoDB, Neo4j has a default database named "`neo4j`" and the name that is displayed on the left-hand side of the `cypher-shell` (command-line) tool is the

name of the database we are currently operating on (indicated by the `neo4j$` if we are on the `neo4j` database). We can use the following command to switch the database we operate on:

`cypher-shell` – Switch to database "`neo4j`" (mind the colon at the beginning).

```
:USE neo4j
```

To see all databases on our Neo4j server, we can use the following command:

`cypher-shell` – Show all databases on our server.

```
SHOW DATABASES
```

In Neo4j, it is required to create a database before we can fill it with data (contrarily to MongoDB, which creates the databases on the fly). Unfortunately, the Community Server edition does not support the creation a database [8]. However, for the sake of completeness, we still discuss how to create a new database (in case you work with the Professional edition eventually). We can use the following command to create a new database named "`assignment3`" and switch onto it:

`cypher-shell` – Create and switch onto database "`assignment3`".

```
CREATE DATABASE assignment3
:USE assignment3
```

In order to create a new database, you need to have administrator/root privileges, otherwise you may see the error message depicted in Figure 3 (for example, if you try to create a new database in the web-based GUI terminal). In this case, please make sure that the Neo4j user has the privileges that are required to create a new database (i.e., write permissions).

Figure 3: Error message in the web-based GUI terminal in case you do not have the required privileges.

As database for this assignment, please use the default database `neo4j`.

2.3 Data Initialization

After the installation, the database contains no data, i.e., it is empty. We can verify this by executing the following command, which shows all nodes and edges in our graph database.

`cypher-shell` – Show all nodes and edges in our database.

```
MATCH (n)-[r]->(m) RETURN n,r,m
```

[8] `https://stackoverflow.com/a/60431349`

Therefore, the next step is to populate the database with some data. In the course of this assignment, we will use data of a publicly available movies dataset, which is provided as database dump.

First, we must download the data from our Nextcloud [9]. Since the data is a dump of a database, we can simply import it directly into our `neo4j` database. For this purpose, Neo4j provides a command-line tool called `neo4j-admin`, which can be used as follows (note that `neo4j-admin` is called from the system command-line tool and may require to be executed with the `neo4j` user):

`cmd.exe` / `terminal` – Import the `movies` database dump.

```
neo4j-admin load --from movies-40.dump --force --database neo4j
```

This imports the `movies` database dump into the `neo4j` database, and we can verify this by executing the following command:

`cypher-shell` – Show all nodes and edges in our database.

```
MATCH (n)-[r]->(m) RETURN n,r,m
```

Remarks: (i) The `--from` option of this command also works with full paths, for example, `--from "C:\Users\dkocher\Desktop\movies-40.dump"`. (ii) In some scenarios, the `neo4j-admin load` command imports the dump with wrong privileges, i.e., the files and directories belong to the wrong user. Figure 4 shows the error message you may see if you try to access the `neo4j` database with wrong privileges. In this case, please ensure that all files and directories in the `neo4j` database belong to the `neo4j` user, e.g., by executing the commands shown in the listing below under Linux or MacOS (please double-check the directories paths since they may differ).

Figure 4: "Database unavailable" error message in the web-based Neo4j GUI.

`terminal` – Change the ownership of all files/directories of the `neo4j` database.

```
sudo chown -R neo4j:neo4j /var/lib/neo4j/data/databases/neo4j/
sudo chown -R neo4j:neo4j /var/lib/neo4j/data/transactions/neo4j/
```

2.4 Start Using Neo4j

Remark: Questions F1 and F2 in the questionnaire may also help to get used to Neo4j.

After importing the database dump, try to further familiarize yourself with the `cypher-shell` command-line tool (or the corresponding web interface). Neo4j does not support SQL but uses the Cypher query language (CQL), which is a declarative query language that provides a visual and intuitive way to match graph patterns. In the course of this assignment, we will use CQL to find patterns in our `movies` graph database and the Neo4j documentation provides a detailed explanation of CQL [10]. In the following, we briefly cover some basics of the Cypher query

[9] `https://kitten.cosy.sbg.ac.at/index.php/s/sg53E2e5TCe8L3i`
[10] `https://neo4j.com/developer/cypher/`

language and provide two (!) queries, Q1-Q2, that can be executed out of the box by entering them into the `cypher-shell` command-line tool (or the corresponding web interface) once the data was imported.

On the `neo4j` database, we can directly use CQL statements to query the data. The following query, for example, selects *all* pairs of nodes that have some relationship (i.e., edge):

`cypher-shell` – Select all pairs of nodes that have some relationship.

```
1 MATCH (n)-[r]->(m) RETURN n,r,m
```

The `MATCH` keyword is used to search for a node, relationship, or pattern in the database. In this specific case, we search for a pattern which consists of two nodes, `(n)` and `(m)`, and a relationship (i.e., edge), `-[r]->` that connects the two nodes. The arrow denotes the direction of the relationship (from `(n)` to `(m)`). As a result, the system will report (`RETURN`) all nodes and edges in the graph that match this specific pattern. `n` and `m` are variable names that can be used to refer to a particular node (e.g., in the `RETURN` statement). Analogously, `r` is the variable name of the corresponding relationship.

To limit the number of nodes/edges that are reported, we can use the following adapted CQL query:

`cypher-shell` – Select 10 pairs of nodes that have some relationship.

```
1 MATCH (n)-[r]->(m) RETURN n,r,m LIMIT 10
```

As mentioned before, in this assignment you are only given two queries in the Cypher query language, Q1-Q2. It is part of this assignment to study the Cypher query language and come up with two additional CQL queries on your own. For query Q3, you should study the *Updating* section of the documentation [11] and either create a new relationship between two (or more) nodes or update an existing relationship between two (or more) nodes. For query Q4, you should study the *Filtering Query Results* section of the documentation [12] and apply a filter on a pattern. It is up to you what you create/update and which filter you apply. However, the queries are supposed to be *meaningful*, i.e., meaningful in the sense that the query has some effect (reports a result or create/updates an edge). Just try to explore the data and the Cypher query language. You are of course free to use more advanced CQL features (e.g., graph algorithms) for queries Q3 and Q4, but creation/update (Q3) and filtering (Q4) are the minimum requirements.

Before we continue with queries Q1 and Q2, we look at the following CQL `MATCH` query in more detail:

`cypher-shell` – Find all movies of person (actor) “Christian Bale”.

```
1 MATCH (n:Person {name:"Christian Bale"})-[r:ACTED_IN]->(m:Movie) RETURN n,r,m
```

Similar to the previous CQL query, we define the pattern to have two nodes that are connected via some relationship. However, this time we specify the following additional details: (i) The type of the node/edge and (ii) a property for the left-hand node. (i) After the variable name of a node/edge, we can specify the type of the node/edge (separated by a colon). In our example, we search for nodes of type `Person` with a connection to nodes of type `Movie`. The relationship between the nodes must be of type `ACTED_IN`. (ii) {`name: ''Christian Bale''`} is a property of a node that must be satisfied to match the pattern. Consequently, we only find

[11] `https://neo4j.com/developer/cypher/updating/`

[12] `https://neo4j.com/developer/cypher/filtering-query-results/`

movies of the person “Christian Bale”. For more details, we refer to the Neo4j documentation on *Querying* [13].

Now that we have learned about the basics of CQL, we give the queries Q1 and Q2 below (please use SHIFT+ENTER for multi-line queries in the web-based GUI, and the blue Play button on the right-hand side to execute it):

Query Q1: `cypher-shell` – Find all persons that acted in a movie with “Christian Bale”.

```
MATCH
  (b:Person {name: "Christian Bale"})
  -[a1:ACTED_IN]->(m:Movie)<-[a2:ACTED_IN]-
  (p:Person)
RETURN b,a1,m,a2,p
```

In query Q1, we specify a pattern that contains three nodes and two `ACTED_IN` relationships (in opposite directions). Semantically, this allows us to identify all persons that acted in a movie with “Christian Bale”.

Query Q2: `cypher-shell` – Find the degree of separation between the persons (actors) “Christian Bale” and “Halle Berry”.

```
MATCH
  p=shortestPath(
    (b:Person {name:"Christian Bale"})-[*]-(b2:Person {name:"Halle Berry"})
  )
RETURN p
```

Query Q2 executes an operation that is common for graphs, namely a *shortest path* query [14]. Almost everybody has implicitly used a shortest path in a navigation system to navigate from one physical location to another one. A shortest path algorithm in a navigation system identifies the path that has the least costs to reach the other location (with respect to some cost model that includes, for example, the path length and/or the current traffic). However, shortest paths are a general concept in graphs and can also be used to identify the shortest path of relationships between persons in social networks. In our specific example, query Q2 finds the degree of separation between the two actors (persons) “Christian Bale” and “Halle Berry”, i.e., the connection between “Christian Bale” and “Halle Berry” in hops through persons and movies.

This concludes the two given queries Q1 and Q2. Recall that it is part of this assignment to come up with two additional queries Q3 (create a new edge or update an existing edge) [15] and Q4 (apply a filter in a CQL query) [16] on your own by studying the Neo4j documentation and the data in the web-based GUI of Neo4j.

2.5 Access the Data Using Python

Similar to most database systems, Neo4j is typically accessed by an application. Therefore, the fourth part of this assignment is to write a small Python application that executes all queries, Q1-Q2 (given in Section 2.1) and Q3-Q4 (developed/introduced by your team), one after another and prints their results to the command line. We recommend to use the `neo4j-driver` module (or driver) for Python to (a) establish a connection to your local database, (b) execute the queries and retrieve the results, and (c) close the connection to your local database. **Re-**

[13] `https://neo4j.com/developer/cypher/querying/`
[14] `https://en.wikipedia.org/wiki/Shortest_path_problem`
[15] `https://neo4j.com/developer/cypher/updating/`
[16] `https://neo4j.com/developer/cypher/filtering-query-results/`

mark: If you cannot execute all queries (for whatever reason), please contact the instructor *before* the submission. We will find a reasonable solution together.

Template Code There are many tutorials regarding the installation [17] and the usage of the `neo4j-driver` [18] module to access the Neo4j database. Nonetheless, we provide a minimum template code in Python3 that can be used as a starting point.

Listing 1: Template code to access a database using `neo4j-driver`.

```
#!/usr/bin/python3

from neo4j import GraphDatabase, basic_auth
# A module that formats the output in a readable format
import pprint

# A function that takes a transaction object "tx" as first parameter and a
# cypher_query as second parameter. The Cypher query is then executed (run)
# using the transaction object and the retrieved data is returned as result.
def get_result(tx, cypher_query):
  return tx.run(cypher_query).data()

if __name__ == "__main__":
  try:
    # The string that is used to connect to the Neo4j server. In this case, we
    # use the "bolt" connection, which has a different default port (7687)
    # compared to the web-based graphical user interface (which has default
    # port 7474).
    connection_string = "bolt://localhost:7687"

    # Establish a connection to the local Neo4j server with the default
    # credentials.
    driver = GraphDatabase.driver(
      connection_string,
      auth=basic_auth("neo4j", "neo4j"))
  except:
    print("Unable to connect to {}".format(connection_string))

  try:
    # Create a Cypher query (result size limited to 10)
    cypher_query = '''
      MATCH (n)-[r]->(m)
      RETURN n, r, m
      LIMIT 10
    '''

    # Create a session (i.e., context) for the transactions to be executed.
    session = driver.session(database="neo4j")

    # Execute a read transaction based on the give cypher_query and the given
    # function (get_result). read_transaction takes a function as parameter
    # (get_result) that in turn takes a transaction object as parameter (cf.
    # "tx" in the definition of get_result). The cypher_query parameter is then
    # passed to the corresponding transaction "tx" in the "get_result" function.
    result = session.read_transaction(get_result, cypher_query)

    # Advanced lambda-based version to get the result of a Cypher query:
    # results = session.read_transaction(
    #   lambda tx: tx.run(cypher_query))

    # Print the result set to the command line. We use the pprint module to
    # print the result in a (more or less) human-readable format (the standard
    # print function prints the result in a single line). It is up to you
    # whether you want to use pprint or not.
    for entry in result:
      pprint.pprint(entry)
```

[17] `https://neo4j.com/docs/api/python-driver/current/` and `https://pypi.org/project/neo4j-driver/`
[18] `https://neo4j.com/developer/python/`

```
  except Exception as e :
    print("Unable to execute simple MATCH query: {}".format(e))
  finally: # The finally - branch is executed independently of an exception.
    if session is not None:
      # Close the session.
      session.close()

    if driver is not None:
      # Close the connection.
      driver.close()
```

2.6 Questionnaire

The questionnaire contains questions about the assignment. These questions are potentially discussed during the after-assignment meetings. The questionnaire can be found in a separate textfile called `assignment3-questionnaire.txt`.

3 Submission

Please submit a single compressed archive (e.g., `.zip` or `.tar.gz`) that contains two files: (a) The code of your Python3 application and (b) the answers to the questionnaire.

Code Please submit a single Python3 file (`.py`) that contains the full code for this assignment, i.e., the connection to the database and the execution of the four queries. **Remark:** Please remove the database credentials (i.e., username and password) before you submit your code.

Questionnaire You can answer the questions directly in the textfile `assignment3-questionnaire.txt`. If you prefer to use a different application to answer the questions (e.g., Microsoft Word and the likes), you are welcome to do so. In any case, the submitted file must be in one of the following formats: `.txt`, `.pdf`, `.odt`, `.doc`, or `.docx`. **Remark:** The recommended formats are `.txt` and `.pdf`.

4 Supplementary Material

This section provides a list of pointers to material that may be helpful to solve the assignment.

- Neo4j: `https://neo4j.com/`
- The full Neo4j documentation: `https://neo4j.com/docs/`
- Neo4j "Getting Started": `https://neo4j.com/developer/get-started/`
- Neo4j "Introduction to Graph Databases Video Series": `https://neo4j.com/developer/intro-videos/`
- Neo4j resources regarding the installation on
  - Linux systems: `https://neo4j.com/docs/operations-manual/current/installation/linux/`
  - MacOS systems: `https://neo4j.com/docs/operations-manual/current/installation/osx/`
  - Windows systems: `https://neo4j.com/docs/operations-manual/current/installation/windows/`
- Other resources regarding the installation and the usage of Neo4j:

  - Two of the many Neo4j "Getting Started" guides:
    1. Windows: `https://www.youtube.com/watch?v=hWDqOGSi7Tc`
    2. Linux/MacOS: `https://www.youtube.com/watch?v=M2JSvN1Afw8` (without the custom configuration at the end)
  - Comparing SQL with Cypher: `https://neo4j.com/developer/cypher/guide-sql-to-cypher/`
- Neo4j data model concepts: `https://neo4j.com/docs/getting-started/current/graphdb-concepts/`
- Reference for the `cypher-shell` command-line tool and its web-based counterpart (the graphical user interface):
  `https://neo4j.com/developer/cypher/` and `https://neo4j.com/docs/cypher-manual/current/`
- Python Modules: `https://docs.python.org/3/installing/index.html`
- The `neo4j-driver` module: `https://neo4j.com/docs/api/python-driver/current/`
- Installation of the `neo4j-driver` module for Python:
  - Official website: `https://neo4j.com/docs/api/python-driver/current/#installation`
  - The Python Package Index: `https://pypi.org/project/neo4j-driver/`
- The `neo4j-driver` tutorial: `https://neo4j.com/developer/python/`
- An introduction to all Neo4j drivers (including the `neo4j-driver` Python module):
  `https://www.youtube.com/watch?v=JwgxkGY-36Q`

5 Grading

For the sake of transparency, this section provides more details on the grading of this assignment, i.e., which part contributes how many points to the total number of 18 points for this assignment.

Code The code contributes at most 10 points and is evaluated based on the following criteria (if the code is executed as submitted; disregarding the credentials):

Max. Points	Criterion
1.5	Q1 is executed and the correct result is printed to the command line.
1.5	Q2 is executed and the correct result is printed to the command line.
1.5	Q3 is executed and the (correct) result is printed to the command line.
1.5	Q4 is executed and the (correct) result is printed to the command line.
1.5	Q3 is a syntactically correct and meaningful query.
1.5	Q4 is a syntactically correct and meaningful query.
1	Answer 1 question regarding your code in the after-assignment meeting.
10	

Questionnaire The questionnaire contributes at most 8 points and is evaluated based on the following criteria (taking the discussion in the after-assignment into account):

Max. Points	Criterion
2	Correctness of answer A1.
2	Correctness of answer A2.
2	Correctness of answer A3.
2	Correctness of answer A4.
8	