UV Distributed Information Management
Summer semester 2023

Assignment 03

Summary:

Deadline: June 21, 2023, 11:55 pm (aka 23:55) CET.

Extended Deadline: June 28, 2023, 11:55 pm (aka 23:55) CET.

Submission: Submit a compressed archive (e.g., a `zip` or a `tar.gz` file) that contains your Python3 code and your answers to the questionnaire via Blackboard.

Grading: 55% Python3 code, 45% answers (incl. meeting; cf. Section 5 for details).

---

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, Neo4j is open-source[1] and rather easy to install. Since the graph data model is very intuitive, we will also make use of the graphical user interface (GUI) of Neo4j to interact with the database. However, also Neo4j provides (1) a command-line tool named `cypher-shell` (where *Cypher* stands for Neo4j's *Cypher query language*), and (2) an application programming interface (API) for the Python3 programming language (i.e., with the `neo4j-driver`[2] module).

Please submit your final Python code *and* your answers to the questionnaire until June 21, 2023, 11:55 pm (aka 23:55) CET via Blackboard[3] (late submission until June 28, 2023, 11:55 pm (aka 23:55) 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 Formatting Conventions

Commands for the Linux command-line tool (`terminal`), the command-line tool of Neo4j (`cypher-shell`), and Python3 code are written in `TrueType` font[4]. In addition, all commands are in a box that specifies the used command-line tool at the beginning of the title (separated by a dash –, i.e., **`terminal`** for Linux and **`cypher-shell`** for Neo4j). Listing 1 shows an example command executed in the Linux `terminal`:

Listing 1: **`terminal`** – Show directories.

```
dbtutorial@database-tutorial:~# ls -l
```

The Linux `terminal` shows a prefix `dbtutorial@database-tutorial:~#` that consists of

[1] Neo4j code on GitHub: `https://github.com/neo4j/neo4j`
[2] `https://neo4j.com/docs/api/python-driver/current/` and `https://pypi.org/project/neo4j-driver/`
[3] Blackboard: `https://elearn.sbg.ac.at`
[4] `TrueType` font: `https://en.wikipedia.org/wiki/TrueType`

- the name of the user that executes the command; `dbtutorial` is the default user of the virtual machine (VM) (this may be different if you do not use the given VM),
- the name of the machine; `database-tutorial` is the name of the given VM, and
- a delimiter that separates the actual command from the "`user@machine`" string; "`:~#`" is the default delimiter of the given VM (where ~ denotes the current directory).

Contrarily, Listing 2 exemplifies a command in Neo4j's command-line tool, i.e., the `cypher-shell`. It shows an example that executes the command `SHOW DATABASES` while connected to a database named `neo4j`:

Listing 2: **`cypher-shell`** – Show all databases.

```
neo4j@neo4j> SHOW DATABASES;
```

Notably, Neo4j provides a graphical user interface to the `cypher-shell` and you are free to choose between command-line and graphical interface. For commands that are to be executed in the `cypher-shell`, the non-bold prefix `neo4j@neo4j>` denotes the user that is connected in combination with the database we are currently connected to (separated by a "@"; cf. Section 2.3). Once again, the Cypher query language (CQL) is different from SQL and MQL, and capitalization of keywords (like `SHOW`) is optional. For readability, we capitalize Neo4j keywords. More information can be found in the supplementary material given in Section 4. Python3 code is simply wrapped in a box.

Like MongoDB, Neo4j has a dedicated command-line tool called `neo4j-admin` that is used to import data into the database. As the name of this tool suggest, `neo4j-admin` can be used for other administrative matters (like exporting or migrating an existing database), but these are not relevant for this assignment.

Remark: Interested students can check the other options of `neo4j-admin`, e.g., by reading the corresponding section in the Neo4j documentation[5].

1.2 Support

Remark: Please notify the instructor as soon as possible if this assignment description is (partially) unclear or if there is a problem with the submission.

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

1. **Lecture:** mondays 10:15 am - 12:00 pm CET, wednesdays 01:00 - 02:30 pm CET (exception: lecture-free periods).
2. **Slack:** `https://dbteaching.slack.com/archives/C04QHH4TR7B` (I will check regularly and do my best to reply fast, but please do not expect me to be available 24/7).
3. **Email:** dkocher@cs.sbg.ac.at (as a last resort).

We recommend to start the assignment early. In case of a problem, it is easier for the instructor and other students to provide help in time if you identify problems early.

2 Assignment Description

We **highly recommend** that you read this section (including all the subsections) to the end before you start working (this should be less error-prone).

[5]The `neo4j-admin` tool: `https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/`

Similar to Assignment 1 and 2, we split this assignment into five parts and only parts 1 (depending on your choice), 4, and 5 contribute to the overall grade of this assignment. We recommend to follow these steps and refer to the corresponding sections for more details.

1. Set up and configure Neo4j, cf. Section 2.2.
2. Fill the database with some data, cf. Section 2.4.
3. Familiarize yourself with Neo4j and the Cypher query language, cf. Section 2.1.
4. Write an example application that accesses the database in Python3; cf. Section 2.6.
5. Answer the questionnaire, cf. Section 2.7.

2.1 Introduction to Graph Databases & Neo4j

Neo4j is a NoSQL database system that is based on the graph data model[6]. 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 (Facebook, Twitter, Instagram, and the likes) and navigation systems (e.g., Google or Apple Maps). From a (very) simplified point of view, social networks resemble specific parts of the real world like the social connections (i.e., the network) and the preferences of a person (e.g., through a like-based system). A graph consists of nodes that represent the actual data, and (directed) edges that represent the interrelations in the data. As an example social network, we consider nodes that represent persons and the connections between the persons are the edges. Typically, a graph data model supports different types of edges and nodes (with numerous different attributes), which 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 readability, the node/edge types are color-coded, i.e., nodes of type "Actor" are shown in **grape**, nodes of type "Movie" are encoded in **green**, whereas edges of type "like" and "knows" are visualized in **orange** and **black**, respectively.

The property graph model of Neo4j also supports node labels that represent the "role" of the node (and can also be used to store meta data). 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[7] [8].

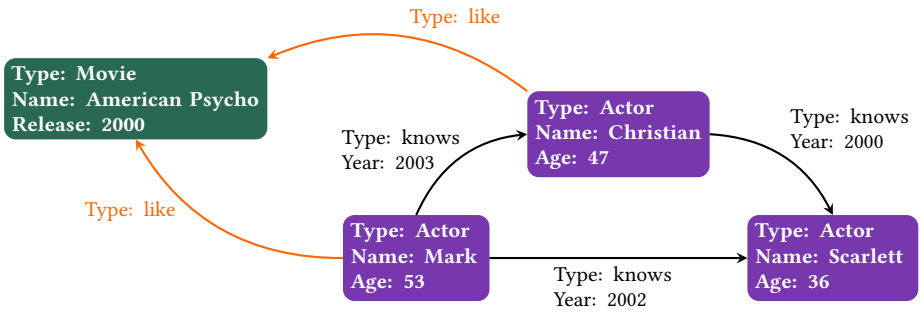

Figure 1: Example graph of a small social network.

[6]Graph database: `https://en.wikipedia.org/wiki/Graph_database`
[7]Graph model: `https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/`
[8]Modeling: `https://neo4j.com/docs/getting-started/data-modeling/relational-to-graph-modeling/`

We conclude with Figure 2, which depicts the popularity changes per database category[9] (i.e., data model) over time, starting from 2013. In Figure 2, we observe that graph databases rise in popularity and are thus highly relevant.

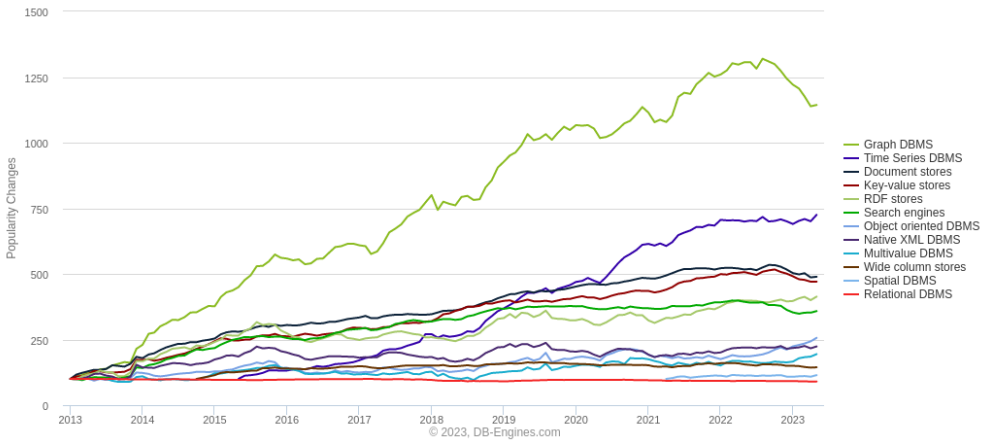

Figure 2: Popularity changes per database category since 2013 (`https://db-engines.com`).

2.2 Neo4j Setup

Similar to Assignment 1 and Assignment 2, you have three possible ways to set up and configure Neo4j in order to solve this assignment:

Neo4j Installation Install Neo4j on your own system (or virtual machine) using an operating system of your choice. This implies that you "pollute" your system with this installation and that the instructor may need additional information on your particular setup to provide help (since we mostly work on Linux systems). Nonetheless, we want to emphasize that it is an **excellent exercise** for students to perform the installation of such a system on their own (at least once in their career). If you choose this option, you can directly jump to Section 2.2.1.

Neo4j using a VM Configure a pre-installed Neo4j instance in a virtual machine (VM) that runs Debian Linux. This implies that you will not experience the process of installing Neo4j on your own, but it may be easier for the instructor to help since the VM runs Debian Linux (and, ideally, the setup is reproducible). On the one hand, it may be cumbersome if you do not have any experience with Linux systems (and VMs), but on the other hand it may also be a nice opportunity to familiarize yourself and play around with a Linux system (and a VM). **Remark:** We want to note that **Neo4j may run slowly in the VM,** in particular when using the graphical user interface. To continue with this option, you can jump to Section 2.2.2.

Neo4j using Docker Configure a pre-installed Neo4j instance in a Docker container that runs Debian Linux. For new chipsets (e.g., Apple Silicon[10]) with different architectural characteristics (compared to typical Intel and AMD chipsets), running VMs properly is still problematic, triggering quite some bugs. Therefore, you can also use an experimental setup (i.e., you are among the first students to try it out) using Docker, which basically mimics a lightweight VM without a graphical user interface (i.e., this is the most advanced

[9] DB-Engines popularity change: `https://db-engines.com/en/ranking_categories`
[10] Apple Silicon chipset: `https://en.wikipedia.org/wiki/Apple_silicon`

option). **Remark:** As Docker is not supposed to resemble a full virtual machine, you **cannot make use of the graphical user interface** (unless you are used to Docker and know how to access the service that is running in the Docker container from outside the container – but we will not explain it here because this is not the focus of this assignment). For this option, you can jump to Section 2.2.3.

From our experience, however, the installation of Neo4j should not cause many troubles. Hence, we encourage everyone to give it a try. You can choose your preferred way and the choice itself will not influence your grade in any way. However, make sure that you can answer questions regarding your choice.

2.2.1 Neo4j Installation

Remarks: (i) If you decide to use the VM image or Docker, you can directly jump to Section 2.2.2 and Section 2.2.3, respectively. Neo4j is already pre-installed in both cases. (ii) 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` or the `Powershell` (and navigate into the directory that contains Neo4j's command-line tools, i.e., the directory that you chose during installation).

As a first step, we are required to install Neo4j (Community Server edition) in version 5.8.0[11] locally on your machine.

Remark: Please make sure to use version 5.8.0 as some commands may be different for older versions of Neo4j, and the data to import into the database may be incompatible with older Neo4j versions.

Neo4j should be available for 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 successfully, 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[12] or from a relational model[13]), we provide you with a so-called *database dump*[14] 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 database system (in our case Neo4j). Database dumps are commonly used to create a backup of a database.

Like many other systems, Neo4j has a (web-based) graphical user interface (GUI; that can be access using 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 Python3 code can be done using the web-based GUI. Nonetheless, we encourage you to try out the Neo4j command-line tool. The `cypher-shell` command-line tool (aka `cypher` *terminal*) is the most basic way to interact with this particular graph database. In order to import the data into Neo4j, the `neo4j-admin` command-line tool ca be utilized.

2.2.2 Virtual Machine Setup

Remark: Neo4j may run slowly in the VM, in particular when using the graphical user interface. This is mainly due to the higher resource requirements of the graphical user interface. If possible, you may want to consider giving the VM more memory (and CPUs) in order to speed it up.

We provide a so-called virtual machine (VM) *image*, which consists of a pre-installed Debian Linux as operating system and a pre-installed Neo4j instance. For this option, you may have

[11] Neo4j download center: `https://neo4j.com/download-center/#community`
[12] CSV import: `https://neo4j.com/docs/getting-started/data-import/csv-import/`
[13] Rel. import: `https://neo4j.com/docs/getting-started/data-import/relational-to-graph-import/`
[14] Database dump: `https://en.wikipedia.org/wiki/Database_dump`

to familiarize yourself with the concept of a virtual machine (VM) and how to host the corresponding image (cf. Assignment 0 for details). In essence, a virtual machine[15] is a software that is designed to provide you with a substitute of a real machine, that is, it runs another operating system on top of your regular operating system – it **virtualizes** another system. For Assignment 3, it is enough to install a software named VirtualBox[16] that acts as a host for virtual machines. After you successfully installed VirtualBox, the VM image can be used to set up the virtual machine for this assignment. The image (as `ova` file) can be downloaded from our Nextcloud[17] and runs Debian Linux[18] as operating system with an installation of Neo4j.

After downloading the `debian-neo4j-vm.ova` file, this file must be imported into VirtualBox. There are many online tutorials[19] on how to accomplish this and it primarily consists of two steps (cf. Assignment 0): (1) Choose the image to import (navigate to the downloaded file) and (2) check/modify the settings of the virtual machine (typically no modifications are necessary, but sometimes, for example, you may have to disable the USB port). Once you click on the *Import* button, VirtualBox should import the image and set up the virtual machine that can be used for this assignment. VirtualBox then shows a new VM on the left-hand side, which can be started/booted with a double-click.

For this VM, there exist two users: (1) `dbtutorial` and (2) `root`. If you are not familiar with the concept of a `root` user (or superuser), please check out the corresponding article [20] on Wikipedia. In essence, the `root` user has all privileges whereas the `dbtutorial` user has not. By default, we will work with the `dbtutorial` user, but we will temporarily switch to the `root` user if we need additional privileges (e.g., to install a software package or to start/stop a service). The password for both users is the same: `dbpwd1`

Once you logged into the VM and before you start working on the actual assignment, try to familiarize yourself a bit with Debian Linux (unless you already have experience using Linux systems and/or solved Assignment 0). For example, open a `terminal` (aka Linux *shell* or *command-line tool*) and try some basic commands (some links can be found in the supplementary material, cf. Section 4). In particular, we will use the `cypher-shell` command-line tool (aka Neo4j *terminal*), which provides the most basic way to use Neo4j.

Remark: Note that you may not be able to use the typical `CTRL+C` and `CTRL+V` sequence to copy and paste between your regular system and the VM as the clipboard is not shared by default[21].

2.2.3 Docker Setup (Experimental)

Remarks: (i) This is the most advanced option (i.e., no graphical user interface is available) and still experimental (i.e., you are among the first students to try it out). (ii) With this particular Docker setup, you will not be able to use the graphical user interface of Neo4j out of the box. In order to access the graphical user interface, the Neo4j service must be made available from *outside* the container to your native operating system and browser. Since this assignment is not about learning Docker, we do not describe the necessary steps, but the interested students must read the documentation on their own.

Please install Docker Desktop[22] for *your* system configuration: Installations exist for Linux, Windows, MacOS with Intel chipset, and MacOS with Apple Silicon chipset. The documentation provides information on how to install and start Docker for your particular system.

[15] Virtual machine: `https://en.wikipedia.org/wiki/Virtual_machine`
[16] VirtualBox: `https://www.virtualbox.org/manual/ch02.html`
[17] VM image for this assignment: `https://kitten.cosy.sbg.ac.at/index.php/s/9La6YkYnNzdsRgA/download`
[18] Debian Linux: `https://en.wikipedia.org/wiki/Debian`
[19] Virtual machine import: `https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html`
[20] The `root` user: `https://en.wikipedia.org/wiki/Superuser`
[21] Shared clipboards in VirtualBox: `https://www.youtube.com/watch?v=fqrJ7qlhJu0`
[22] Getting Started with Docker: `https://www.docker.com/get-started/`

Docker[23] is a software package that also virtualizes at the level of an operating system (e.g., you can run Debian Linux within Windows or MacOS) but follows a different philosophy. Traditional VMs are *stateful*, meaning that the state of your VM can be stored and you are able to resume from it later on. Contrarily, Docker is designed to be *stateless*, meaning that it is not as easy to store the state and resume from it. This typically implies that the **effects of all your commands are lost** once you shut down Docker (or the terminal it is running in); but we will learn about a workaround to avoid this later in this section. Furthermore, Docker operates at the (finer) granularity of so-called *containers* where a container is meant to be a lightweight building block with a single responsibility, i.e., it typically runs a single service. Then, multiple of these containers can be used to implement a broader functionality (the containers can be "stacked" like in a dock). In contrast, a traditional VM is a full-fledged, virtualized machine that may run many different services at once.

For this assignment, we will use Docker similar to our VM image: It uses Debian Linux with a pre-installed Neo4j instance. To this end, please download a so-called *Dockerfile* from our Nextcloud[24]. You can then build and run the Docker *image* in your Linux or MacOS `terminal` as well as Windows' Powershell[25] as shown in Listing 3. Line 1 builds the Docker image and pulls a Debian image from the world wide web (i.e., you must be connect to the internet) and tags/names the resulting image as `neo4j`. In line 2, we run the built Docker image by referring to its tag/name (`run -it neo4j`) and specify a directory that is shared between Docker and your "regular" operating system (in this case, the current directory `${PWD}` can be accessed within Docker in `/home/dbtutorial`). Finally, we end up as user `dbtutorial` in the running Docker image as shown in line 3; and a (at first sight) cryptic machine name (`6fb3eb1459c7` in this example). Recall that Docker uses containers and `6fb3eb1459c7` uniquely identifies the current container (as virtual machine, if you want).

Remark: The container hash will be different on your machine.

Listing 3: **`terminal`** – Build and run the Docker image (mind the trailing dot in line 1).

```
1 dbtutorial@database-tutorial:~$ docker build --pull -f "Dockerfile" -t neo4j .
2 dbtutorial@database-tutorial:~# docker run -v ${PWD}:/home/dbtutorial/ -it neo4j
3 dbtutorial@6fb3eb1459c7:~$
```

Within this container, we can now use Debian Linux (cf. Section 2.3) but we only have a command-line interface (no graphical user interface). To this end, we briefly study a few additional Linux commands that may be helpful in the course of this assignment:

> **`touch assignment3.py`** Creates a new, empty file named `assignment2.py` within the current working directory. If you create this file within `/home/dbtutorial`, the file will also be available on your host system through the shared directory mentioned above.
>
> **`nano assignment3.py`** Opens a command-line editor named `nano`[26] to modify the file `assignment3.py`. At the bottom of `nano`, you find the most important keyboard shortcuts to interact with the editor, e.g., `CTRL+O` saves (i.e., overwrites) the file and `CTRL+X` closes `nano`. As an alternative, you may also use a command-line editor named `vim`[27].

If we close the Docker container using the `exit` command, we lose the effects of all commands that have been executed in our container (including files that we created within the container). This is also reflected by the fact that the container has a different hash (name) if we run it again

[23] The Docker software package: `https://en.wikipedia.org/wiki/Docker_(software)`
[24] Dockerfile for this assignment: `https://kitten.cosy.sbg.ac.at/index.php/s/tDc53PyXHYE5EsD/download`
[25] Windows' Powershell: `https://en.wikipedia.org/wiki/PowerShell`
[26] The `nano` editor: `https://en.wikipedia.org/wiki/GNU_nano`
[27] The `vim` editor: `https://en.wikipedia.org/wiki/Vim_(text_editor)`

(cf. line 2 in Listing 3). For convenience, we may want to save the state (although it is not in line with the Docker philosophy). We can accomplish this by opening another terminal (or Powershell) *outside* of our running Docker container and executing the commands shown in Listing 4.

Listing 4: **terminal** – Commit and resume a container (in this case `6fb3eb1459c7`).

```
1 dbtutorial@database-tutorial:~$ docker commit 6fb3eb1459c7 neo4j-save
2 sha256:6cab.............................
3 dbtutorial@database-tutorial:~$ docker images
4 REPOSITORY        TAG         IMAGE ID        CREATED           SIZE
5 neo4j-save   latest     6cab........   19 seconds ago   630MB
6 dbtutorial@database-tutorial:~# docker run -it 6cab........
7 dbtutorial@c69e4ff74ef6:~$
```

First, we use `docker commit` with the hash of our Docker container (which is still running) and a name, e.g., `neo4j-save` (line 1). This "saves" the current state of our Docker container into an image, which has a unique image ID that is different from the hash of our running Docker container. We can confirm this by executing `docker images` (line 3), which lists all committed images. To resume this specific Docker container, we must use the image ID (*not* the hash of the container) combined with `docker run -it` (line 6). Line 7 then shows a different container hash (i.e., `c69e4ff74ef6`) but this container resembles the state that has been committed before.

Remark: *Any feedback* on the current Docker setup is *very much appreciated*; by "experts" that are familiar with Docker as well as by students that are new to Docker.

2.3 Neo4j Configuration

After the installation, the first step is to validate that the Neo4j daemon (or service) is running on your system. A daemon is a process that runs in the background. As for MongoDB, the Neo4j daemon manages the data and the requests[28]. This is most likely *not* the case and it is part of this assignment to (re-)start the Neo4j daemon named `neo4j` as shown in Listing 5.

Remark: The `dbtutorial@database-tutorial:~$` (as well as `root@database-tutorial:~#`) at the beginning should already be displayed in your terminal and is not part of the command itself (which is `su -`). Do not get confused by the fact that nothing is shown when you type in the password (not even asterisks "*", which are often used for passwords). This is the default behavior of Linux and you will be asked again if the password is incorrect.

First, we confirm that the Neo4j daemon is *not* running (line 1) using the `systemctl` command. This is only a passive command, hence we can execute this command with the `dbtutorial` user. We observe that the state of our Neo4j daemon is `inactive (dead)`, i.e., it exists but is not running. Next, we need to execute an active command that *modifies* the state of our operating system by starting the Neo4j daemon. To this end, we must first switch to the `root` user (line 6) as the `dbtutorial` user does not have enough privileges to perform this action. Then, we use the `systemctl` command to (re-)start the Neo4j daemon (line 8). Line 9 then switches back to the `dbtutorial` user (as we do not want to stay in `root` mode).

Afterwards, we confirm that the Neo4j daemon is running (cf. Listing 6).

On Windows systems, one way to see whether the Neo4j daemon is running is to look for a process called `neo4j` in the Task Manager[29].

[28] Neo4j daemon: `https://neo4j.com/docs/operations-manual/current/installation/linux/systemd/`
[29] Windows' Task Manager: `https://en.wikipedia.org/wiki/Task_Manager_(Windows)`

Listing 5: **terminal** – (Re-)Start the Neo4j daemon.

```
dbtutorial@database-tutorial:~$ systemctl status neo4j
● neo4j.service - Neo4j Graph Database
     Loaded: loaded (/lib/systemd/system/neo4j.service; disabled; vendor ...)
     Active: inactive (dead)
dbtutorial@database-tutorial:~$ su -
Password:
root@database-tutorial:~# systemctl restart neo4j
root@database-tutorial:~# exit
```

Listing 6: **terminal** – Confirm that the Neo4j daemon is running.

```
dbtutorial@database-tutorial:~$ systemctl status neo4j
● neo4j.service - Neo4j Graph Database
     Loaded: loaded (/lib/systemd/system/neo4j.service; disabled; vendor ...)
     Active: active (running) since Tue 2023-05-02 13:12:12 CEST; 5s ago
   Main PID: 57756 (java)
     Memory: 305.7M
        CPU: 5.330s
     CGroup: /system.slice/neo4j.service
             ÄÄ 57756 /usr/bin/java -cp /var/lib/neo4j/plugins:...
```

In this case, the `systemctl` command shows a lot of status information about the Neo4j daemon, e.g., the starting time, a process ID (PID), used memory, and the likes. This information may be different on your system, and the most important information is that the Neo4j daemon is `active (running)`. Once the Neo4j daemon runs, we can connect to our local database using the `cypher-shell` as shown in Listing 7.

Listing 7: **terminal** – Connect to the local Neo4j server.

```
dbtutorial@database-tutorial:~$ 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 (e.g., Firefox in the VM) and connecting to address **`localhost:7474`** (without `www`).

For both methods, we can start to interact with the database once the connection is established. As highlighted in red in Figure 3, the web-based GUI provides a terminal on top where queries and commands can be executed.

Similar to PostgreSQL and MongoDB, Neo4j has a default database named `neo4j`. The name that is displayed on the left-hand side of the `cypher-shell` (command-line) tool as well as in the web-based graphical user interface is the name of the database we are currently operating on (e.g., `neo4j@neo4j>` in the `cypher-shell` and `neo4j$` in the graphical user interface both indicate that we operate on the `neo4j` database). To list all databases that are available on our Neo4j server, we can use the command shown in Listing 8

Listing 8: **cypher-shell** – Show all databases on our server (mind the trailing semi-colon).

```
neo4j@neo4j> SHOW DATABASES;
```

In order to switch to specific database, e.g., the `neo4j` database, we can use the command in Listing 9.

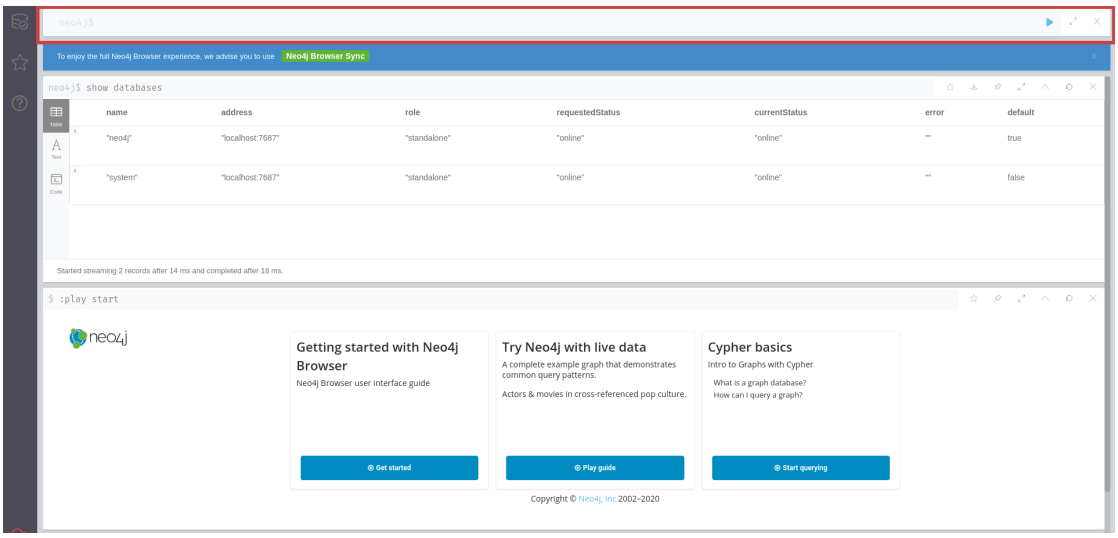

Figure 3: Terminal in the web-based graphical user interface of Neo4j (red).

Listing 9: **cypher-shell** – Switch to database `neo4j` (mind the colon at the beginning).

```
1  neo4j@neo4j> :USE neo4j
```

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[30]. **Therefore, please use the default database `neo4j` as database for this assignment.**

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 command shown in Listing 10 to create a new database named `assignment3` and switch onto it.

Listing 10: **cypher-shell** – Create and switch onto database `assignment3`.

```
1  neo4j@neo4j> CREATE DATABASE assignment3;
2  neo4j@neo4j> :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 4 (for example, if you try to create a new database using the web-based graphical user interface). 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).

2.4 Data Initialization

A fresh Neo4j installation only contains an empty `neo4j` database. We can verify this by executing the command shown in Listing 11, which shows ten nodes and the corresponding edges in our graph database.

Remark: We intentionally limit the number of nodes to ten because otherwise the query may take quite some time to finish in the VM. Consequently, you may want to consider limiting the number of nodes for most of your queries.

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

Figure 4: Error message in the graphical user interface if you do not have sufficient privileges.

Listing 11: **`cypher-shell`** – Show ten nodes and the corresponding edges in our database.

```
1  neo4j@neo4j> MATCH (n)-[r]->(m) RETURN n,r,m LIMIT 10;
```

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 (for Neo4j version 5.8.0).

First, we must download the data from our Nextcloud[31]. Since the data is a dump of a database, we can simply import it directly into our `neo4j` database. For this (and other administrative) purpose(s), Neo4j provides a command-line tool named `neo4j-admin`[32], which can be used as shown in Listing 13 (note that `neo4j-admin` is called from the system command-line tool). First, however, we must stop the database daemon before we can import the data as depicted in Listing 12. To this end, we switch to the `root` user as our `dbtutorial` user does not have enough privileges (line 1). Then, we proceed by stopping the Neo4j daemon (line 2) and checking whether the Neo4j daemon has been stopped indeed (line 3).

Remark: The commands in Listing 13 are to be executed directly after the commands in Listing 12. The two listings are separated to highlight the two separate steps, i.e., (i) stopping the database and (ii) importing the data.

The import using the `neo4j-admin` tool must be executed by a user that has enough privileges to import data into the database. In case of Neo4j, a dedicated user named `neo4j` has been created during installation that owes these privileges, and we can switch to this user as shown in line 1 of Listing 13. Line 2 shows the actual import of the database dump: We tell to `neo4j-admin` tool to `load` a `database` and to overwrite any existing database with the same name (`--overwrite-destination true`). Moreover, we tell the `neo4j-admin` tool to read the database dump from standard input (`--from-stdin`), which means that we must provide the file at the end of the command using the < pipeline. This pipeline operator of the Linux terminal directly redirects the content of the right-hand file to the command on the left-hand side of the operator, i.e., `neo4j-admin ... < movies-50.dump` tells the `neo4j-admin` tool to directly read to content of the file `movies-50.dump`. Right before this operator, we finalize the arguments to the `neo4j-admin` tool by providing the name of the database the given file should be imported to. Subsequently, we log back to our `root` user by entering the `exit` command (line 3), restart the Neo4j daemon once again (line 4), and verify that the Neo4j daemon is running again (line 5). Finally, we log back to our `dbtutorial` user by entering the `exit` command a second time (line 14).

Listing 12: **`terminal`** – Stop the Neo4j daemon.

```
1  dbtutorial@database-tutorial:~# su -
2  root@database-tutorial:~# systemctl stop neo4j
```

[31]Data for this assignment: `https://kitten.cosy.sbg.ac.at/index.php/s/CJAyYHqwswYoZYW/download`
[32]The `neo4j-admin` tool: `https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/`

```
root@database-tutorial:~# systemctl status neo4j
• neo4j.service - Neo4j Graph Database
    Loaded: loaded (/lib/systemd/system/neo4j.service; disabled; vendor ...)
    Active: inactive (dead)
```

Listing 13: **terminal** – Import a database dump into Neo4j.

```
root@database-tutorial:~# su - neo4j
neo4j@database-tutorial:~# neo4j-admin database load --overwrite-destination true
 --from-stdin neo4j < movies-50.dump
neo4j@database-tutorial:~# exit
root@database-tutorial:~# systemctl restart neo4j
root@database-tutorial:~# systemctl status neo4j
• neo4j.service - Neo4j Graph Database
    Loaded: loaded (/lib/systemd/system/neo4j.service; disabled; vendor ...)
    Active: active (running) since Tue 2023-05-02 13:12:12 CEST; 5s ago
  Main PID: 57756 (java)
    Memory: 305.7M
       CPU: 5.330s
    CGroup: /system.slice/neo4j.service
            ÀÄ 57756 /usr/bin/java -cp /var/lib/neo4j/plugins:...
root@database-tutorial:~# exit
```

Remark: The `--from-stdin` option of `neo4j-admin` also works with full paths, for example, `neo4j-admin ... --from-stdin neo4j < /home/dbtutorial/movies-50.dump`. Therefore, please ensure that you are using the correct path to the `movies-50.dump` file. **Hint:** If you unzipped the `assignment3-data.zip` as user `dbtutorial`, the `movies-50.dump` file will be located in the home directory of the `dbtutorial` user, i.e., `/home/dbtutorial/movies-50.dump`.

This imports the `movies` database dump into the `neo4j` database, and we can verify this by reconnecting to our database (using the `cypher-shell` or the graphical user interface) and executing the command shown in Listing 11 again. This time, Neo4j should report ten nodes and the corresponding edges in either a table format (cf. Figure 5) or – if you are using the graphical user interface – as a graph visualization (cf. Figure 6).

Word of Caution: Please use the `-overwrite-destination` option with caution in a production environment! As the name suggests, this option **overwrites** an existing database (in our case the `neo4j` database) and all data in this specific database is lost.

Remark: Neo4j may report different nodes and edges since we do not explicitly impose an order in our `MATCH` query.

```
neo4j@neo4j> MATCH (n)-[r]->(m) RETURN n,r,m LIMIT 10;
+------------------------------------------------------------------------------------------------------------------------------------------------------------+
| n                                               | r                                   | m                                                                                |
+------------------------------------------------------------------------------------------------------------------------------------------------------------+
| (:Person {name: "Keanu Reeves", born: 1964})      | [:ACTED_IN {roles: ["Neo"]}]          | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Carrie-Anne Moss", born: 1967})  | [:ACTED_IN {roles: ["Trinity"]}]      | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Laurence Fishburne", born: 1961}) | [:ACTED_IN {roles: ["Morpheus"]}]    | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Hugo Weaving", born: 1960})      | [:ACTED_IN {roles: ["Agent Smith"]}] | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Lilly Wachowski", born: 1967})   | [:DIRECTED]                          | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Lana Wachowski", born: 1965})    | [:DIRECTED]                          | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Joel Silver", born: 1952})       | [:PRODUCED]                          | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Emil Eifrem", born: 1978})       | [:ACTED_IN {roles: ["Emil"]}]         | (:Movie {tagline: "Welcome to the Real World", title: "The Matrix", released: 1999}) |
| (:Person {name: "Keanu Reeves", born: 1964})      | [:ACTED_IN {roles: ["Neo"]}]          | (:Movie {tagline: "Free your mind", title: "The Matrix Reloaded", released: 2003})   |
| (:Person {name: "Carrie-Anne Moss", born: 1967})  | [:ACTED_IN {roles: ["Trinity"]}]      | (:Movie {tagline: "Free your mind", title: "The Matrix Reloaded", released: 2003})   |
+------------------------------------------------------------------------------------------------------------------------------------------------------------+

10 rows
ready to start consuming query after 3 ms, results consumed after another 15 ms
```

Figure 5: Ten nodes and the corresponding edges in table format.

2.5 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

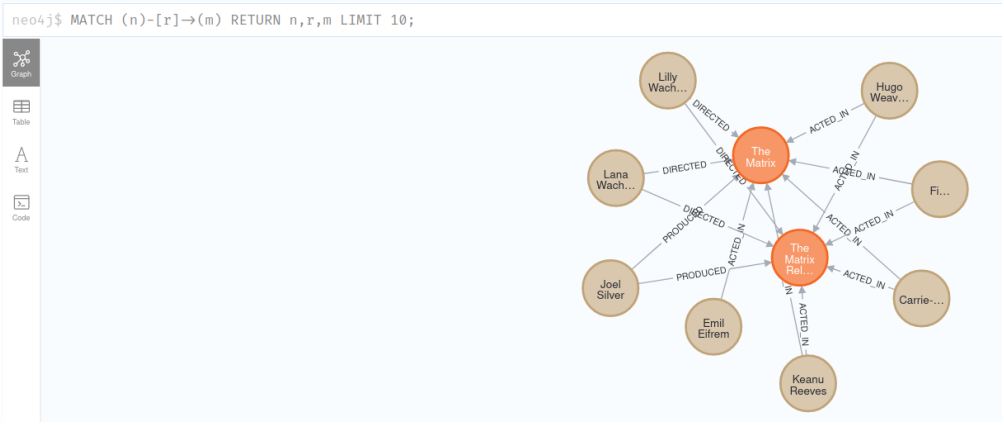

Figure 6: Ten nodes and the corresponding edges as graph visualization.

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[33]. In the following, we briefly cover some basics of the Cypher query language and provide **two (!)** queries, **Q1**–**Q2**, which 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 has been imported.

After connecting to the `neo4j` database, we can directly use CQL statements to query the data. The query shown in Listing 14, for example, selects all pairs of nodes that have some relationship (i.e., edge):

Listing 14: **`cypher-shell`** – Show all nodes and the corresponding edges in our database.

```
neo4j@neo4j> MATCH (n)-[r]->(m) RETURN n,r,m;
```

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

In Listing 11, we have already seen that we can use the `LIMIT` keyword to restrict the number of nodes that are reported (with the corresponding edges connecting them).

In this assignment, you are only given two queries in the Cypher query language, namely **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**, please study the **Updating the Data** section of the documentation[34] and either create a new relationship between two (or more) nodes or update an existing relationship between two (or more) nodes.

For query **Q4**, please study the **Getting the correct results** section of the documentation[35] 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

[33]Cypher query language: `https://neo4j.com/docs/getting-started/cypher-intro/`
[34]Updating the data: `https://neo4j.com/docs/getting-started/cypher-intro/updating/`
[35]Getting the correct results: `https://neo4j.com/docs/getting-started/cypher-intro/results/`

(reports a result or creates/updates an edge). Just try to explore the data and the Cypher query language. Interested students can also study and use more advanced CQL features (e.g., graph algorithms) for queries **Q3** and **Q4** in order to obtain Bonus points. However, creation/update (**Q3**) and filtering (**Q4**) are the minimum requirements.

Before we continue with **Q1** and **Q2**, we study another CQL query in detail (cf. Listing 15):

Listing 15: **cypher-shell** – Find all movies of person (actor) "Tom Hanks".

```
neo4j@neo4j> MATCH (p:Person { name: "Tom Hanks" })-[r:ACTED_IN]->(m:Movie) RETURN p,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: "Tom Hanks" }` is a property of a node that must be satisfied to match the pattern. Consequently, we only find movies of the person "Tom Hanks". For more details, we refer to the Neo4j documentation on Querying[36].

Now that we have learned about the basics of CQL, we give the two queries **Q1** and **Q2** ((cf. Listings 16– 17)) that can be executed out of the box by entering them into the `cypher-shell` (or the web interface; one after another). 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.

Listing 16: **cypher-shell** – Query **Q1**.

```
neo4j@neo4j>  MATCH (p1:Person { name: "Tom Hanks" })
              -[a1:ACTED_IN]->(m:Movie)<-[a2:ACTED_IN]-
              (p2:Person) RETURN p1,m,p2;
```

Listing 17: **cypher-shell** – Query **Q2**.

```
neo4j@neo4j>  MATCH v=shortestPath(
              (p1:Person { name: "Christian Bale" })-[*]-(p2:Person { name: "Halle Berry" })
              ) RETURN v;
```

In query **Q1**, we specify a pattern with three nodes and two `ACTED_IN` relationships (in opposite directions). Semantically, this identifies all persons that acted in a movie with "Tom Hanks".

Query **Q2** executes an operation that is common for graphs, namely a **shortest path** query[37]. 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 smallest 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 example, query **Q2** finds the degree of separation between the two persons (actors) "Christian Bale" and "Halle Berry", i.e., the connection between "Christian Bale" and "Halle Berry" in hops through other 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)[38] and

[36] `https://neo4j.com/docs/getting-started/cypher-intro/`
[37] Shortest path problem: `https://en.wikipedia.org/wiki/Shortest_path_problem`
[38] Updating the data: `https://neo4j.com/docs/getting-started/cypher-intro/updating/`

Q4 (apply a filter in a CQL query)[39] on your own by studying the Neo4j documentation and the data in the web-based graphical user interface of Neo4j.

Listing 18: Python3 template code to access our 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()

def 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"))
    driver.verify_connectivity()
  except:
    print("Unable to connect to {}".format(connection_string))

  try:
    # Create a Cypher query (result size limited to 10)
    cypher_query = '''
      MATCH (n:Person { name: "Tom Hanks" })-[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). execute_read 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.execute_read(get_result, 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)
  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()

if __name__ == "__main__":
  main()
```

[39] Getting the correct results: `https://neo4j.com/docs/getting-started/cypher-intro/results/`

2.6 Access the Data Using Python3

Similar to most database systems, Neo4j is typically accessed by an application. Therefore, the fourth part of this assignment is to write a small Python3 application. We recommend to use the `neo4j-driver` module for Python3 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. Your application should execute the queries **Q1**–**Q2** (given in Section 2.5) and **Q3**–**Q4** (introduced by your team), one after another, and print their results to the command line.

Remark: 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[40] and the usage of the `neo4j-driver`[41] module to access the Neo4j database. Nonetheless, we provide a minimum template code in Python3 that can be used as a starting point.

Hint: For your Python3 code, you may also want to consider the `execute_write` function[42], in particular if you modify the database. Please study the corresponding documentation and use the respective functions properly.

2.7 Questionnaire

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

3 Submission

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

Code

Remark: Please consider removing the database credentials (i.e., username and password) before you submit your code (in case you used them).

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. The code must print the results of *all* four queries **Q1**–**Q4** (one after another) when executed *as submitted*. We will not debug your code, for example, change some variable to make it work. Therefore, please double-check that your Python3 code works as expected and that *all* four queries are executed.

Questionnaire

Remark: The recommended formats are `.txt` and `.pdf`.

You can answer the questions directly in the text file `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`.

[40] `https://neo4j.com/docs/api/python-driver/current/` and `https://pypi.org/project/neo4j-driver/`
[41] `neo4j-driver` tutorial: `https://neo4j.com/docs/getting-started/languages-guides/neo4j-python/`
[42] The `execute_write` function: `https://neo4j.com/docs/api/python-driver/current/api.html#neo4j.Session.execute_write`

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/docs/getting-started/`
- 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/docs/getting-started/cypher-intro/` and `https://neo4j.com/docs/cypher-manual/current/introduction/`
- 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/docs/getting-started/languages-guides/neo4j-python/`
- An introduction to all Neo4j drivers (including the `neo4j-driver` Python module):
  `https://www.youtube.com/watch?v=JwgxkGY-36Q`
- One of the many introductions to the Linux terminal: `https://www.digitalocean.com/community/tutorials/an-introduction-to-the-linux-terminal`

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.

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 submission 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	