appendix
The Neo4j environment

In this book, you will learn graph theory and algorithms through practical examples using Neo4j. I chose Neo4j because I have more than five years of experience with it, building and analyzing graphs.

Neo4j is a native graph database, built from the ground up to store, query, and manipulate graph data. It is implemented in Java and accessible from software written in other languages using Cypher query language, through a transactional HTTP endpoint or the binary Bolt Protocol. In Neo4j, data is stored as nodes and relationships, which are both first-class citizens in the database. Nodes represent entities, such as people or businesses, and relationships represent the connections between these entities. Nodes and relationships can have properties, which are key–value pairs that provide additional information about the nodes and relationships.

Neo4j is designed to be highly scalable. It uses a flexible indexing system to efficiently query and manipulate data and supports atomicity, consistency, isolation, and durability (ACID) transactions to ensure data consistency. It also has a built-in query language, called Cypher, which is designed to be expressive and easy to use for querying and manipulating graph data.

Another benefit of using Neo4j is that it has two practical plugins you will be using:

- *The Awesome Procedures on Cypher (APOC) plugin*—A library of procedures, functions, and plugins for Neo4j that provide a wide range of capabilities, including data import/export, data transformation and manipulation, datetime interval processing, geospatial processing, text processing, and others.
- *The Graph Data Science (GDS) plugin*—A set of graph algorithms and procedures for Neo4j that allow users to perform advanced analytics on their graph data. GDS provides efficient, parallel implementations of common graph algorithms, such as shortest path, PageRank, and community detection. In addition, the plugin includes node embedding algorithms and machine learning workflows that support node classification and link prediction workflows.

A.1 Cypher query language

Cypher is a declarative query language for graph databases used to retrieve and manipulate data stored in a graph database. Cypher queries are written in a simple, human-readable syntax. Here is an example of a simple Cypher query that uses ASCII-art-style diagramming to illustrate the relationships being queried.

Listing A.1 A sample Cypher statement

```
MATCH (a:Person)-[:FOLLOWS]->(b:Person)
WHERE a.name = "Alice"
RETURN b.name
```

The openCypher initiative is a collaboration between Neo4j and several other organizations to promote the use of Cypher query language as a standard for working with graph data. The goal of the openCypher initiative is to create a common language that can be used to query any graph database, regardless of the underlying technology. To achieve this goal, the openCypher initiative is making the Cypher language specification and related resources available under an open source license and is encouraging the development of Cypher implementations by a variety of organizations. So far, the Cypher query language has been adopted by Amazon, Agens Graph, Katana Graph, Memgraph, RedisGraph, and SAP HANA (openCypher Implementers Group; https://opencypher.org/projects/).

There is also an official ISO project to propose a unified Graph Query Language (GQL) to interact with graph databases (GQL Standards Committee). The GQL aims to build on the foundation of SQL and integrate proven ideas from existing graph query languages, including Cypher. That makes learning Cypher a great start to interact with graph databases, as it is already integrated with many of them and will also be part of the official ISO Graph Query Language. Take a look at the graph pattern matching proposal for GQL (Deutsch et al., 2022) for more information.

A.2 Neo4j installation

There are a few different options to set up your Neo4j environment:

- Neo4j Desktop
- Neo4j Docker
- Neo4j Aura

I would advise you to use Neo4j Desktop if you are new to Neo4j.

A.2.1 Neo4j Desktop installation

Neo4j Desktop is a local Neo4j graph database management application. It allows you to create database instances and install official plugins with only a few clicks. If you decide to use Neo4j Desktop, follow these steps to successfully start a Neo4j database instance with installed APOC and GDS plugins:

1 Download the Neo4j Desktop application from the official website (https://neo4j.com/download; figure A.1).

Figure A.1 Download Neo4j Desktop.

2 Install the Neo4j Desktop application on your computer, and then open it.
3 Complete the registration step. You can enter the software key you were assigned when you downloaded the application or skip this step by clicking Register Later (figure A.2).

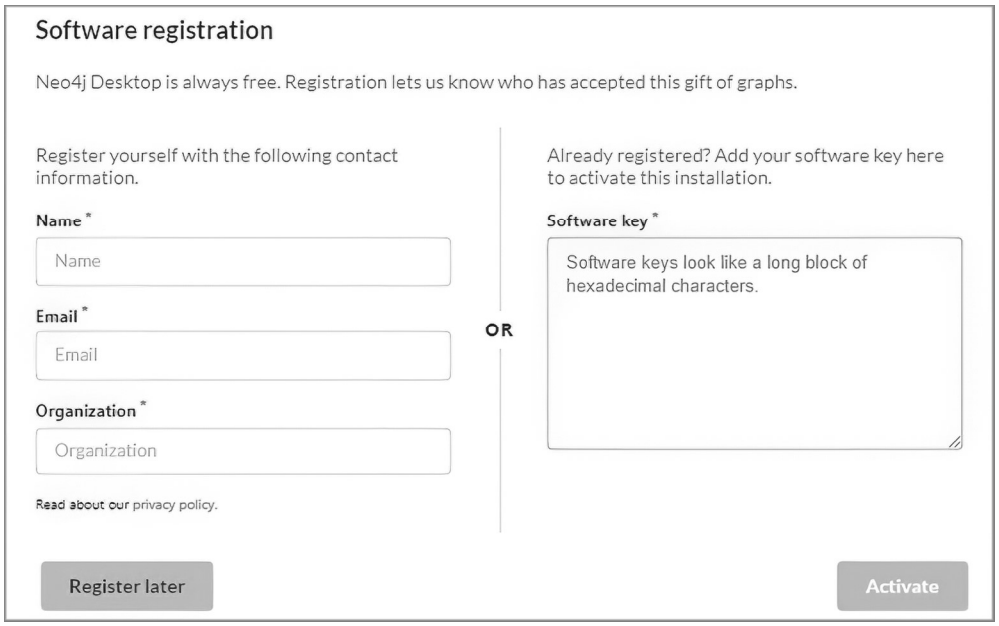

Figure A.2 Enter your personal information, or skip the registration step.

4 The Movie Database Management System (DBMS) is automatically started on the first execution of Neo4j Desktop. Stop the Movie DBMS if it is running (figure A.3).

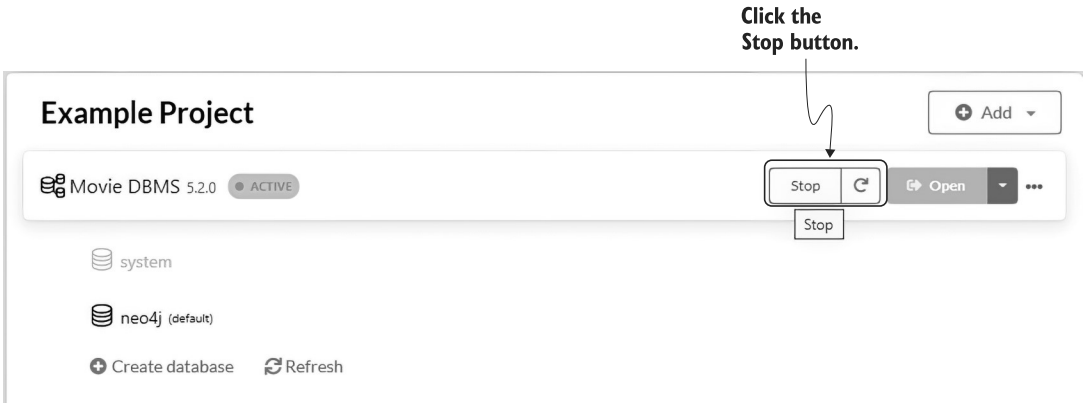

Figure A.3 Stop the default Movie DBMS database.

5 Add a new local DBMS (figure A.4).

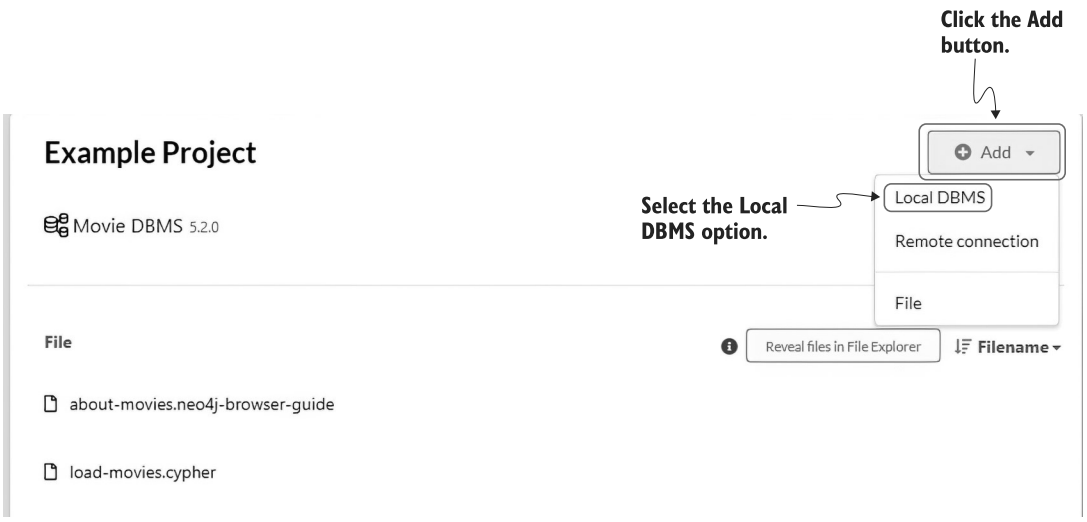

Figure A.4 Add a Local DBMS.

6 Type in any values for the DBMS name and password. Make sure to select version 5.12.0 or greater (figure A.5).

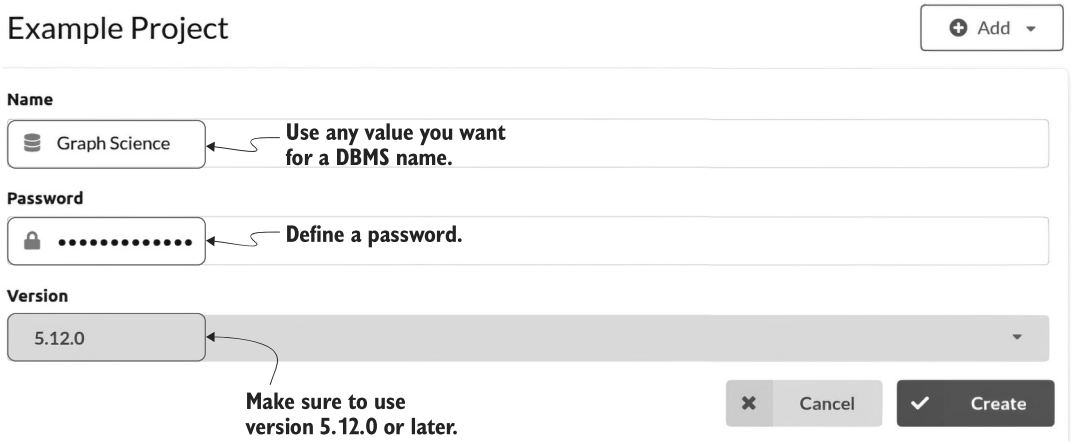

Figure A.5 Define a DBMS password and version.

7 Install the APOC and GDS plugins by selecting the DBMS, which opens a right-hand pane with Details, Plugins, and Upgrade tabs. Select the Plugins tab, and then install the APOC and GDS plugins (figure A.6).

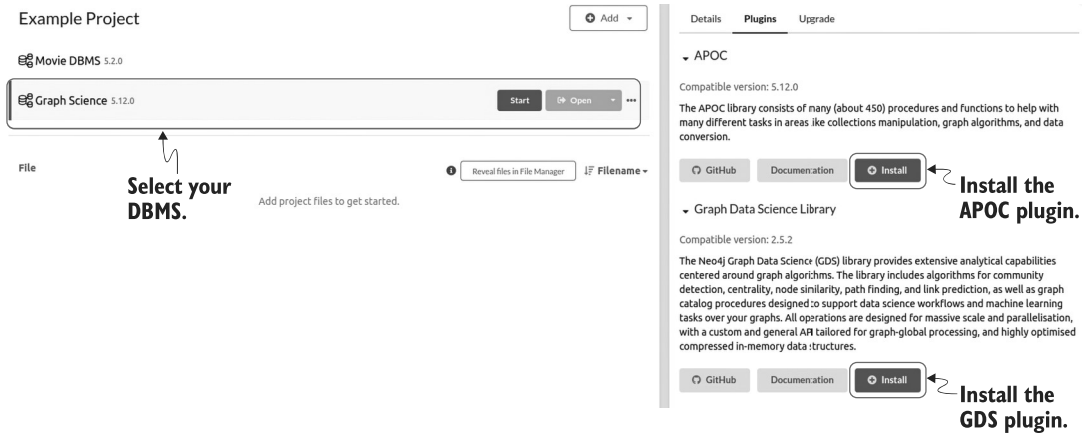

Figure A.6 Install the APOC and GDS plugins.

8 Start the database (figure A.7).
9 Open Neo4j Browser (figure A.8).

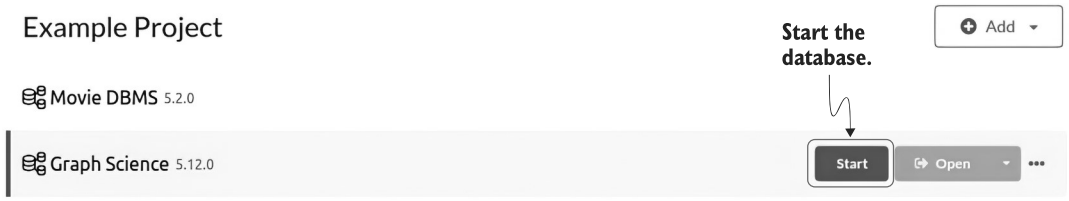

Figure A.7 Start the database.

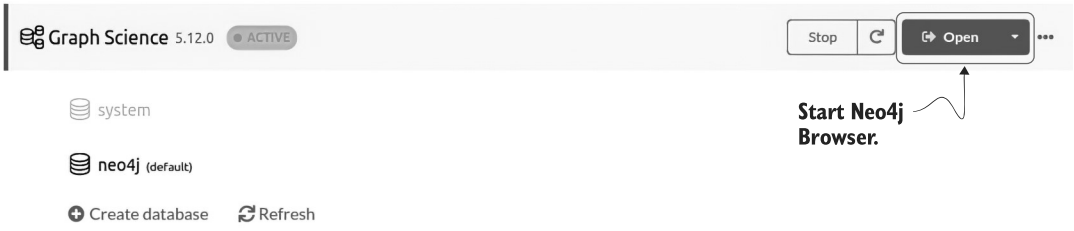

Figure A.8 Open Neo4j Browser.

10 Execute Cypher queries by typing them in the Cypher editor. For longer Cypher statements, you can use the full-screen editor option (figure A.9).

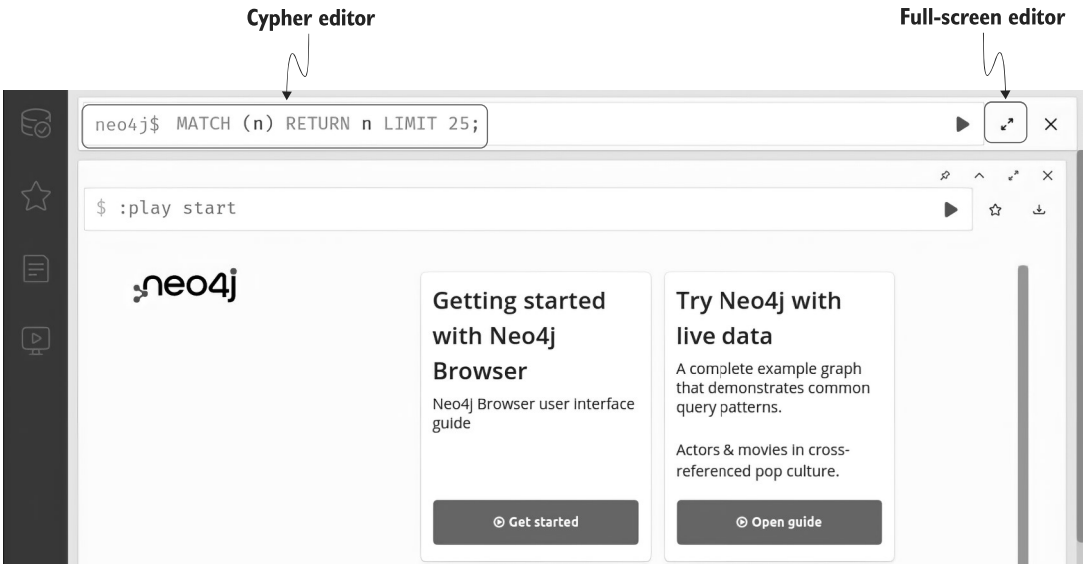

Figure A.9 The Cypher query editor in Neo4j Browser

A.2.2 Neo4j Docker installation

If you select the Neo4j Docker installation, you need to run the command in the following listing in your command prompt.

Listing A.2 Starting Neo4j Docker

```
docker run \
  -p 7474:7474 -p 7687:7687 \
  -d \
  -v $HOME/neo4j/data:/data \
  -e NEO4J_AUTH=neo4j/pleaseletmein \
  -e 'NEO4J_PLUGINS=["apoc", "graph-data-science"]' \
    neo4j:5.12.0
```

This command starts a dockerized Neo4j in the background. The APOC and GDS plugins are automatically added by defining the `NEO4J_PLUGINS` environment variable. It is a good practice to mount the `data` volume to persist the database files. The database username and password are specified with the `NEO4J_AUTH` variable.

Visit http://localhost:7474 in your web browser after you have executed the command in listing A.2. Type in the password, specified with the `NEO4J_AUTH` variable. The password in the example is `pleaseletmein`.

A.2.3 Neo4j Aura

Neo4j Aura is a hosted cloud instance of the Neo4j database. Unfortunately, the free version does not provide the GDS library. If you want to use cloud-hosted Neo4j Aura to follow the examples in this book, you will need to use the AuraDS version, which provides support for GDS algorithms. You can find more information on Neo4j's official website: https://neo4j.com/cloud/platform/aura-graph-database/.

A.3 Neo4j Browser configuration

Neo4j Browser has a beginner-friendly feature that visualizes all the relationships between resulting nodes, even when the relationships are not part of the query results. To avoid confusion, untick the Connect Result Nodes feature, as shown in figure A.10.

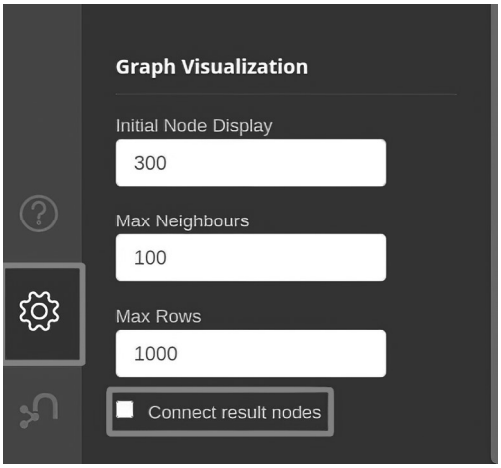

Figure A.10 Untick Connect Result Nodes in Neo4j Browser.