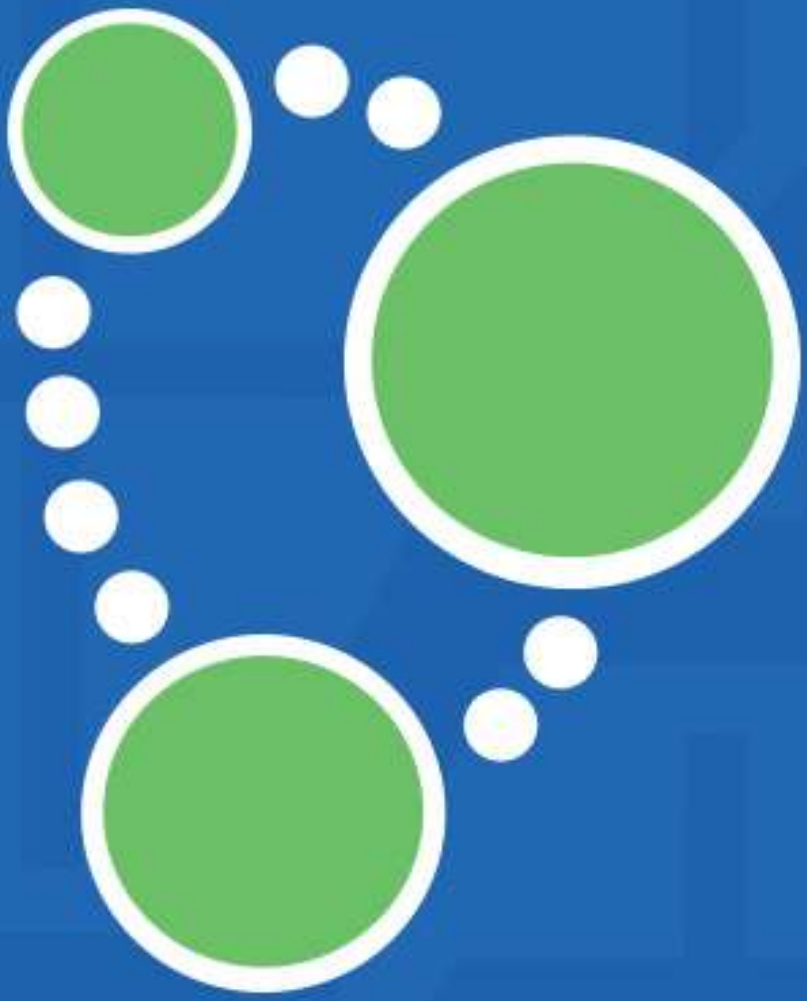


Neo4j

graph database

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Neo4j is one of the popular Graph Databases and Cypher Query Language (CQL). Neo4j is written in Java Language. This tutorial explains the basics of Neo4j, Java with Neo4j, and Spring DATA with Neo4j.

The tutorial is divided into sections such as Neo4j Introduction, Neo4j CQL, Neo4j CQL Functions, Neo4j Admin, etc. Each of these sections contain related topics with simple and useful examples.

Audience

This tutorial has been prepared for beginners to help them understand the basic to advanced concepts of Neo4j. It will give you enough understanding on Neo4j from where you can take yourself to a higher level of expertise.

Prerequisites

Before proceeding with this tutorial, you should have basic knowledge of Database, Graph Theory, Java, and Spring Framework.

Copyright & Disclaimer

Table of Contents

1. Neo4j – Overview

Neo4j is the world's leading open source Graph Database which is developed using Java technology. It is highly scalable and schema free (NoSQL).

What is a Graph Database?

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. It is composed of two elements - nodes (vertices) and relationships (edges).

Graph database is a database used to model the data in the form of graph. In here, the nodes of a graph depict the entities while the relationships depict the association of these nodes.

Popular Graph Databases

Neo4j is a popular Graph Database. Other Graph Databases are Oracle NoSQL Database, OrientDB, HypherGraphDB, GraphBase, InfiniteGraph, and AllegroGraph.

Why Graph Databases?

Nowadays, most of the data exists in the form of the relationship between different objects and more often, the relationship between the data is more valuable than the data itself.

Relational databases store highly structured data which have several records storing the same type of data so they can be used to store structured data and, they do not store the relationships between the data.

Unlike other databases, graph databases store relationships and connections as first-class entities.

The data model for graph databases is simpler compared to other databases and, they can be used with OLTP systems. They provide features like transactional integrity and operational availability.

RDBMS Vs Graph Database

Following is the table which compares Relational databases and Graph databases.

Sr. No.	RDBMS	Graph Database
1	Tables	Graphs

2	Rows	Nodes
3	Columns and Data	Properties and its values
4	Constraints	Relationships
5	Joins	Traversal

Advantages of Neo4j

Following are the advantages of Neo4j.

- **Flexible data model:** Neo4j provides a flexible simple and yet powerful data model, which can be easily changed according to the applications and industries.
- **Real-time insights:** Neo4j provides results based on real-time data.
- **High availability:** Neo4j is highly available for large enterprise real-time applications with transactional guarantees.
- **Connected and semi structures data:** Using Neo4j, you can easily represent connected and semi-structured data.
- **Easy retrieval:** Using Neo4j, you can not only represent but also easily retrieve (traverse/navigate) connected data faster when compared to other databases.
- **Cypher query language:** Neo4j provides a declarative query language to represent the graph visually, using an **ascii-art** syntax. The commands of this language are in human readable format and very easy to learn.
- **No joins**: Using Neo4j, it does NOT require complex joins to retrieve connected/related data as it is very easy to retrieve its adjacent node or relationship details without joins or indexes.

Features of Neo4j

Following are the notable features of Neo4j -

- **Data model (flexible schema):** Neo4j follows a data model named *native property graph model*. Here, the graph contains nodes (entities) and these nodes are connected with each other (depicted by relationships). Nodes and relationships store data in key-value pairs known as properties.

  In Neo4j, there is no need to follow a fixed schema. You can add or remove properties as per requirement. It also provides schema constraints.

- **ACID properties:** Neo4j supports full ACID (Atomicity, Consistency, Isolation, and Durability) rules.

- **Scalability and reliability:** You can scale the database by increasing the number of reads/writes, and the volume without effecting the query processing speed and data integrity. Neo4j also provides support for **replication** for data safety and reliability.

- **Cypher Query Language:** Neo4j provides a powerful declarative query language known as Cypher. It uses ASCII-art for depicting graphs. Cypher is easy to learn and can be used to create and retrieve relations between data without using the complex queries like Joins.

- **Built-in web application:** Neo4j provides a built-in **Neo4j Browser** web application. Using this, you can create and query your graph data.

- **Drivers:** Neo4j can work with –
    - REST API to work with programming languages such as Java, Spring, Scala etc.
    - Java Script to work with UI MVC frameworks such as Node JS.
    - It supports two kinds of Java API: Cypher API and Native Java API to develop Java applications.

  In addition to these, you can also work with other databases such as MongoDB, Cassandra, etc.

- **Indexing:** Neo4j supports Indexes by using Apache Lucence.

2. Neo4j – Data Model

Neo4j Property Graph Data Model

Neo4j Graph Database follows the Property Graph Model to store and manage its data.

Following are the key features of Property Graph Model:

- The model represents data in Nodes, Relationships and Properties
- Properties are key-value pairs
- Nodes are represented using circle and Relationships are represented using arrow keys
- Relationships have directions: Unidirectional and Bidirectional
- Each Relationship contains "Start Node" or "From Node" and "To Node" or "End Node"
- Both Nodes and Relationships contain properties
- Relationships connects nodes

In Property Graph Data Model, relationships should be directional. If we try to create relationships without direction, then it will throw an error message.

In Neo4j too, relationships should be directional. If we try to create relationships without direction, then Neo4j will throw an error message saying that "Relationships should be directional".

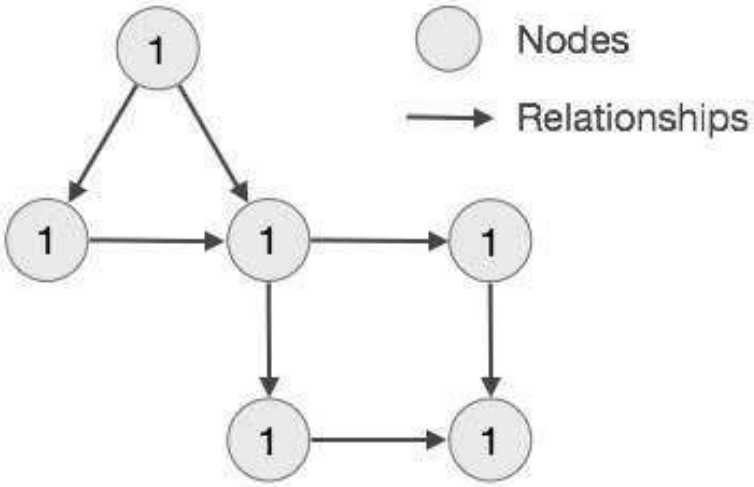
Neo4j Graph Database stores all of its data in Nodes and Relationships. We neither need any additional RRBMS Database nor any SQL database to store Neo4j database data. It stores its data in terms of Graphs in its native format.

Neo4j uses Native GPE (Graph Processing Engine) to work with its Native graph storage format.

The main building blocks of Graph DB Data Model are:

- Nodes
- Relationships
- Properties

Following is a simple example of a Property Graph.



Simple Graph

Here, we have represented Nodes using Circles. Relationships are represented using Arrows. Relationships are directional. We can represent Node's data in terms of Properties (key-value pairs). In this example, we have represented each Node's Id property within the Node's Circle.

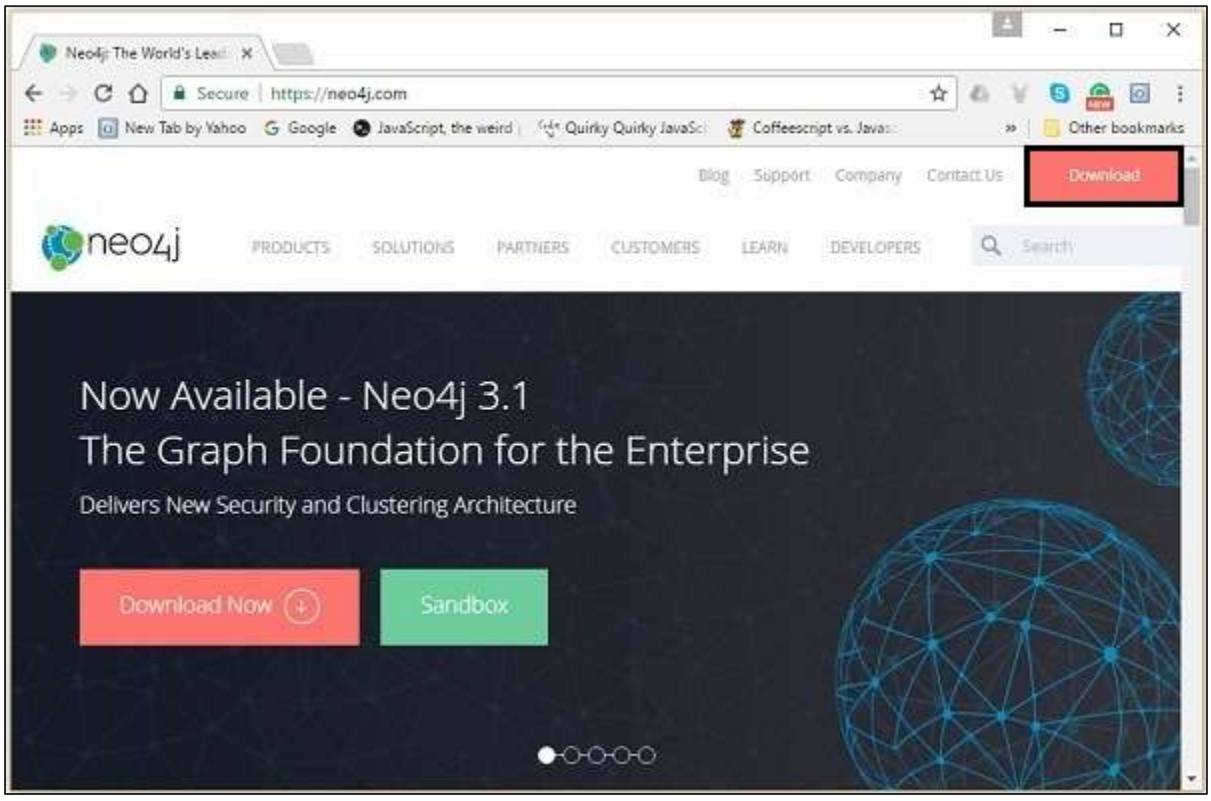
3. Neo4j – Environment Setup

In this chapter, we will discuss how to install Neo4j in your system using exe file.

Neo4j Database Server Setup with Windows exe File

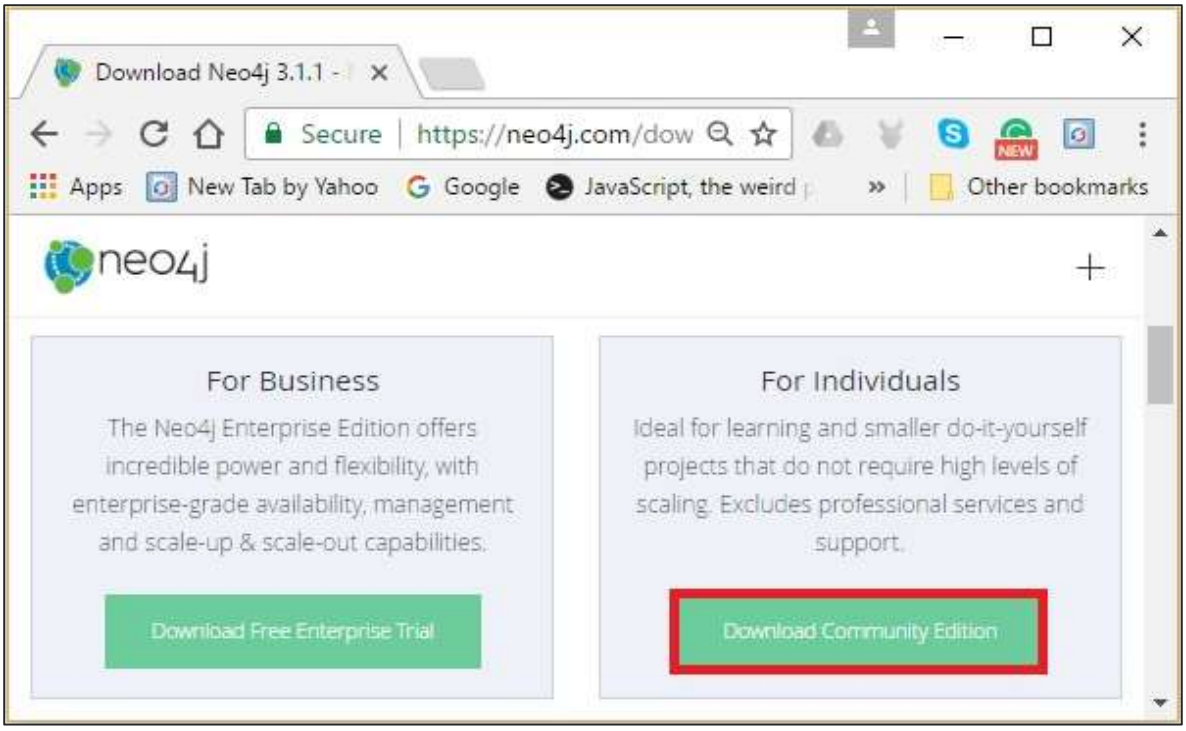
Follow the steps given below to download Neo4j into your system.

Step 1: Visit the Neo4j official site using https://neo4j.com/. On clicking, this link will take you to the homepage of neo4j website.

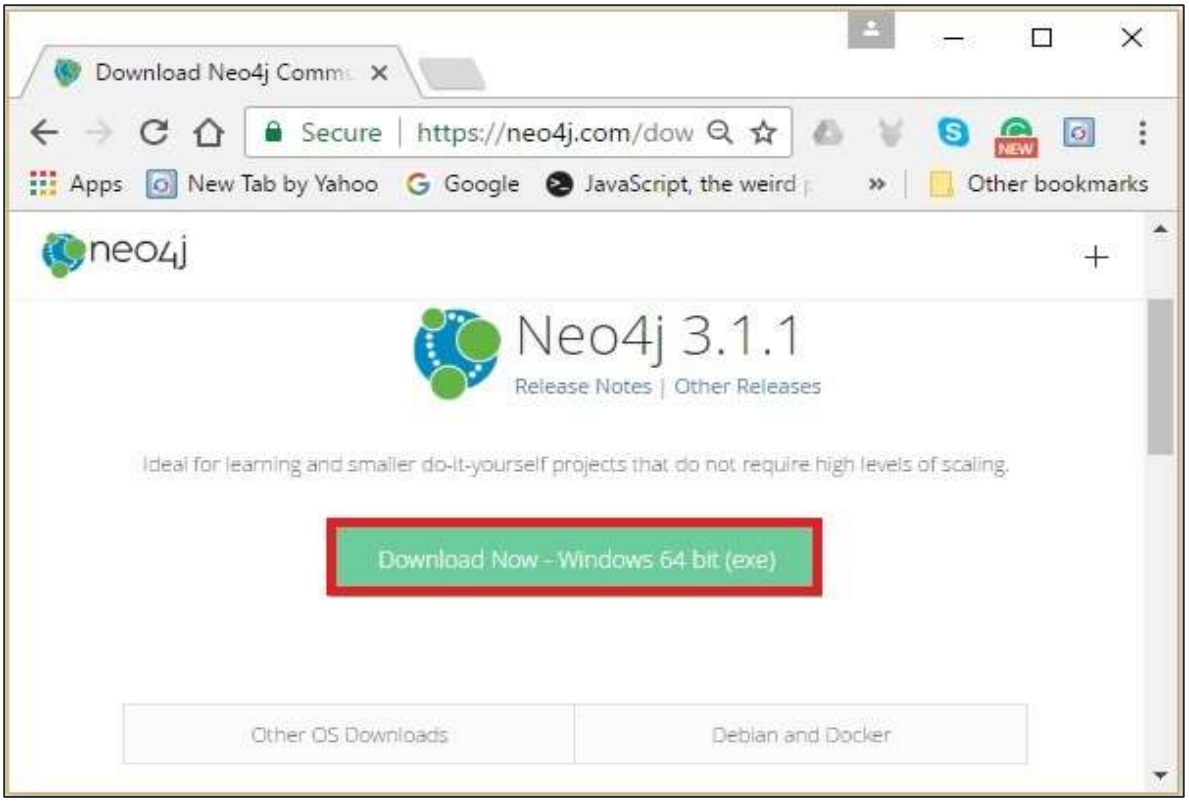


Step 2: As highlighted in the above screenshot, this page has a Download button on the top right hand side. Click it.

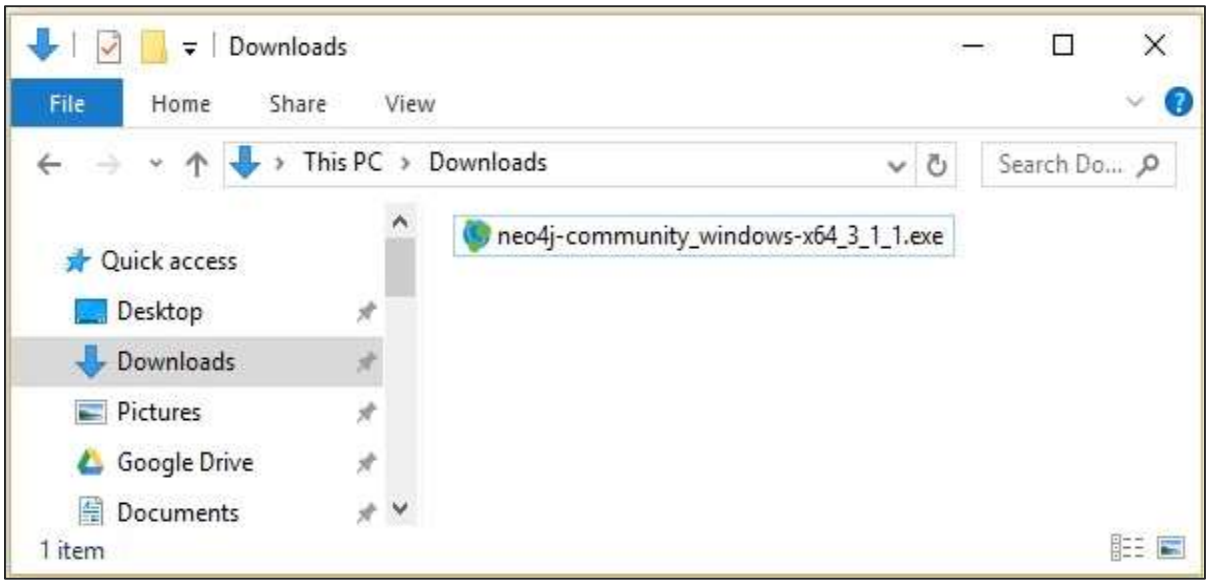
Step 3: This will redirect you to the downloads page, where you can download the community edition and the enterprise edition of Neo4j. Download the community edition of the software by clicking the respective button.



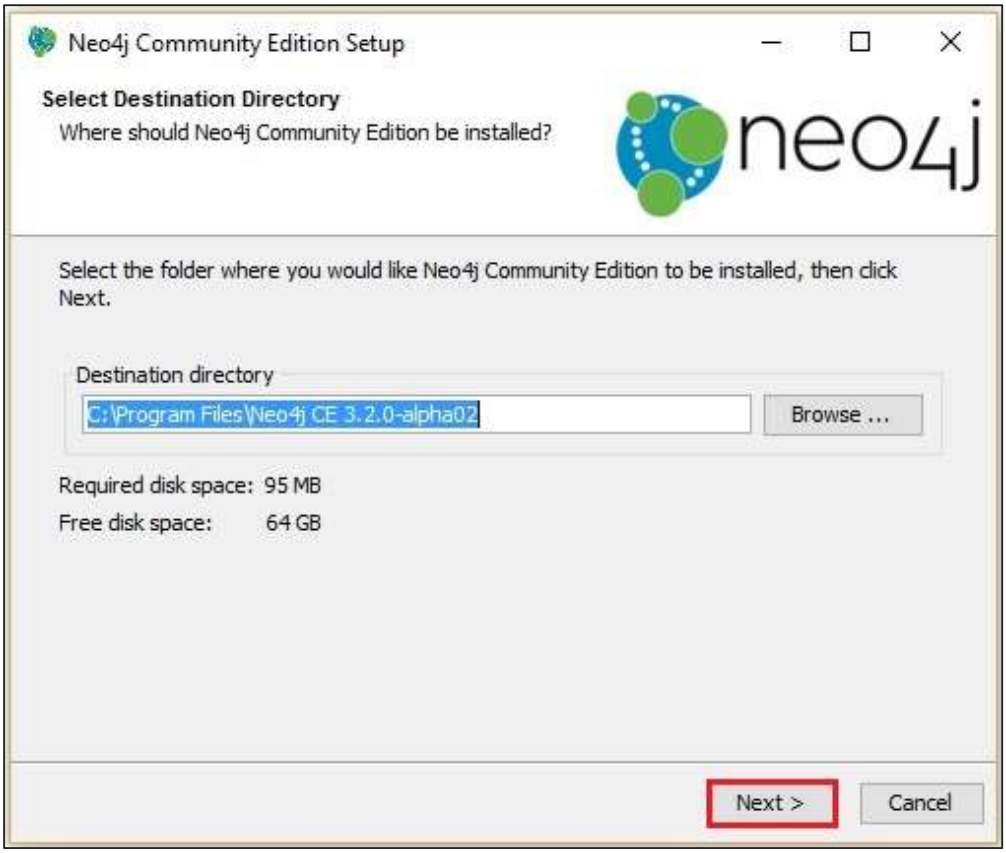
Step 4: This will take you to the page where you can download community version of Neo4j software compatible with different operating systems. Download the file respective to the desired operating system.



This will download a file named **neo4j-community_windows-x64_3_1_1.exe** to your system as shown in the following screenshot.



Step 5: Double-click the exe file to install Neo4j Server.



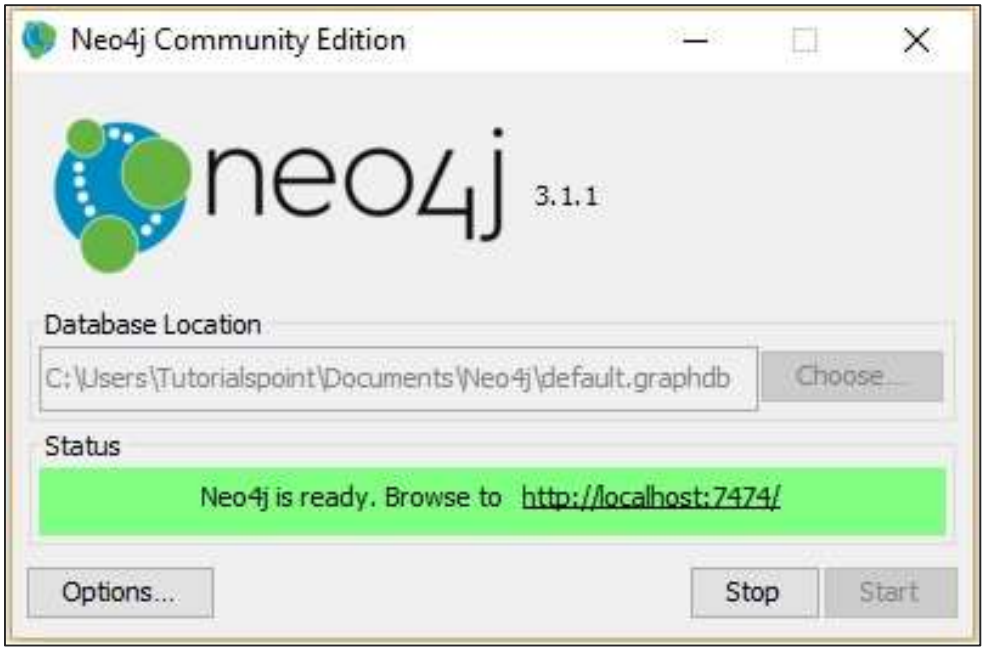
Step 6: Accept the license agreement and proceed with the installation. After completion of the process, you can observe that Neo4j is installed in your system.

Starting the Server

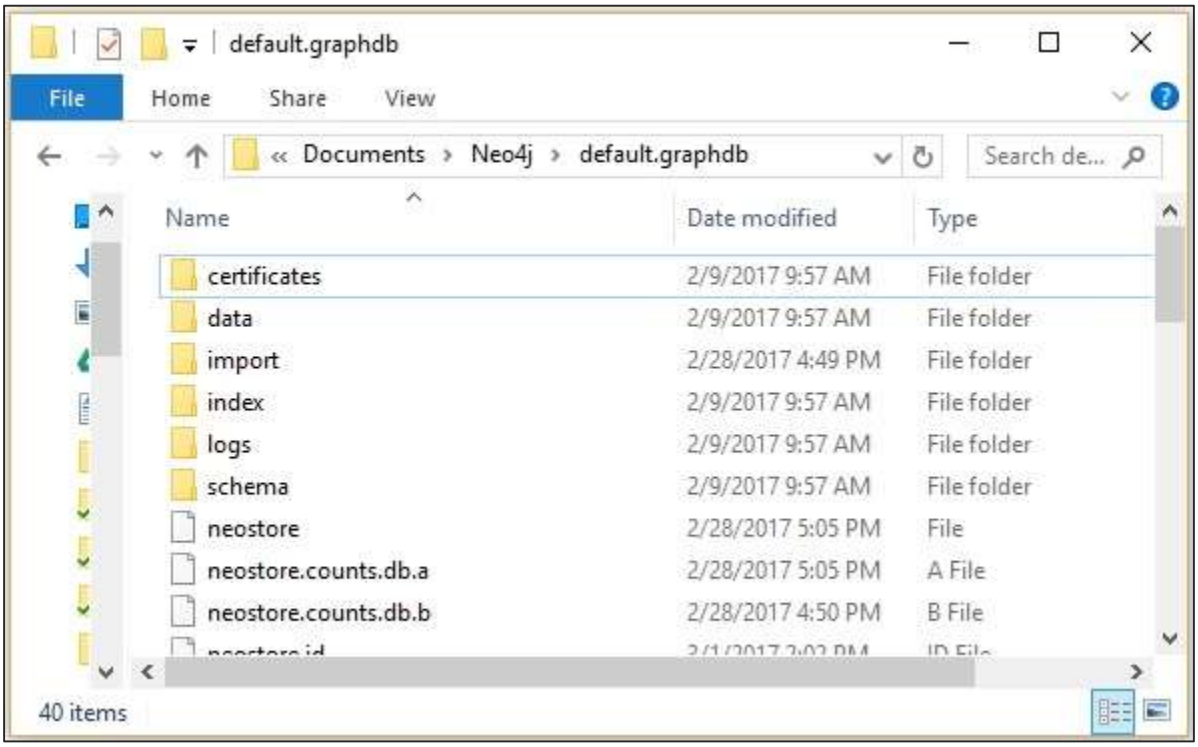
Step 1: Click the Windows startmenu and start the Neo4j server by clicking the start menu shortcut for Neo4j.

Step 2: On clicking the shortcut, you will get a window for Neo4j Community edition. By default, it selects c:\Users\[username]\Documents\Neo4j\default.graphdb. If you want, you can change your path to a different directory.

Step 3: Click the "Start" button to start the Neo4j server.

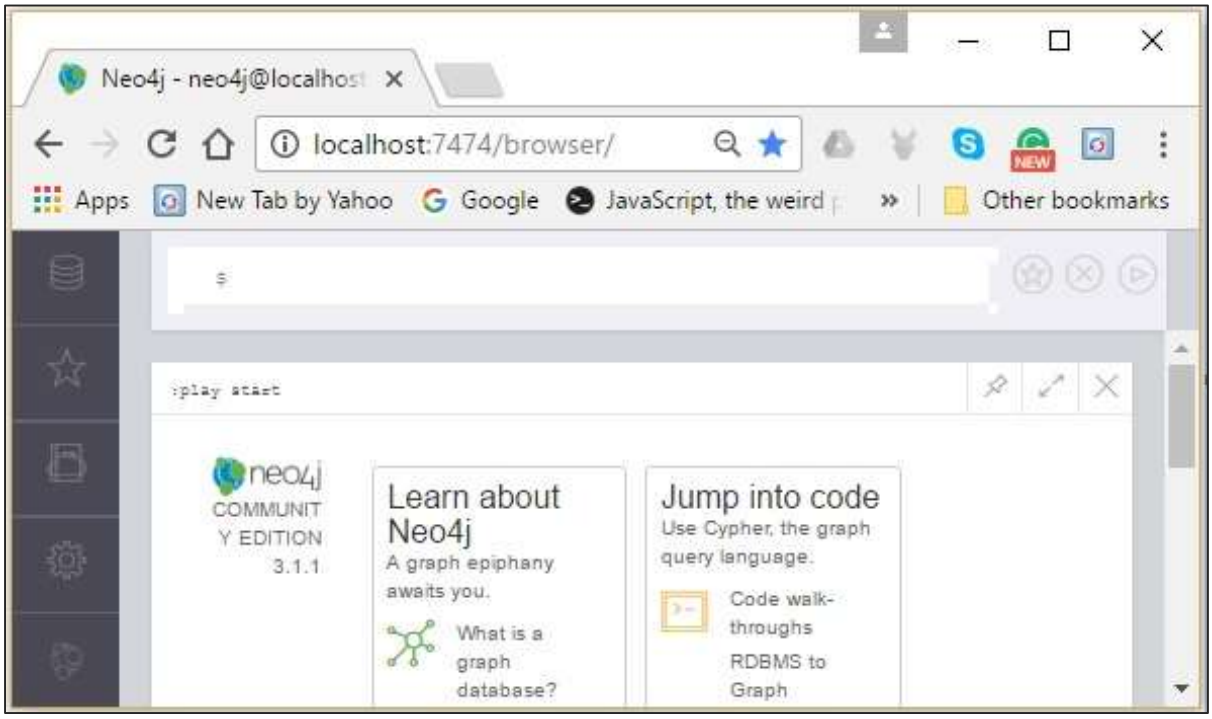


Once the server starts, you can observe that the database directory is populated as shown in the following screenshot.



Working with Neo4j

As discussed in the previous chapters, neo4j provides an in-built browse application to work with Neo4j. You can access Neo4j using the URL http://localhost:7474/



4. Neo4j – Building Blocks

Neo4j Graph Database has the following building blocks -

- Nodes
- Properties
- Relationships
- Labels
- Data Browser

Node

Node is a fundamental unit of a Graph. It contains properties with key-value pairs as shown in the following image.

Here, Node Name = "Employee" and it contains a set of properties as key-value pairs.

Properties

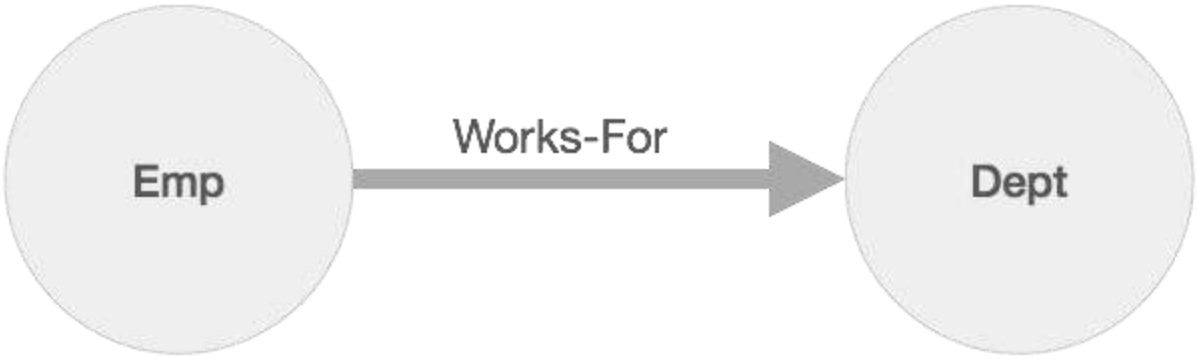
Property is a key-value pair to describe Graph Nodes and Relationships.

```
Key = Value
```

Where Key is a String and Value may be represented using any Neo4j Data types.

Relationships

Relationships are another major building block of a Graph Database. It connects two nodes as depicted in the following figure.



Here, Emp and Dept are two different nodes. "WORKS_FOR" is a relationship between Emp and Dept nodes.

As it denotes, the arrow mark from Emp to Dept, this relationship describes -

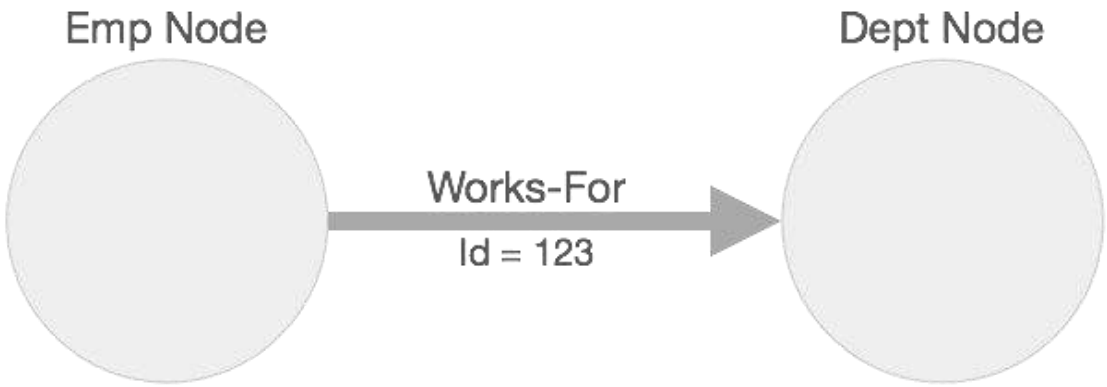
```
Emp WORKS_FOR Dept
```

Each relationship contains one start node and one end node.

Here, "Emp" is a start node, and "Dept" is an end node.

As this relationship arrow mark represents a relationship from "Emp" node to "Dept" node, this relationship is known as an "Incoming Relationship" to "Dept" Node and "Outgoing Relationship" to "Emp" node.

Like nodes, relationships also can contain properties as key-value pairs.



Here, "WORKS_FOR" relationship has one property as key-value pair.

```
Id=123
```

It represents an Id of this relationship.

Labels

Label associates a common name to a set of nodes or relationships. A node or relationship can contain one or more labels. We can create new labels to existing nodes or relationships. We can remove the existing labels from the existing nodes or relationships.

From the previous diagram, we can observe that there are two nodes.

Left side node has a Label: "Emp" and the right side node has a Label: "Dept".

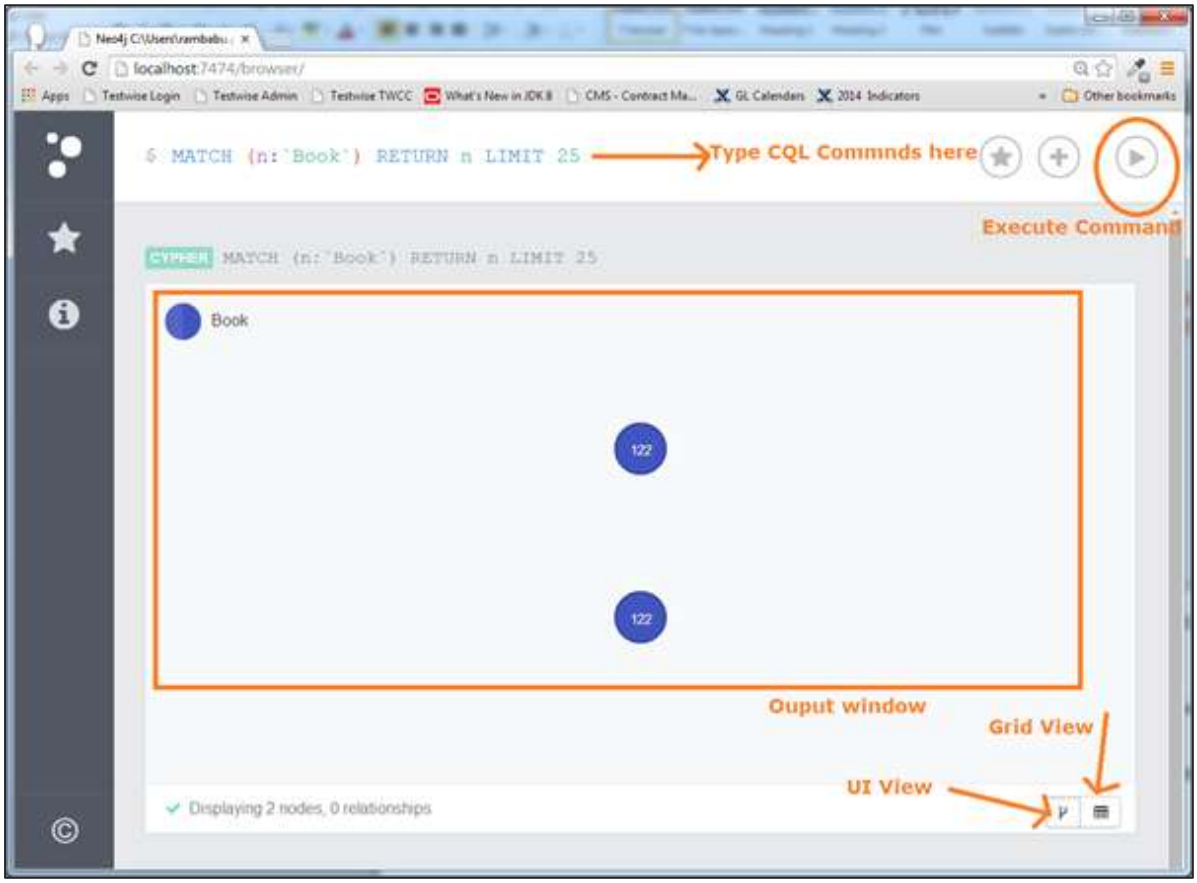
Relationship between those two nodes also has a Label: "WORKS_FOR".

Note: Neo4j stores data in Properties of Nodes or Relationships.

Neo4j Data Browser

Once we install Neo4j, we can access Neo4j Data Browser using the following URL

```
http://localhost:7474/browser/
```



Neo4j Data Browser is used to execute CQL commands and view the output.

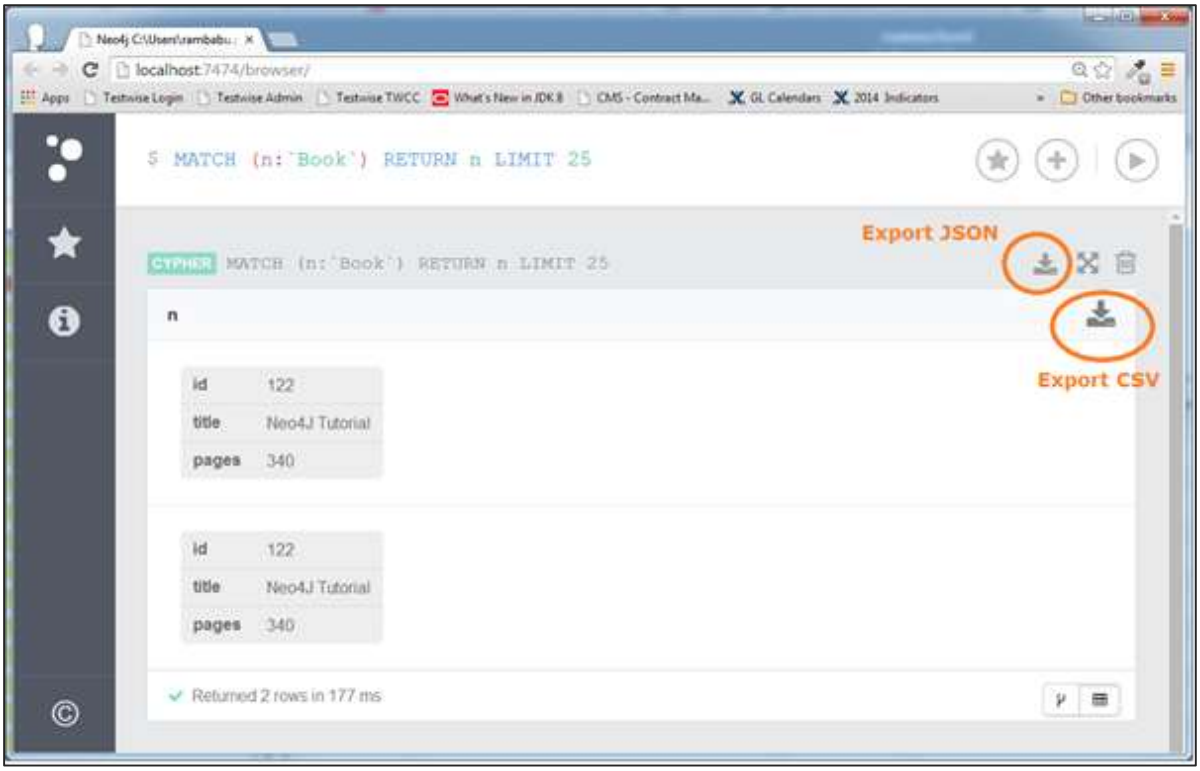
Here, we need to execute all CQL commands at dollar prompt: "**$**"

Type commands after the dollar symbol and click the "Execute" button to run your commands.

It interacts with Neo4j Database Server, retrieves and displays the results just below the dollar prompt.

Use "VI View" button to view the results in diagrams format. The above diagram shows results in "UI View" format.

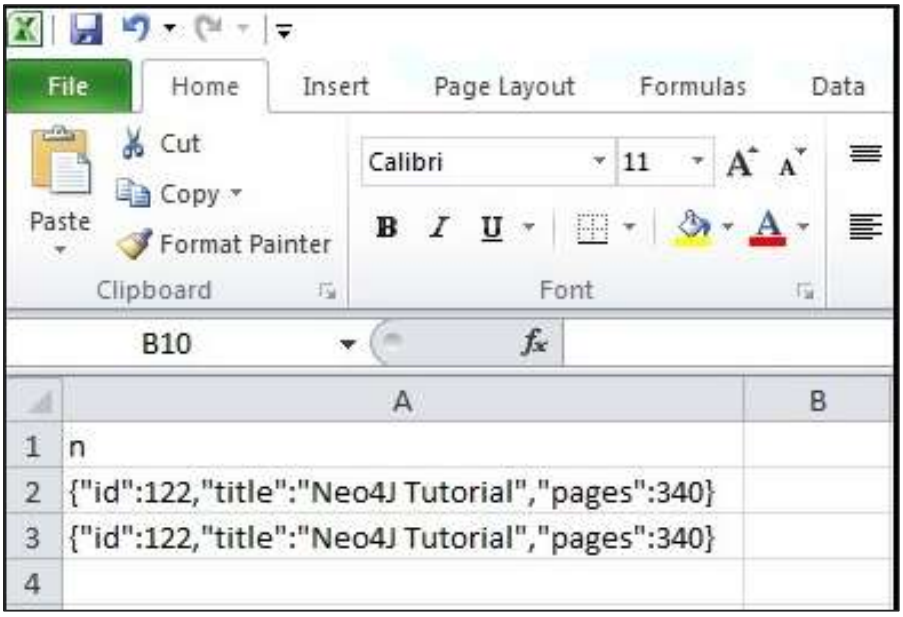
Use "Grid View" button to view the results in Grid View. The following diagram shows the same results in "Grid View" format.



When we use "Grid View" to view our Query results, we can export them into a file in two different formats.

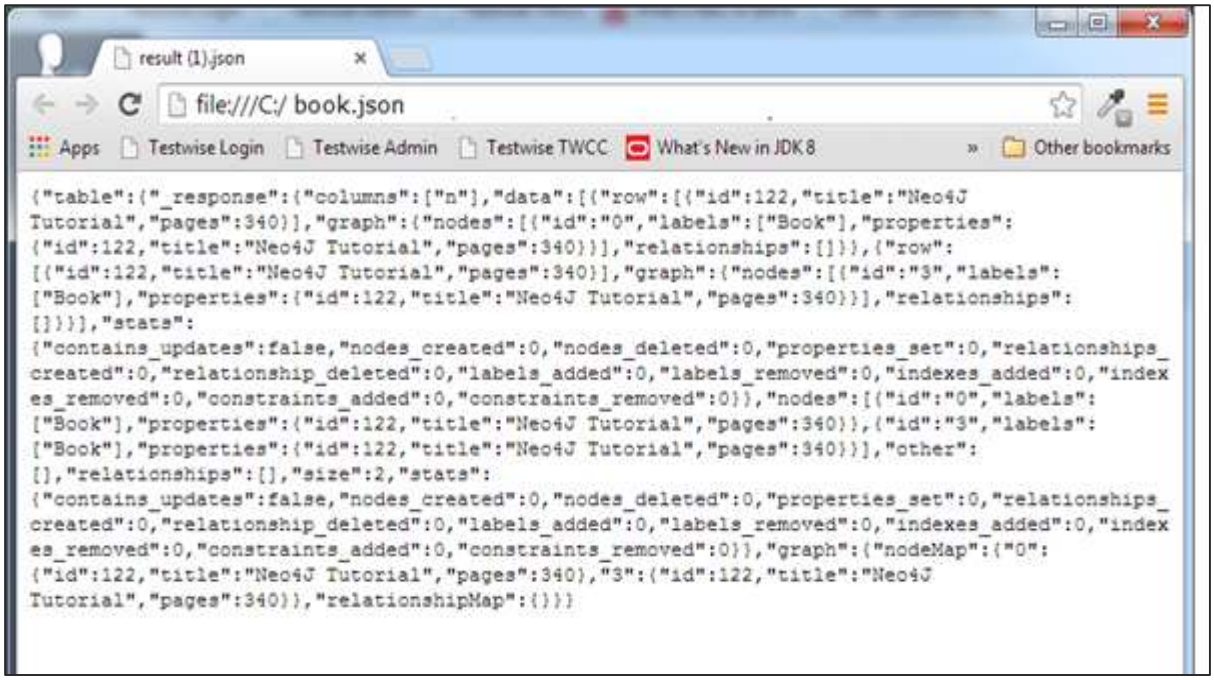
CSV

Click the "Export CSV" button to export the results in csv file format.



JSON

Click the "Export JSON" button to export the results in JSON file format.



However, if we use "UI View" to see our Query results, we can export them into a file in only one format: JSON

Neo4j – CQL

5. Neo4j CQL – Introduction

CQL stands for Cypher Query Language. Like Oracle Database has query language SQL, Neo4j has CQL as query language.

Neo4j CQL -

- Is a query language for Neo4j Graph Database.
- Is a declarative pattern-matching language.
- Follows SQL like syntax.
- Syntax is very simple and in human readable format.

Like Oracle SQL -

- Neo4j CQL has commands to perform Database operations.
- Neo4j CQL supports many clauses such as WHERE, ORDER BY, etc., to write very complex queries in an easy manner.
- Neo4j CQL supports some functions such as String, Aggregation. In addition to them, it also supports some Relationship Functions.

Neo4j CQL Clauses

Following are the read clauses of Neo4j **C**ypher **Q**uery **L**anguage:

Sr. No.	Read Clauses	Usage
1	MATCH	This clause is used to search the data with a specified pattern.
2	OPTIONAL MATCH	This is the same as match, the only difference being it can use nulls in case of missing parts of the pattern.
3	WHERE	This clause id is used to add contents to the CQL queries.
4	START	This clause is used to find the starting points through the legacy indexes.

5	LOAD CSV	This clause is used to import data from CSV files.

Following are the write clauses of Neo4j **C**ypher **Q**uery **L**anguage:

Sr. No.	Write Clauses	Usage
1	CREATE	This clause is used to create nodes, relationships, and properties.
2	MERGE	This clause verifies whether the specified pattern exists in the graph. If not, it creates the pattern.
3	SET	This clause is used to update labels on nodes, properties on nodes and relationships.
4	DELETE	This clause is used to delete nodes and relationships or paths etc. from the graph.
5	REMOVE	This clause is used to remove properties and elements from nodes and relationships.
6	FOREACH	This class is used to update the data within a list.
7	CREATE UNIQUE	Using the clauses CREATE and MATCH, you can get a unique pattern by matching the existing pattern and creating the missing one.
8	Importing CSV files with Cypher	Using Load CSV you can import data from .csv files.

Following are the general clauses of Neo4j **C**ypher **Q**uery **L**anguage:

Sr. No.	General Clauses	Usage
1	RETURN	This clause is used to define what to include in the query result set.
2	ORDER BY	This clause is used to arrange the output of a query in order. It is used along with the clauses **RETURN** or **WITH**.
3	LIMIT	This clause is used to limit the rows in the result to a specific value.

4	SKIP	This clause is used to define from which row to start including the rows in the output.
5	WITH	This clause is used to chain the query parts together.
6	UNWIND	This clause is used to expand a list into a sequence of rows.
7	UNION	This clause is used to combine the result of multiple queries.
8	CALL	This clause is used to invoke a procedure deployed in the database.

Neo4j CQL Functions

Following are the frequently used Neo4j CQL Functions:

Sr. No.	CQL Functions	Usage
1	String	They are used to work with String literals.
2	Aggregation	They are used to perform some aggregation operations on CQL Query results.
3	Relationship	They are used to get details of relationships such as startnode, endnode, etc.

We will discuss all Neo4j CQL commands, clauses and functions syntax, usage and examples in-detail in the subsequent chapters.