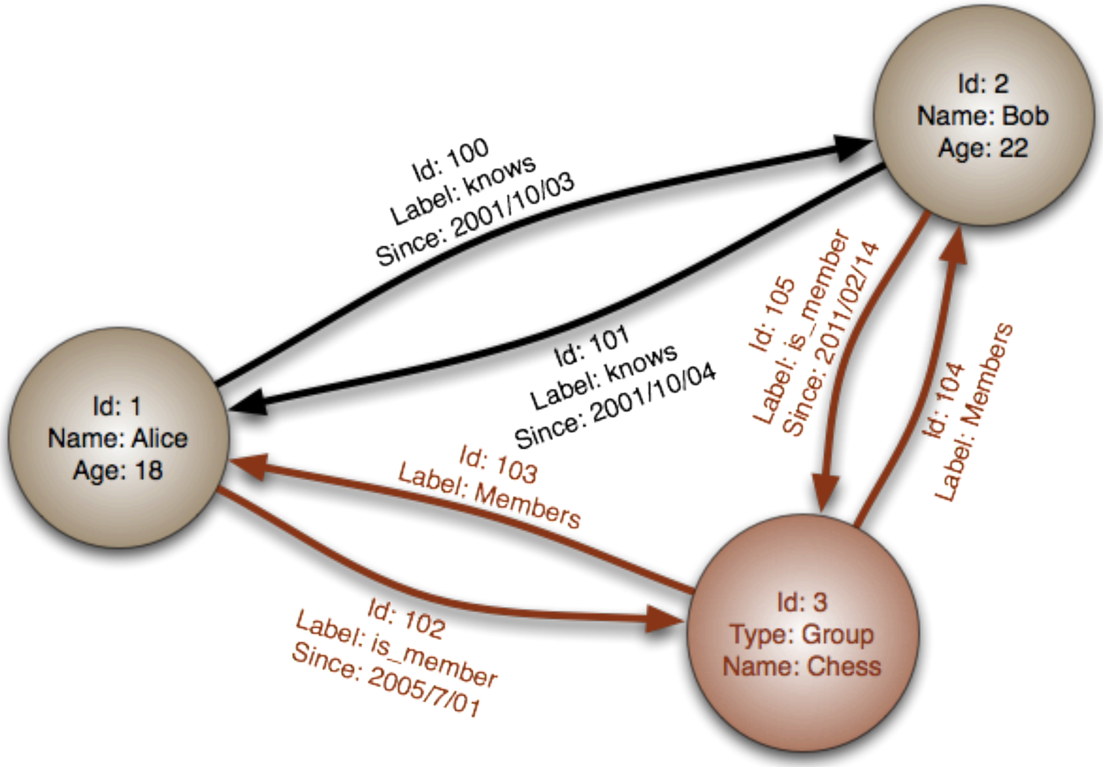
Graph Database

neo4j

Shreyas Kalrao 33255
Someshwar Gaikwad 33256
Sushant Sontakke 33257
Yash Rathi 33262

Introduction

- A **graph database** (**GDB**) is a database that uses graph structures for semantic queries with **nodes, edges, and properties** to represent and store data.
- Graph databases are a type of **NoSQL database**, created to address the limitations of relational databases.
- **Storage Mechanism**: Logical Table, Key-Valued or Document-Oriented Database.
- Retrieving data requires a **query language** other than SQL: like Gremlin, SPARQL, Cypher (Also APIs Available)



Graph databases employ nodes, properties, and edges.

Sales

Customer	Item	Time
0001	1A	20:34
0001	1A	21:15
0003	2A	21:16
0002	1A	21:16
0002	5C	

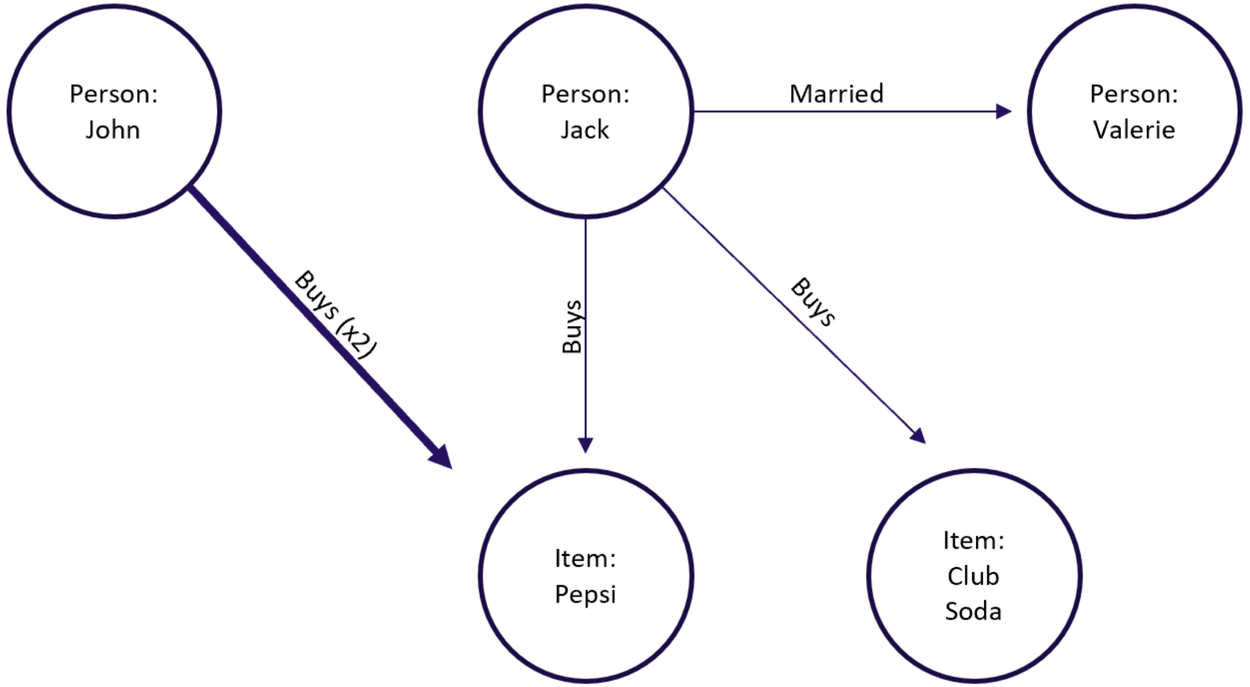
Inventory

Description	SKU
Pepsi	1A
Club Soda	2A
.	.
.	.
Diet Coke	5C

Customer

Name	CustID
John	0001
Jack	0002
Ted	0003
Ken	0004
Valerie	0005

Traditional database store data to efficiently store facts, but relationships must be rebuilt with JOINs and other inexact techniques.



Graph databases store both facts and the relationships between the facts, making certain types of analysis more intuitive.

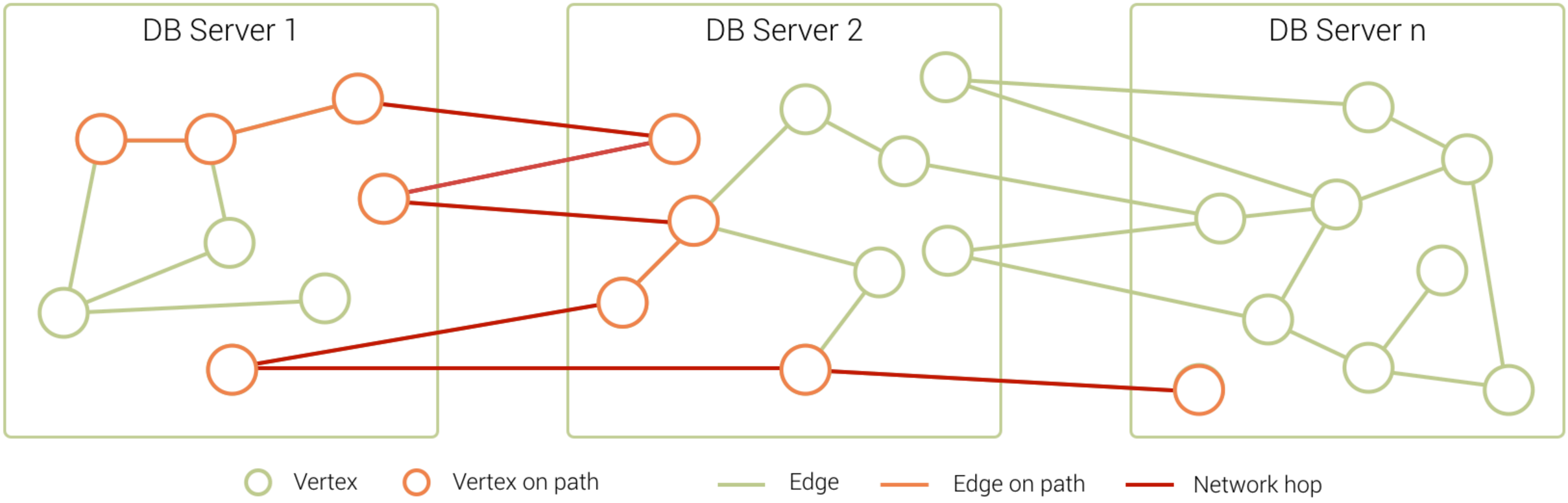
Why Use Them? Advantages over RDMS

- Most Important Difference: Nodes directly joined, Finding Related Data-> Follow Connections, whereas use Join-> more complex; Hence very fast response time
- Property: **Index-Free Adjacency**, Fast Queries (millions of nodes per second)
- Flexibility of a graph database->add new nodes and relationships without compromising your existing network.

What can go Wrong?

- Technology is Fairy New whereas RDBMS have been existing for generations
- On paper Graph Database sounds Powerful, but it is of no use if we are not able to implement them.
- Though it uses a Graph Structure, the underlying structure is that of a traditional table

Scalability



Characteristics

- **Performance**: They have superior performance for querying related data, big or small. A native graph has the so-called ***index-free adjacency*** property, where each vertex maintains its neighbor vertices information only, no global index about vertex connections exists.

- **Better Problem Solving**: Graph databases solve problems that are both impractical and practical for relational queries. Examples include iterative algorithms such as PageRank, gradient descent, and other data mining and machine learning algorithms.

- **AI Infrastructure**: Graph databases serve as great AI infrastructure due to well-structured relational information between entities, which allows one to further infer indirect facts and knowledge. Machine learning experts love them

Characteristics Continued…

- **Storage**: The underlying storage mechanism of graph databases can vary. Some depend on a relational engine and “store” the graph data in a table .Others use a key-value store or document-oriented database for storage, making them inherently NoSQL structures

- **Index-Free Adjacency**: Data lookup performance is dependent on the access speed from one particular node to another. Because index-free adjacency enforces the nodes to have direct physical RAM addresses and physically point to other adjacent nodes, it results in a fast retrieval.

Features of Neo4j

- **Cypher**, a declarative query language similar to SQL, but optimized for graphs. Now used by other databases like SAP HANA Graph and Redis graph via the **openCypher project**.

- **Constant time traversals** in big graphs for both depth and breadth due to efficient representation of nodes and relationships. Enables scale-up to billions of nodes on moderate hardware.

- **Flexible** property graph schema that can adapt over time, making it possible to materialize and add new relationships later to shortcut and speed up the domain data when the business needs change.

- **Drivers** for popular programming languages, including Java, JavaScript, .NET, Python, and many more.

Performance

- The more **connected** your data is, and the more **complex** your query is, the **faster** graph technology is at processing your requests.
- With **no need to translate requests** into non-graph methods or syntax, Neo4j delivers fast, accurate results for even complex queries of connected graph data.
- Relational approaches are appropriate for tabular data with static schemas, but not for the demands of **highly connected or changing datasets**.
- **Index-free adjacency** to traverse millions of data records with sub-second response times, even when those queries mine data several layers deep.
- Graph databases are **designed to traverse stored data** very quickly and retrieve results in milliseconds.
- Exceed performance expectations when traversing relationships among data for **data science and analytics** purposes where decision making is required.

Multi-Level Query Results: Neo4j vs. Relational

Query Depth	Requested Results	Approximate Results	**Neo4j** Execution Time	**Relational** Execution Time	Speed Advantage of Neo4j
Two levels	Friends of friends	2,500 records	0.01 seconds	0.016 seconds	**1.6** times faster
Three levels	Friends of friends of friends	110,000 records	0.17 seconds	30.27 seconds	**178** times faster
Four levels	Friends of friends of friends of friends	600,000 records	1.36 seconds	1,543 seconds	**1,135** times faster
Five levels	Friends of friends of friends of friends of friends	800,000 records	2.13 seconds	Query did not finish	**Infinitely** faster

Applications

The Neo4j graph database platform can be used for a wide variety of business applications, but there are many for which native graph technology is absolutely required to meet performance and reliability requirements. Some samples of those applications appear below.

Retail and E-Commerce (eg : Walmart, fortune 50)

- Purchase history recommendations
- Peer recommendations

Social and Business Networks (eg : LinkedIn, gamesys)

- Friendship and relationship sites
- Professional networks and organizations
- Event recommendation and review sites
- Membership and organization management

Information Management

- Identity and access management
- Master data management
- Network operations
- Internet of Things and sensor monitoring
- Support case solution

Applications Continued…

Customer and Sales Management (eg : eBay, medium.com)

- Prospect identification
- Product recommendation

Finance (eg : Citi Bank, UBS bank, Global 50 bank)

- Identity and access management
- Fraud detection
- Product and service recommendations

Healthcare (eg : Novartis)

- Care provider recommendations
- Clinical data analysis and bioinformatics
- Insurance fraud

Logistics (eg : Transparency One, Volvo, Caterpillar, Airbus)

- Real-time routing
- Supply-chain management and risk assessment

Installation of Neo4j

Download

- First, download a copy of Neo4j from the Neo4j download page. You can choose from either a free Enterprise Trial, or the free Community Edition.
- This tutorial uses the Community Edition.
- To download, simply click the Download button. Your download should start automatically.

Installation

- Once the file has downloaded, you can install Neo4j.
- The download screen includes step by step instructions on installing Neo4j on to your operating system.

Windows(exe)

1. Launch the installer you just downloaded. You might have to give the installer permission to make changes to your computer.
2. Follow the prompts, and choose the option to Run Neo4j.
3. Click on the Start button to start the Neo4j server.
4. Open the provided URL in your local web browser.
5. Change the password for the neo4j account.

Linux/UNIX Install (tar)

1. Open up your terminal/shell.

2. Extract the contents of the archive, using: tar -xf <filecode>.

3. For example,
   tar -xf neo4j-enterprise-2.3.1-unix.tar.gz the top level directory is referred to as NEO4J_HOME

4. Run Neo4j using $NEO4J_HOME/bin/neo4j console. Instead of ‘neo4j console’, you can use neo4j start to start the server process in the background.

5. Visit http://localhost:7474 in your web browser.

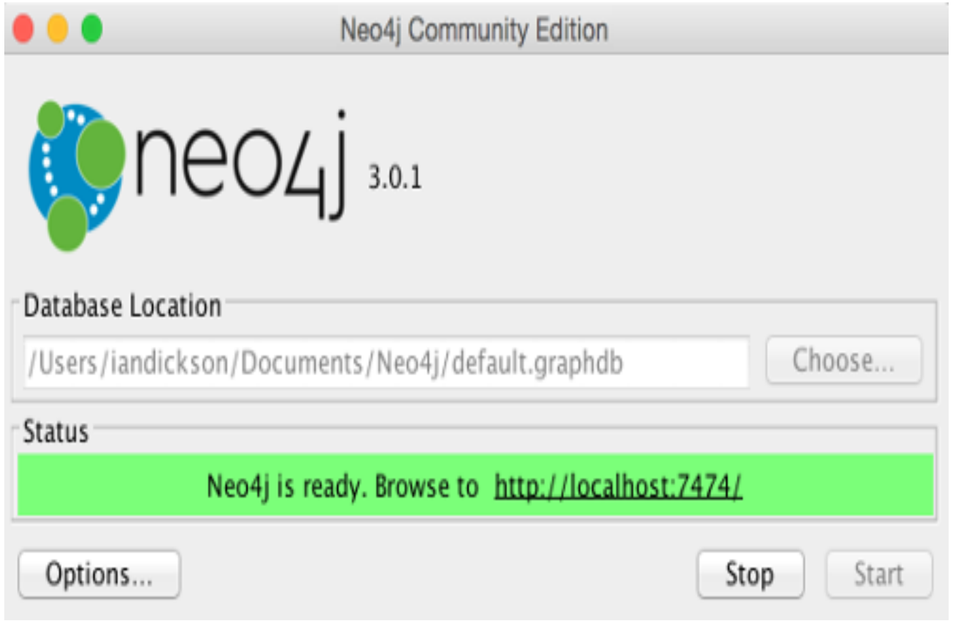
6. Change the password for the neo4j account.

Starting and Connecting to the Neo4j

1. Start the Server

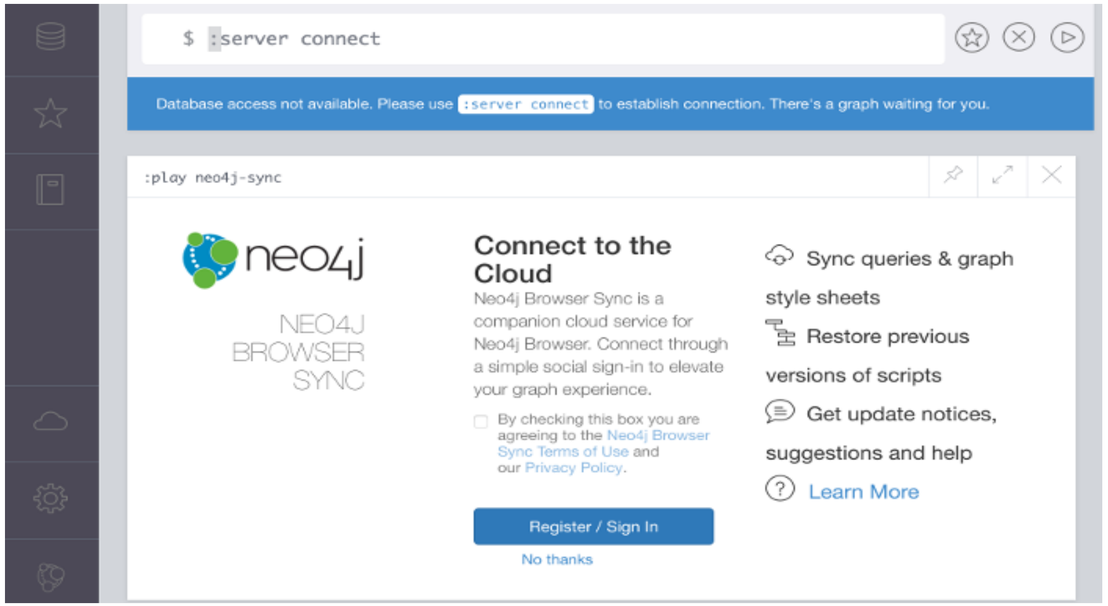
The installation instructions include instructions for starting the Neo4j server. The exact method you use will depend on your operating system.

For example, on a Mac, clicking `Neo4j Community Edition` from the `Applications` folder will launch this screen that allows you to start the Neo4j server.
Once the server has been started, open the following URL in your web browser: http://localhost:7474/ and follow the prompts.
Below are the screens I encountered when logging in for the first time (note that future versions may be different).



2.Connect to Neo4j

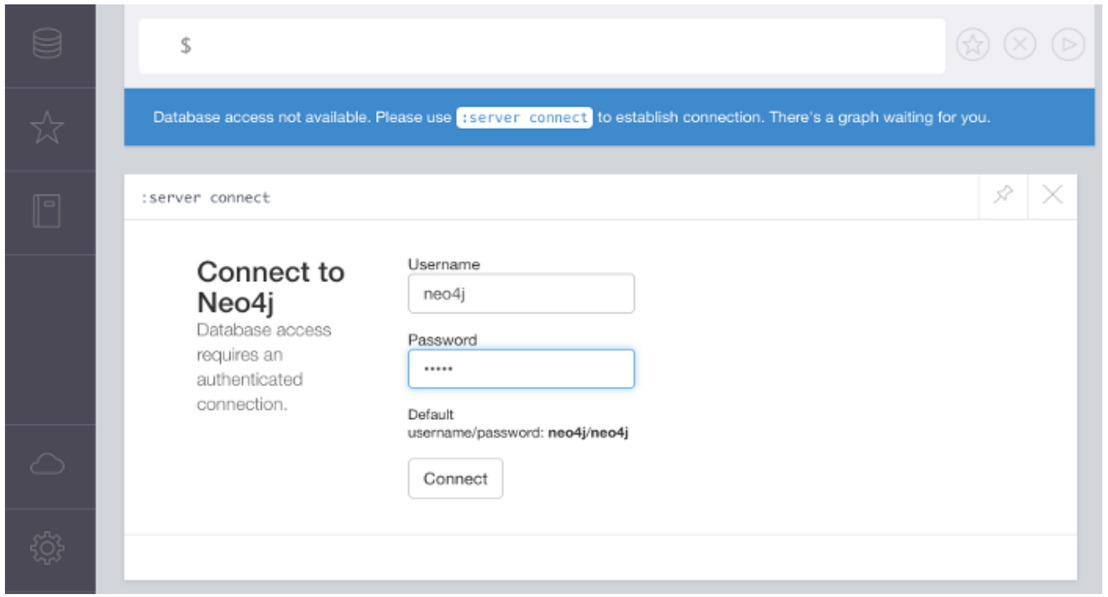
You can either create an account for the cloud service or click `No thanks`



3.Log In

Log in using the username and password provided on the screen.

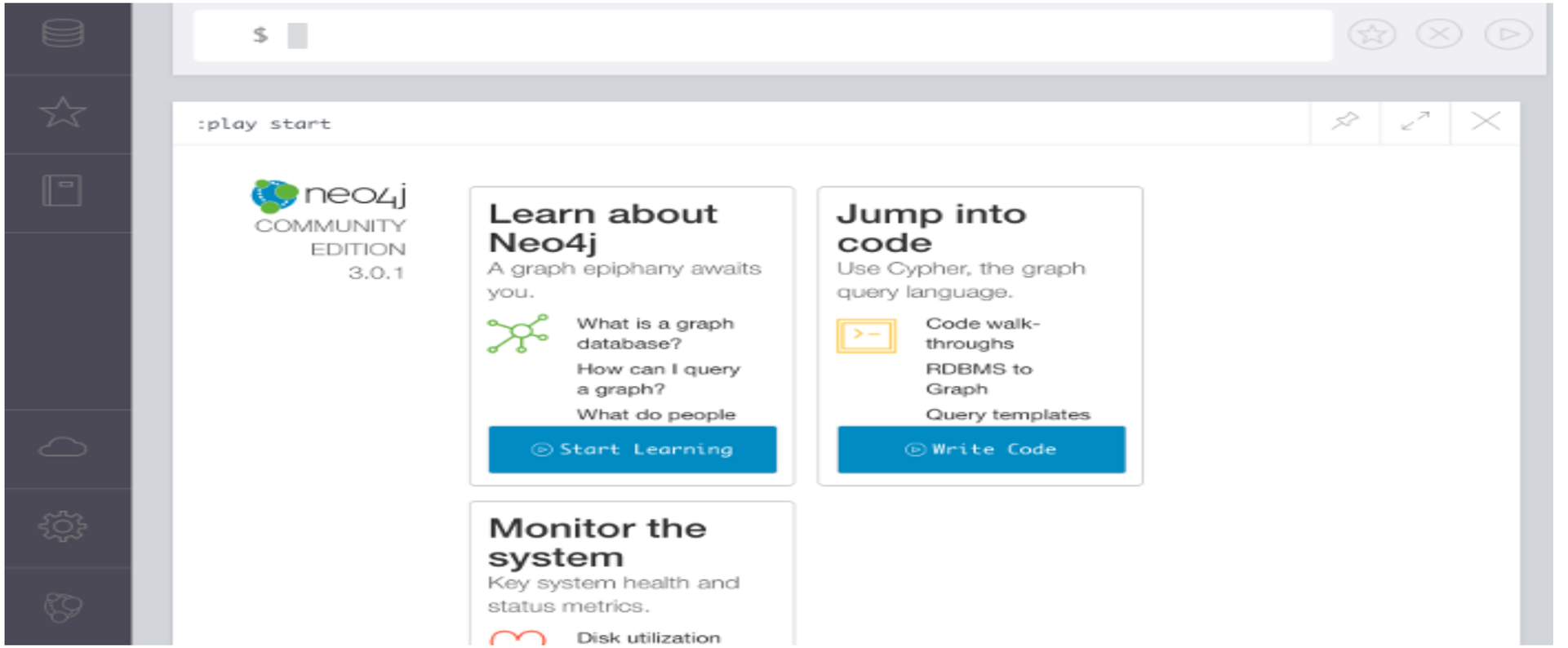
The first time you log in you will be prompted to change your password.



4.The Result

Once the password has been changed, this screen is displayed.

Here, you can use the links to learn more about Neo4j and how to create databases and run queries.



Operations in Neo4j

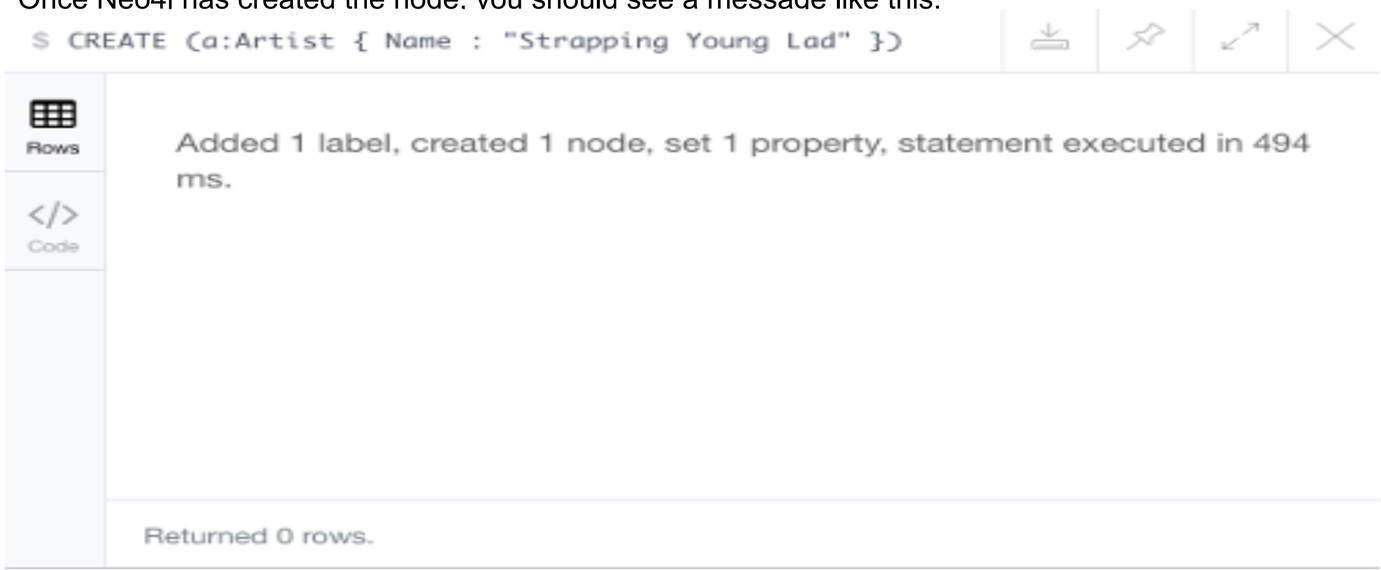
Neo4j has its own query language called Cypher. Cypher uses a similar syntax to SQL (Structured Query Language).

To create nodes and relationships using Cypher, use the CREATE statement.

Creating a Node:

```
CREATE (a:Artist { Name : "Strapping Young Lad" })
```

Once Neo4j has created the node, you should see a message like this:



Displaying the Node

```
CREATE (b:Album { Name : "Heavy as a Really Heavy Thing", Released : "1995" })
RETURN b
```

The CREATE statement *creates* the node but it doesn't *display* the node.
To display the node, you need to follow it up with a RETURN statement.

Creating Relation between Nodes

```
MATCH (a:Artist),(b:Album)
WHERE a.Name = "Strapping Young Lad" AND b.Name = "Heavy as a Really Heavy Thing"
CREATE (a)-[r:RELEASED]->(b)
RETURN r
```

It creates the following relationship:

References

- https://en.wikipedia.org/wiki/Graph_database
- https://neo4j.com/news/relational-vs-graph-databases/
- https://sdtimes.com/databases/guest-view-relational-vs-graph-databases-use/
- https://neo4j.com/blog/why-graph-databases-are-the-future/
- https://towardsdatascience.com/graph-databases-whats-the-big-deal-ec310b1bc0ed
- https://neo4j.com/blog/why-graph-databases-are-the-future/
- https://neo4j.com/customers/?ref=home
- https://en.wikipedia.org/wiki/Neo4j
- https://www.arangodb.com/why-arangodb/arangodb-enterprise/arangodb-enterprise-smart-graphs/
- https://dzone.com/articles/scaling-graph-databases