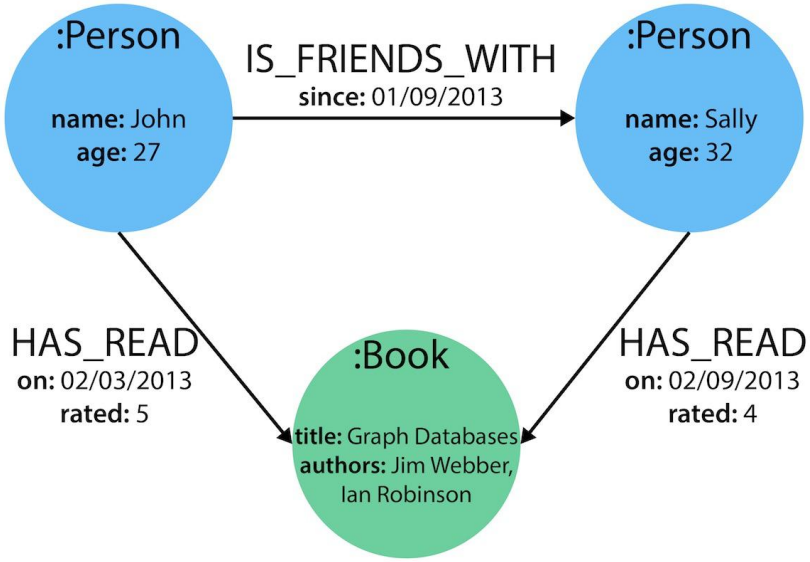
Neo4j Phase 2

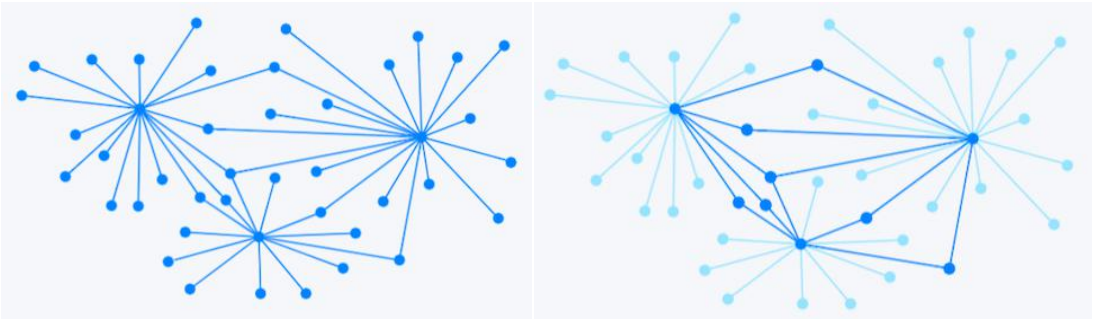
Introduction

Neo4j is a graph-based NoSQL system. Unlike traditional Relational DBMS like MySQL, Neo4j stores nodes and edges of graphs. Below shows a Neo4j database storing 2 Person nodes, a Book node, and edges labeled 'HAS_READ' and 'IS_FRIEND_WITH'. Each node contains multiple properties, just like a row has multiple columns in MySQL.



Motivations

1) Easily and intuitively store graph datasets (social networks, etc).
2) No need for joins, data is stored as it's connected.
3) Handle many-to-many relationships, handle deep hierarchies.
4) Find patterns and hidden connections of the data (generate recommendations).

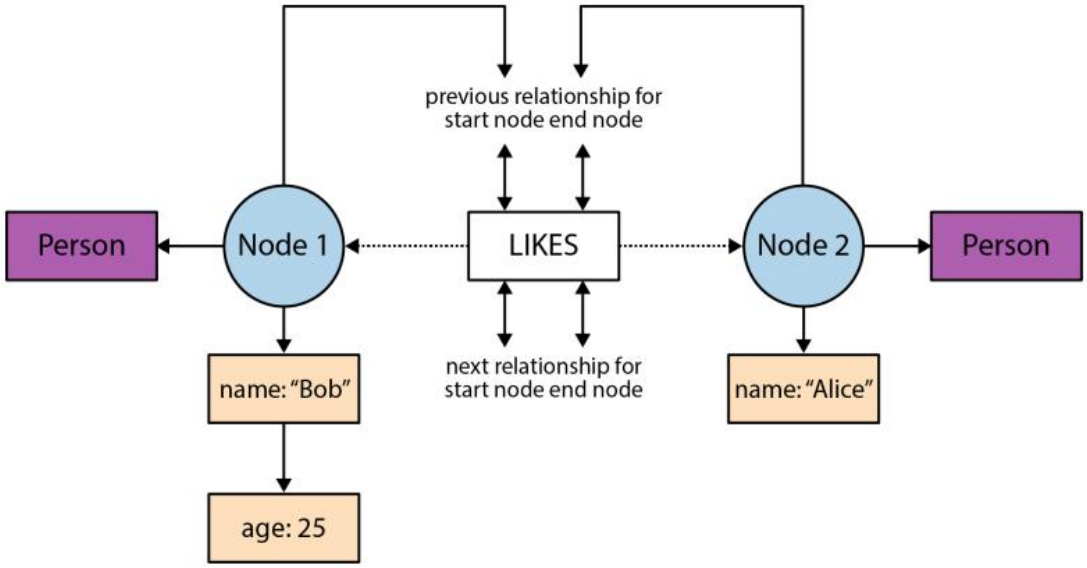


Characteristics

1) Fast query by graph query language, no more complex joins.
2) Scalable architecture to store huge amount of data.
3) ACID compliance to guarantee integrity of relationship-based queries.
4) Handful libraries for faster development in Java, Python, Node.js, etc.

Data Model

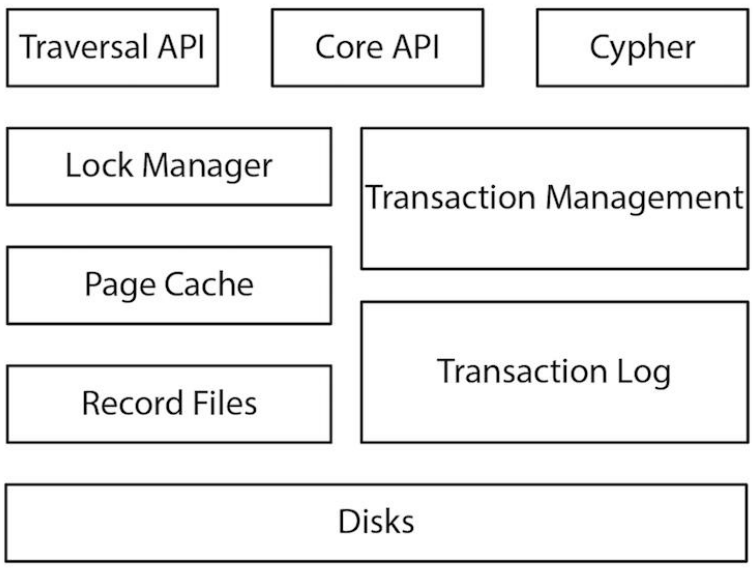
1) Labeled Property Graph Model: information is organized as nodes, relationships and properties. These elements are interacted within linked list structures.



2) Native graph database: it implements a true graph model all the way down to the storage level.

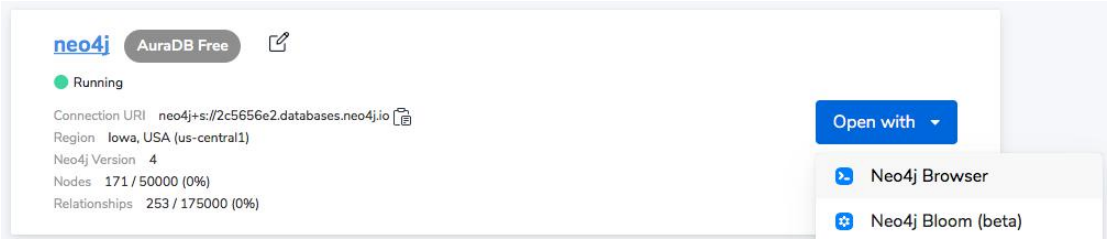
Store File	Record size	Contents
neostore.nodestore.db	15 B	Nodes
neostore.relationshipstore.db	34 B	Relationships
neostore.propertystore.db	41 B	Properties for nodes and relationships
neostore.propertystore.db.strings	128 B	Values of string properties
neostore.propertystore.db.arrays	128 B	Values of array properties
Indexed Property	1/3 * AVG(X)	Each index entry is approximately 1/3 of the average property value size

3) High-Level Architecture



Installation

1. Go to the homepage of Neo4j: https://neo4j.com
2. Click ‘Get Started’, choose ‘Neo4j AuraDB’, then click ‘Start Free’
3. Login with Google or Register with email
4. Select free type, input a name for database, click ‘Create Database’
5. Take a note of user and password for access to database
6. Database is created, click ‘Open with Neo4j Browser’, click ‘Connect’
7. Follow the guide to get familiar with Neo4j’s graph query language: Cypher



Data Definition & Manipulation

Let’s take a tour first. For more, look up Neo4j documentation: https://neo4j.com/docs/.

1) Create a node, labeled ‘Person’, with 3 properties (ee is a temporary variable):

2) Match a Person node with a given name:

3) Create 4 more persons, create 7 ‘KNOWS’ relationships:

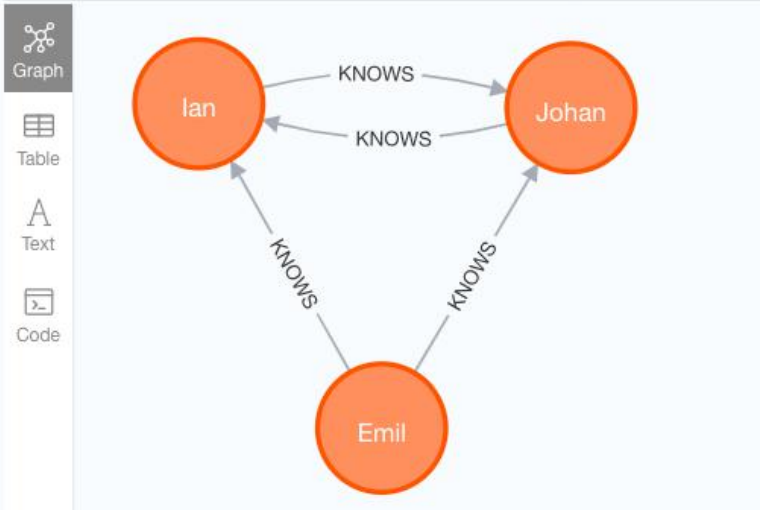
```
MATCH (ee:Person) WHERE ee.name = 'Emil'
CREATE (js:Person { name: 'Johan', from: 'Sweden', learn: 'surfing' }),
(ir:Person { name: 'Ian', from: 'England', title: 'author' }),
(rvb:Person { name: 'Rik', from: 'Belgium', pet: 'Orval' }),
(ally:Person { name: 'Allison', from: 'California', hobby: 'surfing' }),
(ee)-[:KNOWS {since: 2001}]→(js),
(ee)-[:KNOWS {rating: 5}]→(ir),
(js)-[:KNOWS]→(ir),
(js)-[:KNOWS]→(rvb),
(ir)-[:KNOWS]→(js),
(ir)-[:KNOWS]→(ally),
(rvb)-[:KNOWS]→(ally)
```

Table

Added 4 labels, created 4 nodes, set 14 properties, created 7 relationships, completed after 30 ms.

4) Match persons that Emil knows:

```
MATCH (ee:Person)-[:KNOWS]-(friends)
WHERE ee.name = 'Emil' RETURN ee, friends
```



Conclusion

Neo4j graph database is built on a powerful architecture, provides straightforward data modeling interfaces while keeping uncompromised performance, reliability, and integrity. Neo4j is perfect to store graph data (social network, knowledge graph) and perfect for doing analysis on it.

Reference

[1] Neo4j Documentation from https://neo4j.com/.
[2] Graph Databases by Ian Robinson, Jim Webber, and Emil Eifrem.