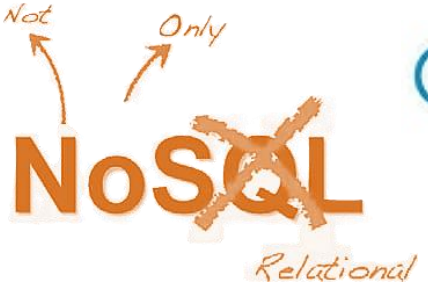


Graph Database

neo4j

Dr. SOHAIL IMRAN

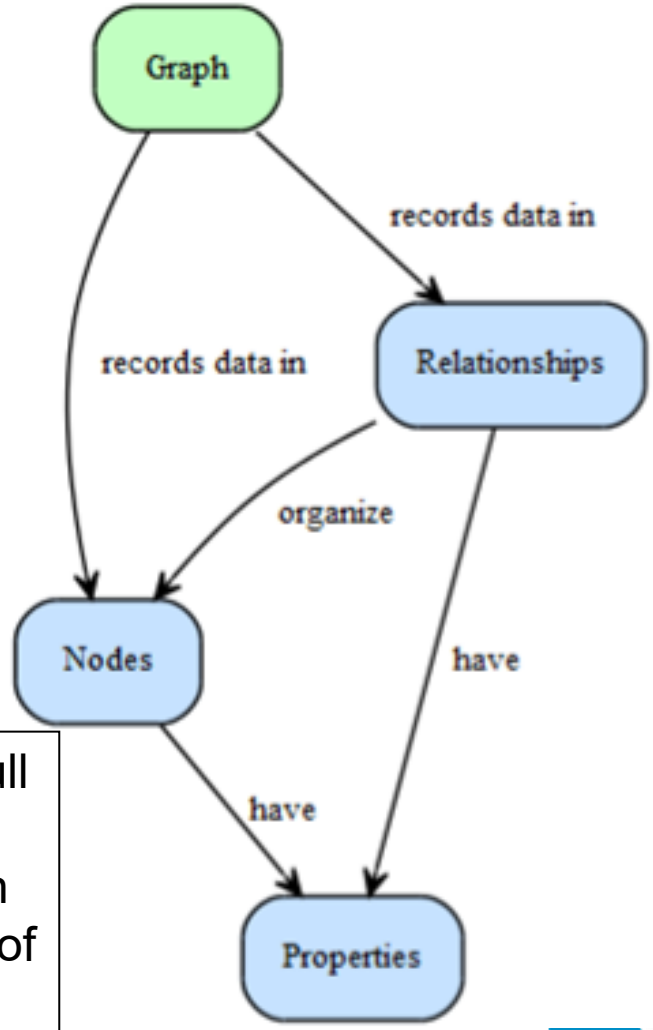
Intro

sohailimran2@yahoo.com

Ref: Web

intro

- A Graph contains **Nodes** (things or an attribute) and **Edges** (relationships) that connect pairs of nodes.
- A Graph – records data in → **Nodes** – which have → **Properties**
- A **record** that has named **values** referred to as **Properties**.
- **Relationships** connect two nodes and both nodes and relationships can hold an arbitrary amount of properties (**key-value pairs**).
- Each node and edge can have any number of attributes.
- Both the nodes and edges can be **labeled**.
- Labels can be used to narrow searches.



Graph database can be thought of as a key-value store, with full support for relationships.
Relationships organize Nodes into structures, allowing a Graph to resemble a List, a Tree, a Map, or a compound Entity – any of which can be combined into yet more complex richly inter-connected structures.

structure

"A Graph Database —manages a→ Graph and —also manages related→ Indexes"

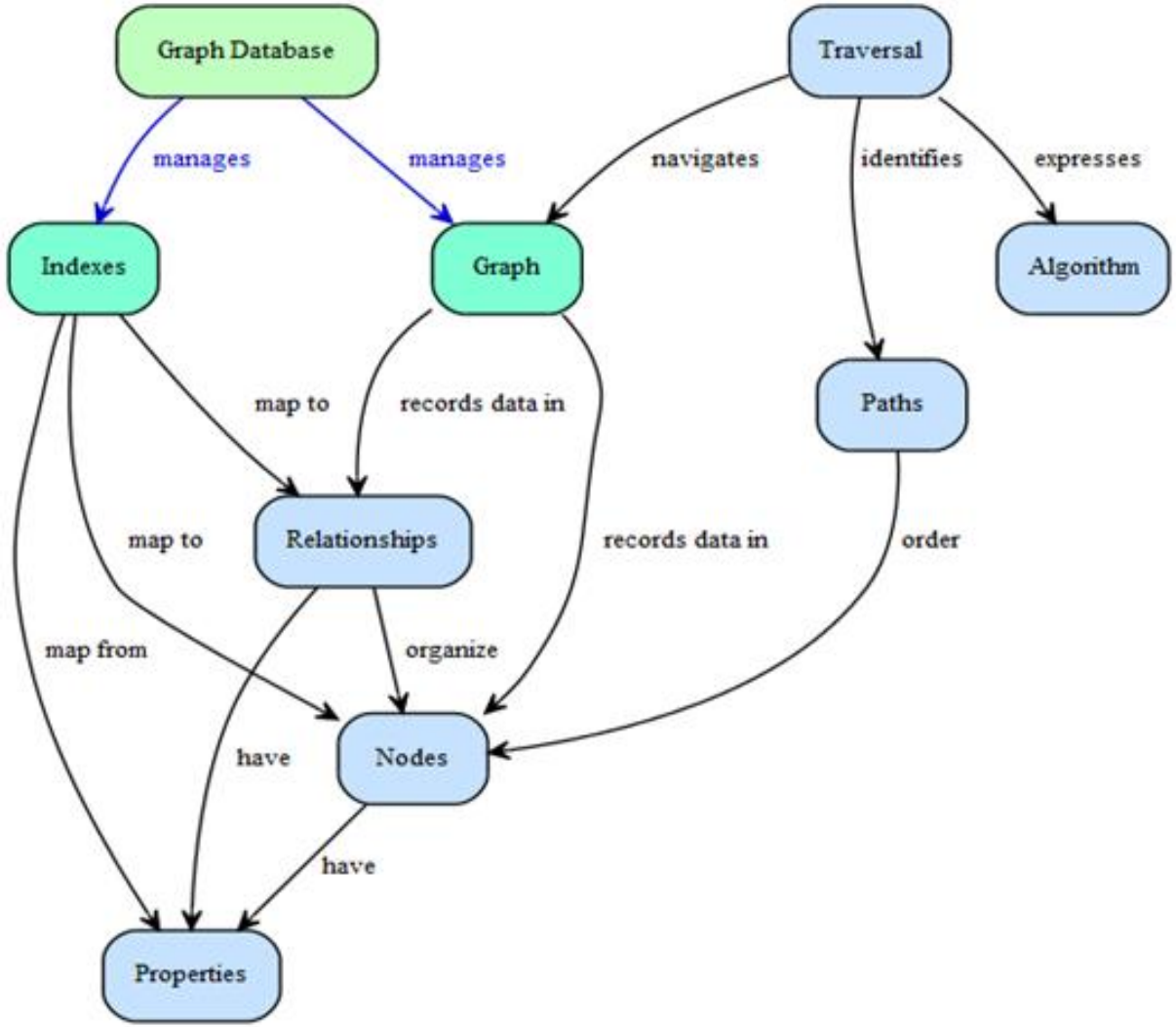
Neo4j is a commercially supported open-source graph database.

It was designed and built from the ground-up to be a reliable database, optimized for graph structures instead of tables.

Working with Neo4j, your application gets all the expressiveness of a graph, with all the dependability you expect out of a database.

Object (Vertex, Node)

Link (Edge, Arc, Relationship)



features

- Implemented in Java and accessible from software written in other languages using the Cypher query language through a transactional HTTP endpoint.
- Embeddable and server
- ACID-compliant transactional database with native graph storage and processing
- Schema free
- Neo4j is stable
- Neo4j is under active development
- High-performance graph operations
  - Traverses 1,000,000+ relationships / second on commodity hardware

components

Nodes (equivalent to vertices in graph theory). These are the main data elements that are interconnected through relationships. A node can have one or more labels (that describe its role) and properties (i.e. attributes).

Relationships (equivalent to edges in graph theory). A relationship connects two nodes that, in turn, can have multiple relationships. Relationships can have one or more properties.

Labels. These are used to group nodes, and each node can be assigned multiple labels. Labels are indexed to speed up finding nodes in a graph.

Properties. These are attributes of both nodes and relationships. Neo4j allows for storing data as key-value pairs, which means properties can have any value (string, number, or boolean).

node

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

Employee Node

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

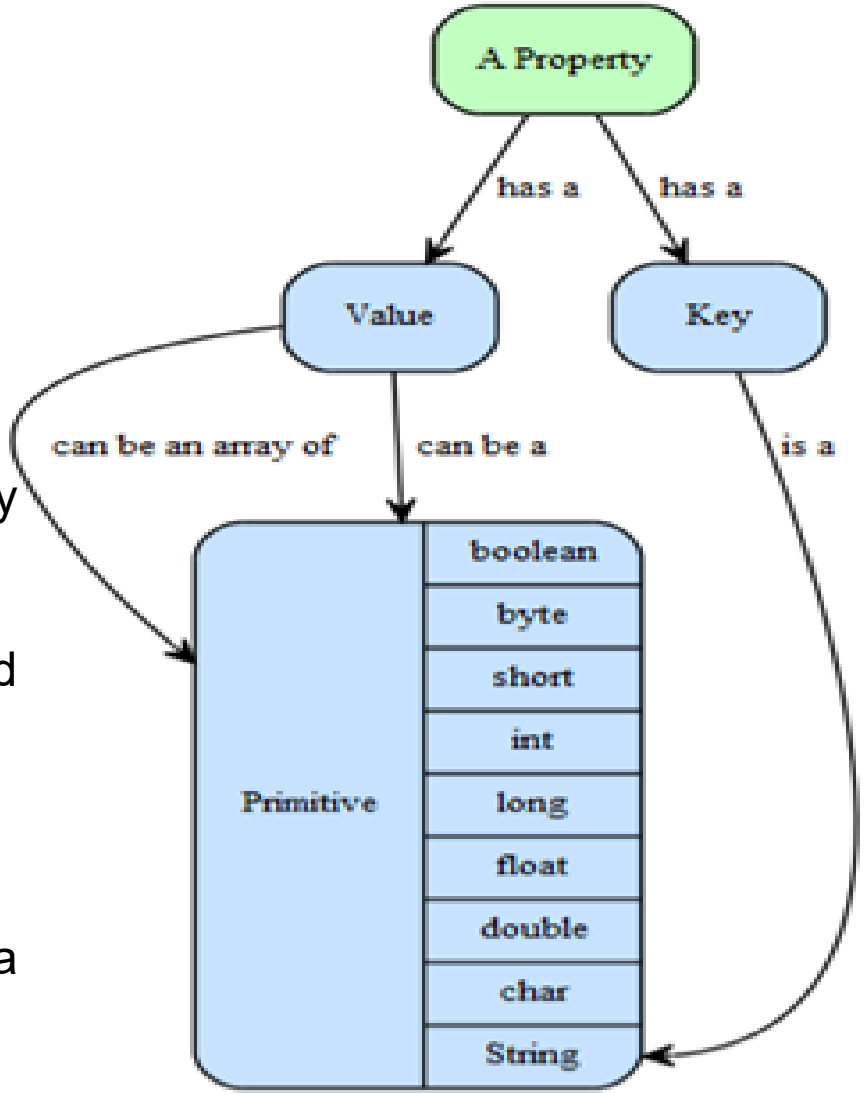
properties

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.
Property values can be either a primitive or an array of one primitive type.

For example: **String**, **int** and **int[ ]** values are valid for properties.

Note:
null is not a valid property value.
Nulls can instead be modeled by the absence of a key.



relationships

Relationships are another major building block of a Graph Database
It connects two.
Here, Emp and Dept are two different nodes.
"WORKS_FOR" is a relationship between Emp and Dept nodes.

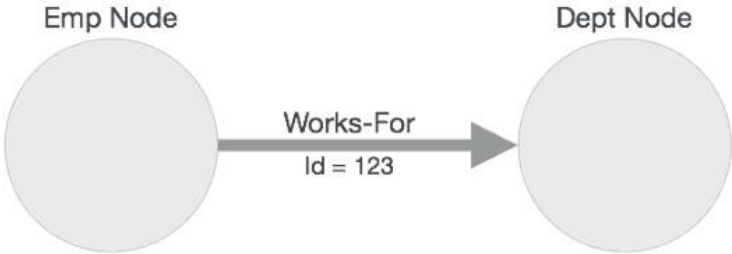
As it denotes, the arrow mark from Emp to Dept, this relationship describes −

```
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.

label

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.

Cypher Query Language

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.

read clauses

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.

write clauses

Sr.No	Write Clause	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 d

general clauses

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.

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.

data types

Sr.No	CQL Data Type	Usage
1	Boolean	It is used to represent Boolean literals: true, false.
2	byte	It is used to represent 8-bit integers.
3	short	It is used to represent 16-bit integers.
4	int	It is used to represent 32-bit integers.
5	long	It is used to represent 64-bit integers.
6	float	It is used to represent 32-bit floating-point numbers.
7	double	It is used to represent 64-bit floating-point numbers.
8	char	It is used to represent 16-bit characters.
9	String	It is used to represent Strings.

operators

Sr.No	Type	Operators
1	Mathematical	+, -, *, /, %, ^
2	Comparison	+, <>, <, >, <=, >=
3	Boolean	AND, OR, XOR, NOT
4	String	+
5	List	+, IN, [X], [X…..Y]
6	Regular Expression	=-
7	String matching	STARTS WITH, ENDS WITH, CONSTRAINTS

boolean operators

Sr.No	Boolean Operators	Description
1	AND	It is a Neo4j CQL keyword to support AND operation. It is like SQL AND operator.
2	OR	It is a Neo4j CQL keyword to support OR operation. It is like SQL AND operator.
3	NOT	It is a Neo4j CQL keyword to support NOT operation. It is like SQL AND operator.
4	XOR	It is a Neo4j CQL keyword to support XOR operation. It is like SQL AND operator.