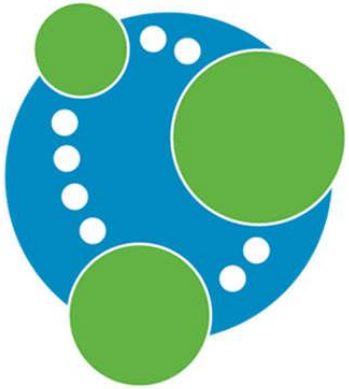


neo4j

Kyle Rainville
Littleton Coin Company

What is Neo4j?

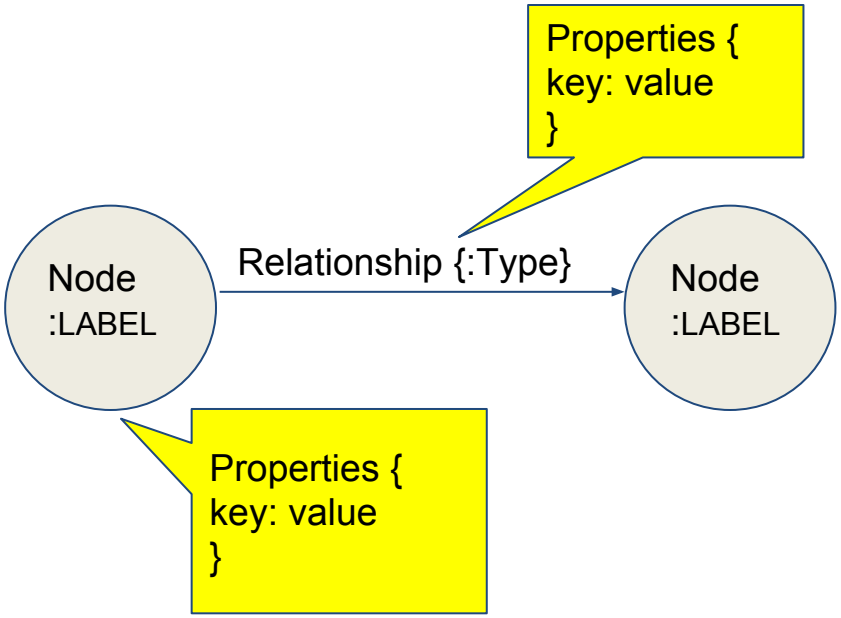
- Graph Database (GDBMS)
- Open source NoSQL DB – provides ACID-compliant transactional backend
- Most popular graph database
- Implemented in Java
- Accessible from software using the Cypher query language

Neo4j and IBM i

- Unfortunately, Neo4j, at this point in time cannot run on IBM i natively
- Neo4j can run on Linux on POWER

The Property Graph Model

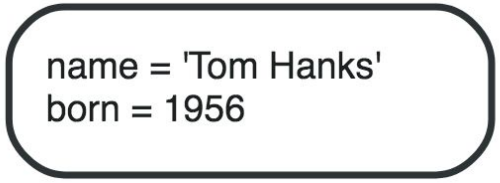
- Nodes
  - Labels
- Relationships
  - Types
- Properties

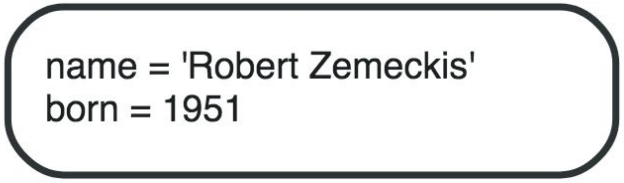


Nodes

- Nodes, in graph database terms, commonly represent entities
- The most basic of graphs is one single node

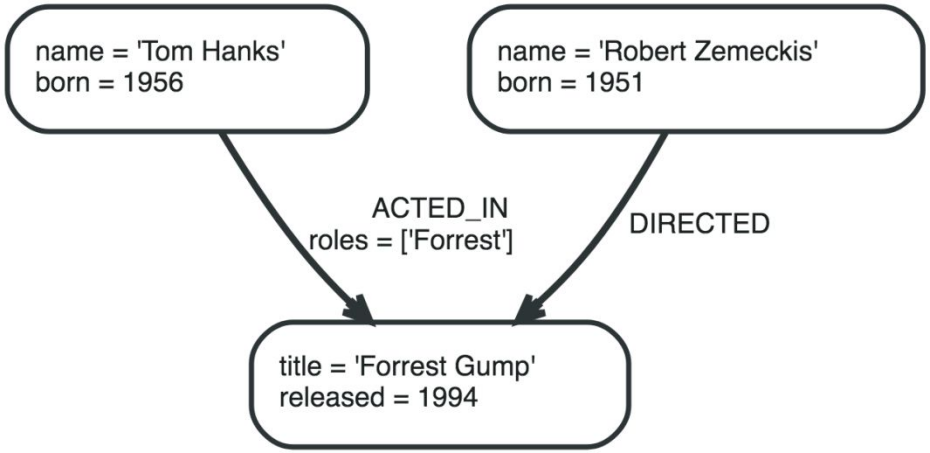
- The above is not very meaningful but graphs can quickly develop meaning





Relationships

- Explicitly connects nodes to each other
- Allows for the finding of related data
- Always have a type, start and end node, and direction
- Broken relationships are disallowed – guarantees source and target
- Self-referencing relationships are allowed
- Allows nodes to be organized into compound entities



Properties

- Named values, similar to key-value pairs
- Nodes and Relationships both have properties
- Supported property value types:
  - Numeric
  - String
  - Boolean
  - *null is not valid (null can be modeled by the absence of a property key)

Property Value Types

Type	Description	Value range
`boolean`	binary logic value	true/false
`integer`	64-bit integer	-9223372036854775808 to 9223372036854775807, inclusive
`float`	64-bit IEEE 754 floating-point number	-
`String`	sequence of Unicode characters	infinite

Labels

- Labels allow nodes to be grouped into sets
- Nodes may have multiple labels
- Labels can be indexed to allow for faster finding
- Label indexes optimized for speed
  - Allow for index free adjacency

Graph Databases

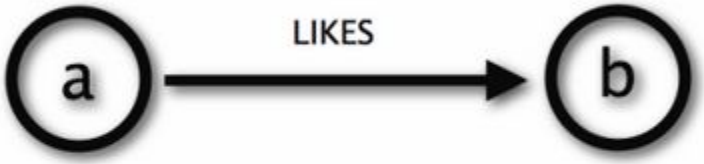
- A focus on relationships
- RDBMS – relationships computed at query time through joins
- GDBMS - Relationships are ‘first class citizens’
- Accessing nodes/relationships generally a constant-time operation

SQL and Cypher

- SQL has been the de facto language for RDBMS
- Cypher is a declarative language that serves the same purpose as SQL – for GDBMS
- Cypher – originally created by Neo Technology, was released in 2015 as open-source
- Has been adopted by several other GD vendors
- Graph Databases - need for a unified query language

Cypher Relationships

Cypher using relationship 'likes'



Cypher

```
(a) -[:LIKES]-> (b)
```

A Closer Look at Cypher

- Uses ASCII-Art to represent patterns
- Nodes are surrounded with parentheses
- Use arbitrary variables to refer to nodes
- Variable scope restricted to single statement
- To optimize execution and provide distinction, labels can be used [e.g. (p:Person) – [:LIKES] -> (f); matches the 'likes' (node-entities) of nodes labeled as 'Person']

Cypher Cont.

- Relationships are specified using an arrow (- ->) between nodes
- Square bracket inside arrow for specification (e.g. types)
  - Relationships - 1 type
  - Nodes - 0 or more labels
- Cypher allows patterns to be assigned to variables – to increase modularity and reduce repetition

Cypher General Examples

- Get all users by label

```
MATCH (user:User)
RETURN user
```

- Get specific user

```
MATCH (user:User)
WHERE user.Id = 1234
RETURN user
```

- Get a User and a count of their friends

```
OPTIONAL MATCH (user:User)-[FRIENDS_WITH]-(friend:User)
WHERE user.Id = 1234
RETURN user, count(friend) AS NumberOfFriends
```

- Get a user and all of their friends

```
OPTIONAL MATCH (user:User)-[FRIENDS_WITH]-(friend:User)
WHERE user.Id = 1234
RETURN user, collect(friend) AS NumberOfFriends
```

Cypher General Examples Cont.

- Create a user

```
CREATE (user:User { Id: 456, Name: 'Jim' })
```

- Create a user, if they do not already exist

```
MERGE (user:User { Id: 456 })
ON CREATE user
SET user.Name = 'Jim'
```

- Relate two existing users

```
MATCH (user1:User), (user2:User)
WHERE user1.Id = 123 AND user2.Id = 456
CREATE (user1)-[:FRIENDS_WITH]->(user2)
```

- Update a single property on a user

```
MATCH (user:User)
WHERE user.Id = 123
SET user.Age = 25
```

Cypher General Examples Cont.

- Replace all the properties of a user

```
MATCH (user:User)
WHERE user.Id = 123
SET user = { Id: 123, Age: 25, Email: 'tatham@oddie.com.au' }
```

- Delete a user

```
MATCH (user:User)
WHERE user.Id = 123
DELETE user
```

- Delete a user and all inbound relationships

```
OPTIONAL MATCH (user:User)<-[r]-()
WHERE user.Id = 123
DELETE r, user
```

- Get all labels for a specific user

```
MATCH (user:User)
WHERE user.Id = 1234
RETURN labels(user)
```

Case Sensitivity

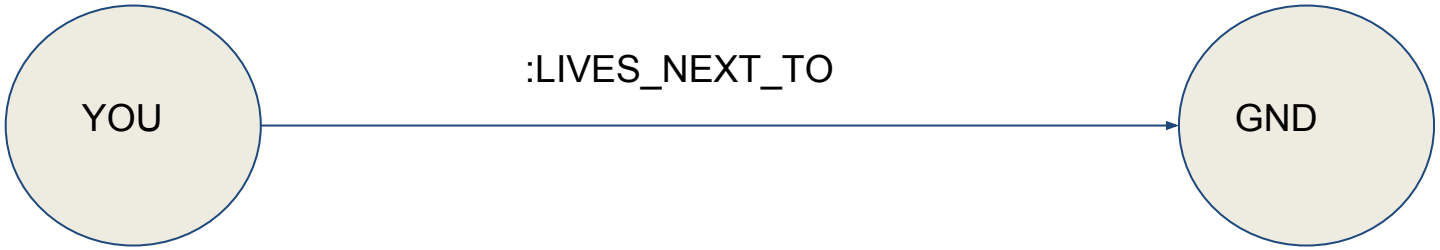
- Node labels
- Relationship types
- Property names
- (node:PERSON) is not the same as (node:Person)
- Other clauses, keywords, and functions are not – should still maintain consistent style
- Neo4j provides style/casing recommendations:
- https://neo4j.com/docs/developer-manual/current/cypher/syntax/naming/

Index Free Adjacency

- In relational DBs, indexes are computationally expensive to use - when joining two tables, the indexes on both tables need to be scanned completely to find every element fitting the query - the more joins, the longer the query runtime will take
- With graph databases you only use an index on the data once - to find starting point for traversals
- When starting point determined, the ‘network’ is walked by hopping through relationships without indexes - this is called ‘index-free adjacency’ and is a fundamental concept of GDBMS

“Direct Neighbor Walk” metaphor

- Girl-next-door - apple pie on pi day
- You want to get it to her while it’s still hot
- The steps:
  - Walk out your door
  - Turn towards her house
  - Walk directly to her house and give her the pie
- This is how a graph database would handle getting the pie to her



Delivering the Pie - RDBMS

- Use a central index(es) to find the address
- Steps
  - Walk out your door
  - Walk to DMV - use service that tells you how to get to your neighbor's house
  - When you get to the DMV, you take a number
  - When your number comes up, you get called
  - DMV agent has list of people sorted by address (called the index) - they search list and finally give you GPS coordiantes to girl's house
  - You take coordinates, enter them in tracker and follow directions to house
  - By the time you get there, pie is cold - she says "Thanks for the pie" and declines your invitation to lunch later that week

Officially Supported Drivers

- Connecting through programming languages
- Neo4j officially supported drivers
  - Java
  - Javascript
  - C#
  - Python
- Community support incredibly strong
  - Many other languages supported

Bolt Protocol

- Binary protocol, created by Neo Technology
- The official drivers all use this protocol - before Bolt, HTTP was used, but payload size caused inefficiencies
- Bolt drivers aim to act as an idiomatic API for officially supported languages

Popular Use Cases

- Recommender System
- Fraud Detection
- Social Media
- Identity & Access Management
- Knowledge Graph
- https://neo4j.com/use-cases/

How to Start

- Multiple ways to start playing
  - Neo4j Sandboxes (cloud containers)
  - Desktop installation
  - VMs (VirtualBox - Windows & Mac)
  - VMs (Linux - VirtualBox or KVM)

Red Hat/CentOS/Fedora Install

- Add the Neo4j repository
- Import the GPG key using ‘rpm --import’
- Use ‘yum install’ to download and install the Neo4j package (Red Hat/CentOS)
- Start Neo4j (‘systemctl start neo4j’)
- Test the installation by directing your browsers URL to “http://localhost:7474”
- Will bring you to the Neo4j browser interface

Neo4j Browser Interface

- Accessible from a browser after install
- Built in guides
  - Neo4j browser
  - Graph Database Concepts
  - Cypher
  - Graph examples
- Neo4j Browser Sync
- Ability to save and categorize queries
- Visually rendered results
- Ability to export results (CSV/PNG)

Demo - Loading into DB from CSV

- CSV file
- http://localhost:7474/browser/

Demo - Movie Graph

- http://localhost:7474/browser/
- :play movie-graph

When to use graph database

- If your data is highly interconnected
- Queries have a focus on relationships
- Expectation of data growing more interconnected
- Index-free adjacency

References

- https://neo4j.com/developer/graph-db-vs-rdbms/
- https://neo4j.com/developer/cypher-query-language/
- https://neo4j.com/docs/developer-manual/current/introduction/graphdb-concepts/
- https://neo4j.com/developer/cypher-query-language/
- https://github.com/Readify/Neo4jClient/wiki/cypher-examples
- https://neo4j.com/developer/guide-neo4j-browser/
- https://medium.com/@dmccreary/how-to-explain-index-free-adjacency-to-your-manager-1a8e68ec664a
- Van Bruggen, Rick. *Learning Neo4j*. Packt Publishing, 2014.