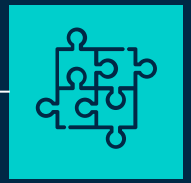
GRAPH DATABASES

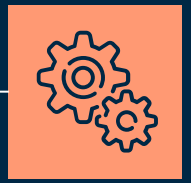
Jordi Romero Suárez
Ferran Sanchez Llado

TABLE OF CONTENTS



01

GRAPH DB



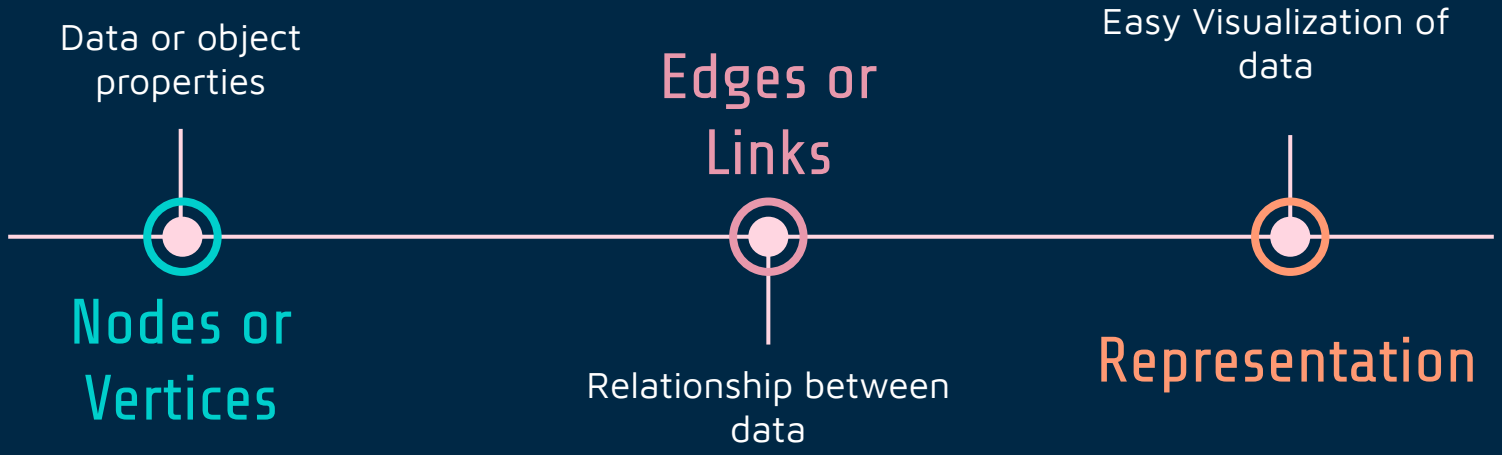
02

NEO4J

03

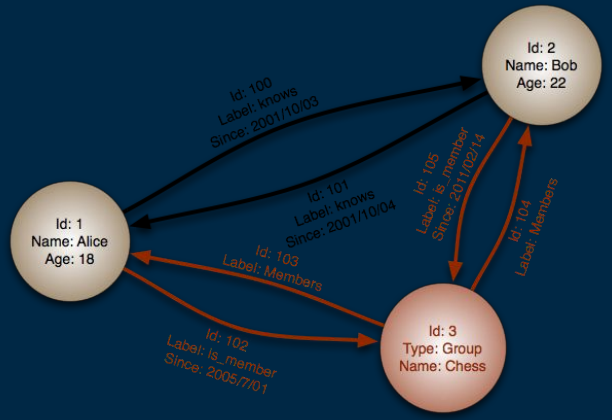
DEMO

WHAT IS A GRAPH-ORIENTED DATABASE



Evolution of Graph Database

- 1969: In CODASYL, definition of Network Database Language.
- 1980s: Labeled graphs.
- 1990s: Performance improvements.
- 2000s: Neo4j and Oracle Spatial and Graph became available.
- 2010s: Horizontal scalability and further enchantments.

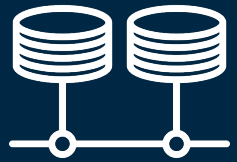


HOW DO THEY WORK?

- There is not a uniform query language.
- Use of graph algorithms (simplify and speed up complex queries)
  - DFS -> search for the next deepest node
  - BFS -> moves from one level to another
- Algorithms allow finding patterns or shortest path
- Relationships are saved in the database itself.
- Database avoids recalculating all relationships bring a big performance increase even with complicated queries.

A WAY OF REPRESENTING DATA

Relational Database



Graph Database

Good for:

- Well-understood data structures that don't change too frequently
- Known problems involving discrete parts of the data with minimal connectivity

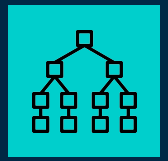
Good for:

- Dynamic systems: where the data topology is difficult to predict and has a high connectivity
- Dynamic requirements: the evolve with the business
- Problems where the relationships in data contribute meaning and value

USE OF GDB

Hierarchical trees

Organizational structures of companies



Recommendation systems

Relationship of similarity or proximity of products

Route calculation in logistics

Most optimal route to distribute products

Social relationships

Finding relationships between people

ADVANTAGES AND DISADVANTAGES

Advantages

- Search speed depends only on the number of specific relationships, not on the data set
- Real-time results
- Intuitive and summary presentation of relationships
- Flexible and agile structures

Disadvantage

- It is difficult to scale, as it is designed for single server architectures
- Without a fixed schema to make consistent queries

MOST KNOWN GDB

Neo4j

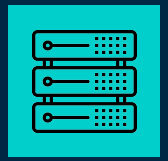
Intuitive, using a graph model for data representation

Janus Graph

Elastic and linear scalability for a growing data and user base

DGraph

Low enough latency to be serving real time user queries

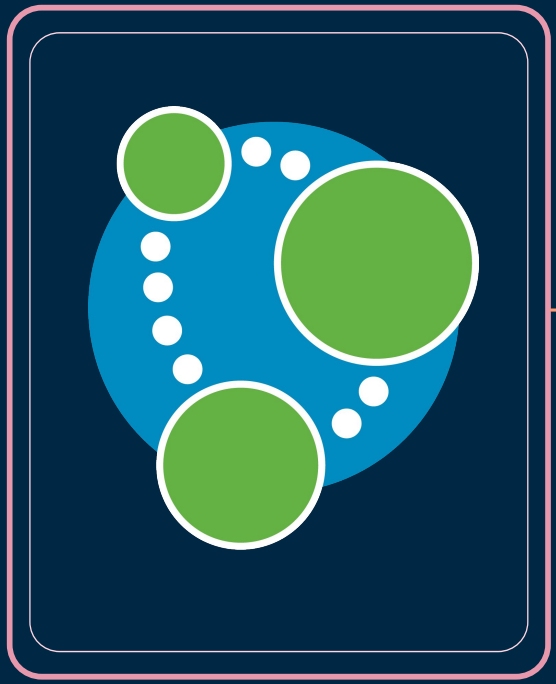


Giraph

Analyze social media data

WHY NEO4J?

- Highly Performant Read and Write Scalability
- High Performance Thanks to Native Graph Storage & Processing
- Easy to Learn (UI with intuitive interaction)
- Easy to Use(**Cypher**, the world's most powerful and productive graph query language)



NEO4J PROPERTIES

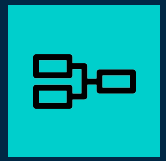
WHAT IS CYPHER

- Declarative graph query language that allows for expressive and efficient querying, updating and administering of the graph.
- Designed to be suitable for both developers and operations professionals.
- Designed to be simple, yet powerful.
- Inspired by a number of different approaches and builds on established practices for expressive querying.

NEO4J CLAUSES

MATCH

Search for the pattern described in it.



CREATE

Create nodes and relationships.

SET

Update labels on nodes and properties on nodes and relationships

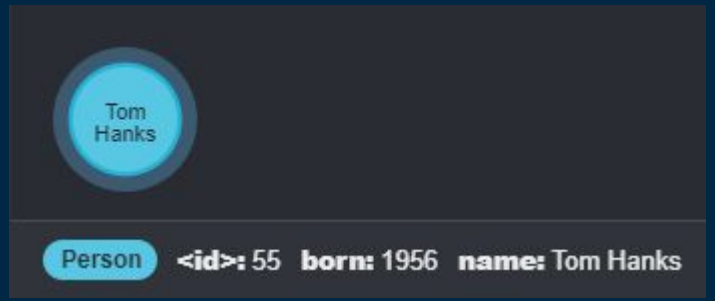
DELETE

Delete nodes or relationships

NODE VISUALIZATION

Syntax: MATCH(var {Property1:'PropertyName1') RETURN var

Example: MATCH (tom {name: "Tom Hanks"}) RETURN tom



NODE CREATION

Syntax: CREATE (Name:Label {Property1:'PropertyName1', Property2:'PropertyName1'})

Example: CREATE (n:Person {name: 'Andy', title: 'Developer'})

CREATION OF RELATIONSHIPS

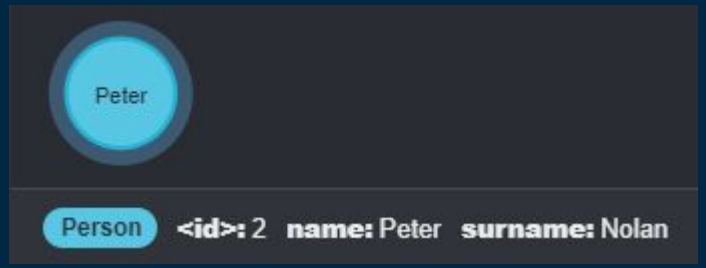
Syntax: (node1)-[:RELATION_TYPE]->(node2),

Example: (Madonna)-[:ACTED_IN] -> (A League of Their Own)

UPDATE LABELS/PROPERTIES

Syntax: (var{Property1:'PropertyName1'}) SET var.Property2 = 'newProperty'

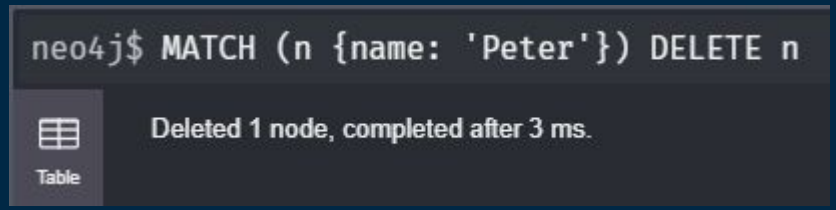
Example: (n {name: 'Andy'}) SET n.surname = 'Taylor'



DELETE LABELS/PROPERTIES

Syntax: (var{Property1:'PropertyName1'}) DELETE var

Example: (n {name: 'Peter'}) DELETE n



CONCLUSIONS

- Learned a new engine
- Useful at the moment that there are many relationships
- Easy to learn and use

Now let's see the demo