Linked Open Data and Knowledge Graph

Basic Topic: Property Graphs

Table of Content

Introduction

- Property Graph is a data model for representing relationships where set of nodes (Vertices) and edges (relationships) can carry properties or attributes (key-value pairs).
- They allow rich descriptive meta data on each edge and node, making them versatile for capturing nuanced information.
- In knowledge graphs, property graphs provide more flexible schema, which is beneficial for representing diverse data.

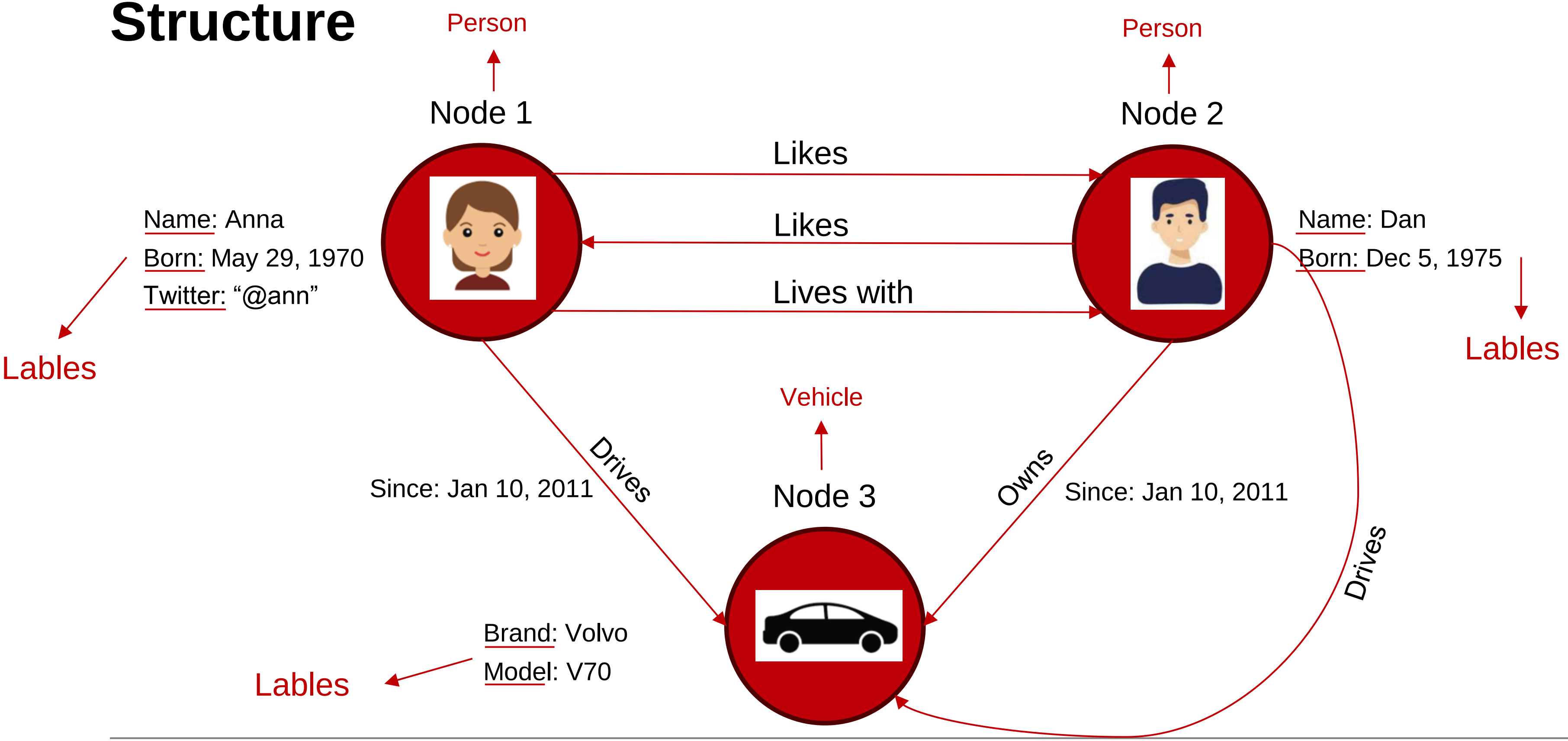
History

- **1970s**: Foundations of graph theory emerge; relational databases struggle with complex data relationships.
- **1980s**: Early research into graph models highlights the need for connected data solutions.
- **1990s**: Object-oriented databases attempt flexibility but lack true graph capabilities.
- **2000**: Neo4j development begins, pioneering the property graph database.
- **2007**: Neo4j officially launches, popularizing the property graph model for connected data.
- **2010s**: Property graphs gain attraction in social networks, recommendations, and real-time analytics.
- **Late 2010s**: Knowledge graphs and Linked Data drive demand for property graph solutions.
- Now, in the **2020s**, property graphs have gone fully mainstream, supporting big data, cloud, and enterprise applications, with evolving standards to meet the demands of increasingly complex data systems."

Structure

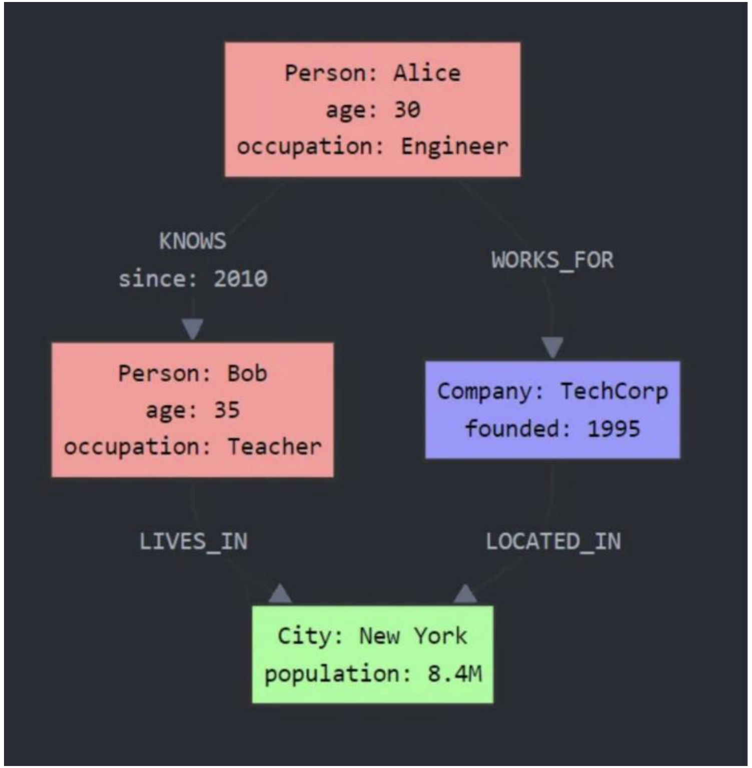
- Nodes represents entities or objects like people, places, items etc.
- Edges captures the relationships between these nodes, like “works at”, “located in” etc.
- Edges are directional, they have a start and end node.
- Labels are used to categorize nodes and edges.
- Properties stores additional meta data about nodes and edges, which can includes attributes like names, dates or measurements.

Structure



Technology
Arts Sciences
TH Köln

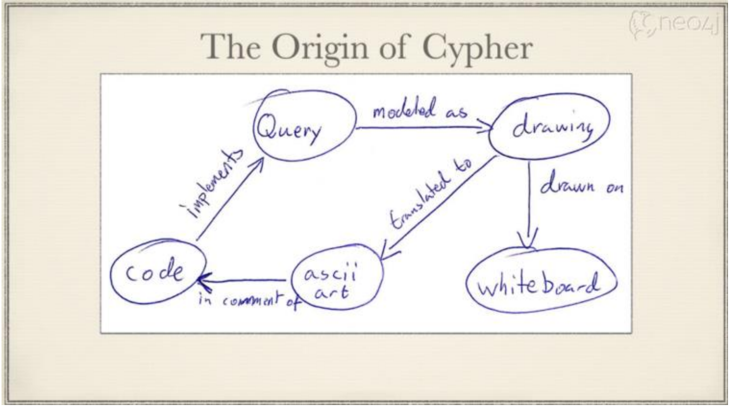
Example Structure

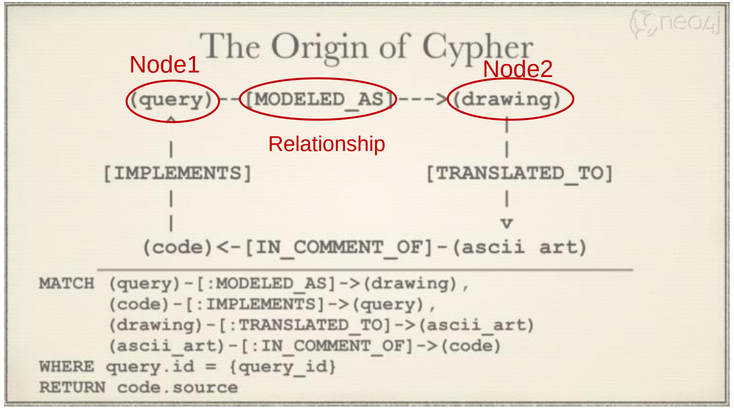


https://medium.com/@taaha.com/what-on-earth-is-property-graph-fa7a17eb8087

Language

- Gremlin is domain-specific language created specifically for graphs and written in Groovy.
- The language is the ancestor of most graph query languages today and is known as Cypher.
- Cypher is a powerful, SQL-like language designed specifically for querying and manipulating graph data.
- Cypher's syntax is based on the use of ASCII art to describe graph patterns.
- **Pattern Matching**: Allows querying nodes and relationships in a visually intuitive way.





https://docs.nebula-graph.io/3.3.0/1.introduction/0-1-graph-database/#the_creation_of_cypher

Related topics

GQL (Graph Query Language)

- **ISO Standard**: An emerging international standard for querying graph databases.
- **Unified Language**: Aims to standardize querying across both property graphs and RDF.
- **Cypher-Based Syntax**: Builds on Cypher for intuitive, powerful querying.
- **Cross-Platform**: Enables consistent querying across different graph database systems.

GMI (Graph Model Integration)

- **Interoperability Goal**: Integrates different graph models (property graphs, RDF, etc.).
- **Seamless Querying**: Allows data from diverse graph sources to be queried uniformly.
- **Future-Proofing**: Supports greater compatibility and flexibility in graph-based applications.
- **Enhanced Data Integration**: Bridges graph models, making data connections easier and more versatile.

Comparison

Aspect	Property Graph	RDF (Resource Description Framework)
Data Model	Nodes (entities) and Edges (relationships) with properties	Triples (subject, predicate, object), representing statements of data
Relationships	Edges have properties	Relationships are described by predicates but not properties directly
Schema Flexibility	Schema-less (flexible, dynamic)	Flexible but often follows a formal **ontology** or schema
Identifiers	Nodes and edges can have **unique IDs** or labels (optional)	Uses **URIs** to uniquely identify resources and relationships
Data Linking	No inherent focus on external linking; focuses on **local** graph structure	Designed for linking **data across datasets** using **URIs** (Linked Data)
Query Languages	**Cypher** (Neo4j), **Gremlin** (Apache TinkerPop)	**SPARQL**
Interoperability	Typically optimized for single graph database environments	Designed for global data interoperability (across distributed datasets)
Focus	Fast traversal of relationships within a single graph	Data integration and queryability across **distributed** data sources
Data Integrity	Focus on **internal** graph structure and traversal	Focus on **semantic** consistency and relationships across datasets
Data Representation Standard	No strict formal standard; various graph database implementations	Defined by **RDF specification**, providing a formal standard for data representation

Applications

1. Social Networks → Facebook, Linkedin
2. Recommendation Systems → Netflix, Amazon, Spotify
3. Fraud detection → Bank institutions, E-commerce
4. ------------

Advantages

1. Improved Performance on Large graphs
2. Flexible schema
3. Agility
4. Rich Data Representation
5. Enhanced Query Capabilities
6. Interoperability with Other Data Models
7. Simplified Relationship modeling

Conclusion

1. Property graphs are a powerful tool for modeling and analyzing complex, interconnected data.
2. Their flexible, schema-less structure enables efficient querying and real-time insights, making them ideal for dynamic, data-intensive environments.
3. As industries seek deeper insights from connected data, property graphs continue to be essential for unlocking valuable information and driving innovation across various domains.

References

1. https://www.youtube.com/watch?v=NH6WoJHN4UA
2. https://medium.com/@taaha.com/what-on-earth-is-property-graph-fa7a17eb8087
3. https://docs.nebula-graph.io/3.3.0/1.introduction/0-1-graph-database/#the_creation_of_cypher
4. https://www.precisely.com/resource-center/productsheets/property-graph
5. https://neo4j.com/use-cases/knowledge-graph/?utm_source=GSearch&utm_medium=PaidSearch&utm_campaign=GenAI-KG-EMEA-EMEANorth&utm_ID=&utm_term=knowledge%20graph&utm_adgroup=genai-specific_knowledge-graph&utm_content=-Sitetraffic--&utm_creative_format=Text&ut
6. https://rubygarage.org/blog/neo4j-database-guide-with-use-cases