BDA Report: Neo4J

Tobias Kneidinger, Rafael Hochedlinger

December 2, 2024

Introduction to Graph Databases

Graph Database Management Systems (GDBMS) are a type of NoSQL database which - instead of traditional tables with rows and columns - uses graph structures with nodes, edges, and properties to represent and store data, as seen in Figure 1. This is especially useful for data that is highly interconnected, such as:

- Social networks
- Recommendation engines
- Fraud detection
- Network and IT operations
- Master data management
- Real-time pricing and promotions
- Identity and access management
- etc…

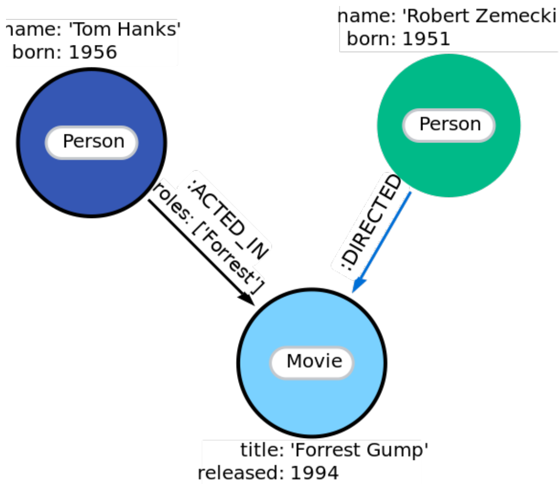

Figure 1: Graph Database Structure by Neo4j, 2024a

Graph databases are optimized for traversing and querying graph structures, which makes them very efficient for certain types of queries. As they are NoSQL databases, they are also schema-less and can store data in a flexible way. This makes them also very suitable for machine learning and data science applications, as the data can be stored in a way that is directly usable for the algorithms.

Neo4J

Neo4J is a popular GDBMS that is used by many companies and organizations. It is developed by Neo Technology and is written in Java, the first version was released in 2007. Neo4J is an open-source database, but there is also a commercial version available with additional features and support. (Neo4j, 2024a)

It uses the property graph model, which consists of the following elements, as described by Lal, 2015:

- **Nodes**: Entities that represent objects or entities in the graph and can have multiple properties and labels
- **Relationships**: Connect nodes and represent the connections between them, can have a direction and a type and can also have properties
- **Properties**: Key-value pairs that can be attached to nodes and relationships
- **Labels**: Represents the type of a node, can be used to group nodes together

The main query language for Neo4J is Cypher, which is a declarative query language that is similar to SQL. These queries can be used to create, read, update, and delete data in the database. The language is more thorouhgly described in the next section.

For data science and machine learning applications, Neo4J also provides the Graph Data Science (GDS) library, which contains a collection of graph algorithms that can be used to analyze and process graph data. This library is also described in more detail in a later section.

Core Concepts in Neo4j

Rafi

Graph Algorithms with GDS

To efficiently analyze and process graph data for data science and machine learning applications, the Graph Data Science (GDS) library can be used. The algorithms in this library are optimized for graph data and can be used to perform various tasks, as described by Neo4j, 2024b:

- Community detection
- Centrality analysis
- Link prediction
- Similarity analysis
- Pathfinding
- etc...

To be able to use conventional neo4j graphs with the GDS library, the graph must be projected into a named graph. This projected graph is a subgraph of the original graph and can be created with the query shown in Figure 2, where 'socialgraph' is the name of the newly projected graph, 'Person' is the node label to project, and 'FRIENDS' is the relationship type to project.

```
CALL gds.graph.create('socialgraph', 'Person', 'FRIENDS')
```

Figure 2: Projected graph creation

After the graph has been projected, the GDS algorithms can be used to analyze the graph. In the example shown in Figure 3, the Louvain algorithm is used to detect communities in the graph. The algorithm is called with the 'socialgraph' as the graph name, the 'Person' node label, and the 'FRIENDS' relationship type. The algorithm yields the nodeId and communityId, which can be used to return the node name and community id. The results are then ordered by the community id.

```
CALL gds.louvain.stream('socialgraph')
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).name AS nodeName, communityId
ORDER BY communityId;
```

Figure 3: GDS algorithm execution

Instead of `stream` the algorithm can also be executed with `write` to write the results back to the graph. The results can then be used for further analysis or visualization and don't have to be recalcualted every time.

Performance and Scalability

Rafi

Use Cases and Applications

As specified in the introduction, graph databases are especially useful for data that is highly interconnected. This makes them very suitable for a wide range of applications, as described by Neo4j, 2024a:

- **Social networks**: To store and analyze user data, relationships, and interactions
- **Recommendation engines**: To provide personalized recommendations based on user behavior
- **Fraud detection**: To detect fraudulent activities and patterns in financial transactions
- **Network and IT operations**: To monitor and analyze network traffic and IT infrastructure
- **Master data management**: To store and manage master data and relationships between entities
- **Real-time pricing and promotions**: To calculate and provide real-time pricing and promotions
- **Identity and access management**: To manage and analyze user identities and access rights

The flexibility and scalability of graph databases make them very versatile and suitable for many different use cases, as they can also be combined with other technologies and databases such as relational databases, document databases, or key-value stores.

However, not all use cases are suitable for graph databases. If the data is changing frequently and the relationships don't have to be queried often, another technology -

like a relational database - might be the better choice. Also, if the data is not highly interconnected and the queries are simple, a key-value store or document database might be more efficient.

Summary and comments

Rafi

Chapter 1

References

Lal, M. (2015). *Neo4j graph data modeling*. Packt Publishing. https://books.google.it/books?id=WTlECgAAQBAJ

Neo4j. (2024a). *Neo4j* [Accessed: 2024-12-02]. https://neo4j.com/

Neo4j. (2024b). *Neo4j graph data science documentation* [Accessed: 2024-12-02]. https://neo4j.com/product/graph-data-science/