Neo4j Graph Database

2000: Development begins as the co-founders were working on a content-management system and couldn't solve a problem with relational databases so they came up with the property graph model (combines best of graphs and relational databases by combining focus on relationships between nodes with possibility to add properties to both)

2003: Their fist graph database was in 24/7 production
2007: First native graph database (storing the relationships directly rather than indirectly through joins; will explain in more detail in design and architecture section); Neo4j1.0 released as open source and commercial package
2011: Cypher launched as first declarative language designed to query property graphs
2013: Model extended to 'labelled' property graph to allow for easy lookup and creating of groups that go together

Originally written to support near realt-ime online transactional processing, but focus gradually shifted to also include strong capabilities around analytical

Design and Architecture

English prose and iconography, making queries eay to write and read

```
(nodes) – [:ARE_CONNECTED_TO] –> (otherNodes)
```

neo4j Aura cloud version of Neo4j database, fully managed updates and patches (no interruption to service), ACID compliant, robust security and reliability; built-in tools for learning, building and visualising

Analytics Graph data science, software component whose main purpose is to run graph algorithms on in-memory projects of Neo4j database data, including ML modelling.

Neo4j tools Includes wide range of tools, such as:

- a browser client which can interact with the database,
- an ops manager that enables administrators to monitor, administer and operate all of the Neo4j Databases
- cypher shell, a CLI that can run queries and perform administrative tasks against a Neo4j instance
- Kubernetes helm charts, to help deploy and manage the Neo4j databse on kubernetes

Visualization Bloom is their visualisation tool which allows to quicly explore and freely interact with Neo4js graph data platform - with no coding required

English prose and iconography, making queries eay to write and read

```
(nodes) – [:ARE_CONNECTED_TO] –> (otherNodes)
```

neo4j Aura cloud version of Neo4j database, fully managed updates and patches (no interruption to service), ACID compliant, robust security and reliability; built-in tools for learning, building and visualising

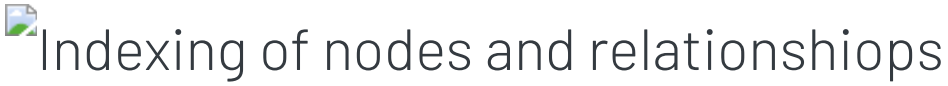
Analytics Graph data science, software component whose main purpose is to run graph algorithms on in-memory projects of Neo4j database data, including ML modelling.

Neo4j tools Includes wide range of tools, such as:

- a browser client which can interact with the database,
- an ops manager that enables administrators to monitor, administer and operate all of the Neo4j Databases
- cypher shell, a CLI that can run queries and perform administrative tasks against a Neo4j instance
- Kubernetes helm charts, to help deploy and manage the Neo4j databse on kubernetes

Visualization Bloom is their visualisation tool which allows to quicly explore and freely interact with Neo4js graph data platform - with no coding required

operators, which are combined into a tree-like structure that forms the execution plan *execution engine* performs execution plan and aggregates the results. The leaf nodes of the tree-like structure resolve into records, and the output is then piped back up the tree. Non-leaf nodes require sub-selection output set from the upstream branches. It is possible to configure and fine-tune the execution plan manually if you know what you're doing.


Indexing of nodes and relationshiops

Neo4j uses index-free adjacency to organise data records on disk. This means that rather than mapping record keys to a location on a disk (which can take up significant disk space for large databases and impact write performance), every node references its adjacent or next node directly. Each node is stored as a singly-linked list and the referenceing to the next node functions as a micro-index stored in the node itself.
Edges are stored as doubly-linked lists, which means they store a reference to both the preceding node (start point) and the next node (end point). This means that processing occurs at a constant rate regardless of data size, rather than traditional indexing which

physcial storage of a graph on disk

It relies heavily on memory for caching disk contents for performance or as temporary storage

scaling

The cloud version offers architecture that scales with your data needs and minimizes infrastructure costs while maximizing performce across connected datasets. Autonomous clustering lets you horizontally scale out your data while taking advantage of infrastructure elasticity, with less manual effort. You can also scale out very large graphs across multiple databases while maintaining query simplicity and performance

Atonomous clustering

architecture automatically allocates copies to the optimal servers based on default business rules or specified operations requirements

Useful when throughput demand for the same dataset arises

cluster support

Though one of the early criticism of Neo4j, later editions offer extensive cluster support via causal clustering. It facilitates a single unit of servers (cluster) to maintain its state across replicas, even when communication may be interrupted. Applies:

- causal consistency: concurrent writes are applied in the same order that they occur by looking at the order causally related operations are processed. Uses transaction IDs to determine how far (in transaction number) a follower is behind its leader Ensures a **read-your-writes** consistency: https://docs.oracle.com/cd/E17076_05/html/gsg_db_rep/C/rywc.html (Follower keeps track how consistent they are with a Leader copy, if determined a copy is not consistent enough, developer can specify action to be taken (e.g. reroute to different server that is consistent enough, block the read, reject the request outright, etc)).
- consensus algorithm: relies on the Raft protocol: https://thesecretlivesofdata.com/raft/. This handle the election of the leader copy and how to maintain a consistent log across the cluser.

Position in the CAP Model

Neo4j aligns with the **Consistency and Partition Tolerance** aspects of the CAP theorem.
More details

What about this

- **Consistency**: Neo4j ensures that all database clients see the same data at the same time, maintaining a high level of data accuracy.
- **Partition Tolerance**: It can function across a distributed network, ensuring continuous operation despite possible node failures.

The trade-off is in terms of Availability in some scenarios, where the system may sacrifice immediate data availability to maintain consistency.

- Neo4j aura designed to be always on and available, with all corrections, fixes, and upgrades automatically applied in the background. Releases for the Neo4j database are also deployed when they become available. Operations are non-disruptive, and you shouldn't experience downtime as a result
- You can also scale out very large graphs across multiple databases while maintaining query simplicity and performance
- Read-your writes level of consistency https://docs.oracle.com/cd/E17076_05/html/gsg_db_rep/C/rywc.html (copies track how consistent they are with a master copy, if determined a copy is not consistent enough, developer can specify action to be taken (e.g. reroute to different server that is consistent enough, block the read, reject the request outright, etc.))

Use Cases

Neo4j finds applications across various domains, offering solutions for complex data relationships:

1. Social Networks Analysis

- *User Connection Insights*: Mapping intricate social connections to provide deep insights into user relationships, community formation, and influence patterns.
- *Tailored Content Delivery*: Analysing user preferences and interactions to deliver personalised content, thus enhancing user engagement and experience.

2. Recommendation Engines

- *E-Commerce Personalisation*: Utilising customer purchase history and preferences to suggest relevant products, thereby increasing sales and customer satisfaction.
- *Media and Content Recommendations*: Empowering streaming services to suggest films, shows, or music based on user behaviour, enhancing personalised user experiences.

3. Fraud Detection in Financial Services

- *Pattern Recognition*: Identifying unusual patterns to detect fraudulent activities in real-time, crucial for financial institutions.
- *AML (Anti-Money Laundering)*: Playing a key role in uncovering complex money laundering schemes through transaction network analysis.

4. Network and IT Operations

- *Infrastructure Management*: Aiding in visualising and managing network infrastructures, thus improving efficiency in issue identification and resolution.
- *Security Analysis*: Employed for detecting vulnerabilities and intrusion attempts by analysing network traffic patterns.

Pros and Cons of Neo4j

Pros

Highly Intuitive for Connected Data

- *Deep Link Analysis*: Facilitates in-depth analysis of relationships and interconnections, crucial in domains like social network analysis and recommendation systems.

Flexible Schema

- *Adaptability*: Neo4j's schema adapts easily to changing data structures, essential in dynamic sectors like e-commerce and digital marketing.

Powerful Query Language - Cypher

- *Efficient Data Retrieval*: The expressiveness of Cypher simplifies complex queries, enhancing data analysis and retrieval efficiency.

Cons

Learning Curve

- *Specialised Knowledge Required*: Mastery of graph databases and Cypher entails a learning curve, potentially challenging for teams more familiar with SQL or traditional databases.

Scalability Challenges

- *Handling Large-Scale Data*: Although improving, Neo4j faces challenges in managing extremely large datasets and ensuring performance at scale, especially when compared to certain other NoSQL databases.

Resource Intensity

- *High Memory Requirements*: Managing large graphs demands significant memory resources, which can escalate operational costs.

Resources

- Getting Started with Neo4j