Introduction to Neo4j

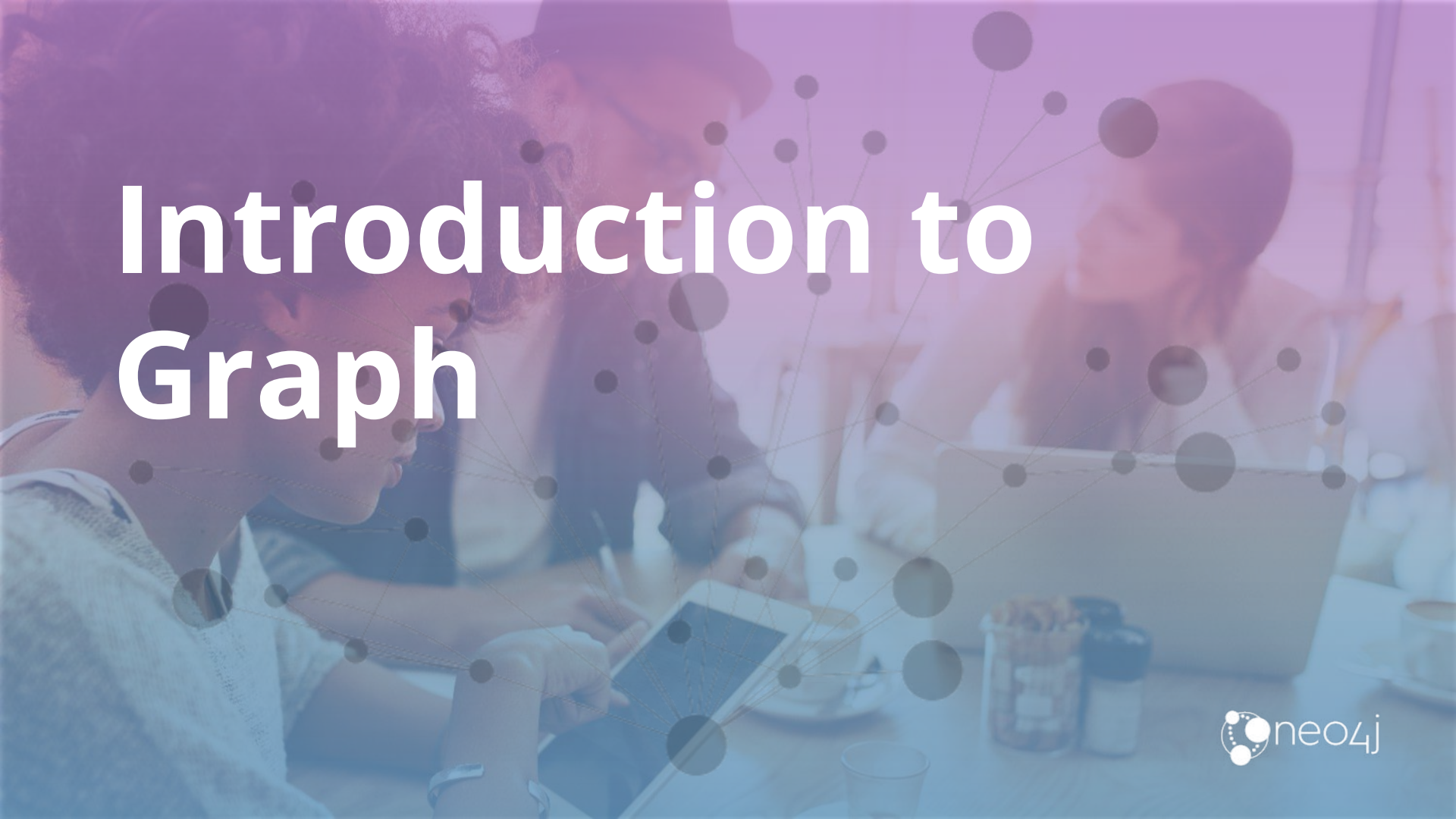
(WE)-[:LOVE]->(GRAPHS{name:"Neo4j"})

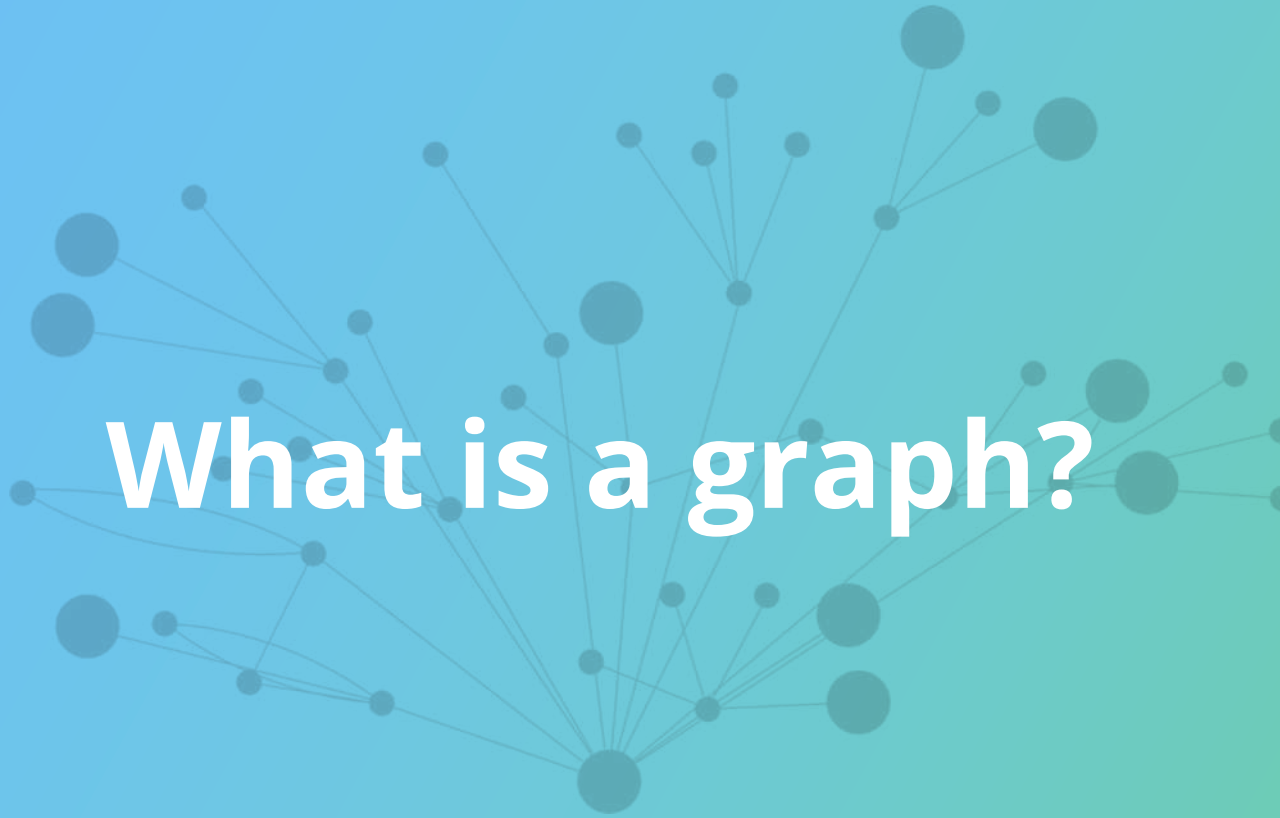
Soham Dhodapkar
Field Engineer
Neo4j
soham.dhodapkar@neo4j.com

Stuart Laurie
Senior Field Engineer
Neo4j
stuart.laurie@neo4j.com

Mark Quinsland
Senior Field Engineer
Neo4j
mark.quinsland@neo4j.com

Introduction to Graph



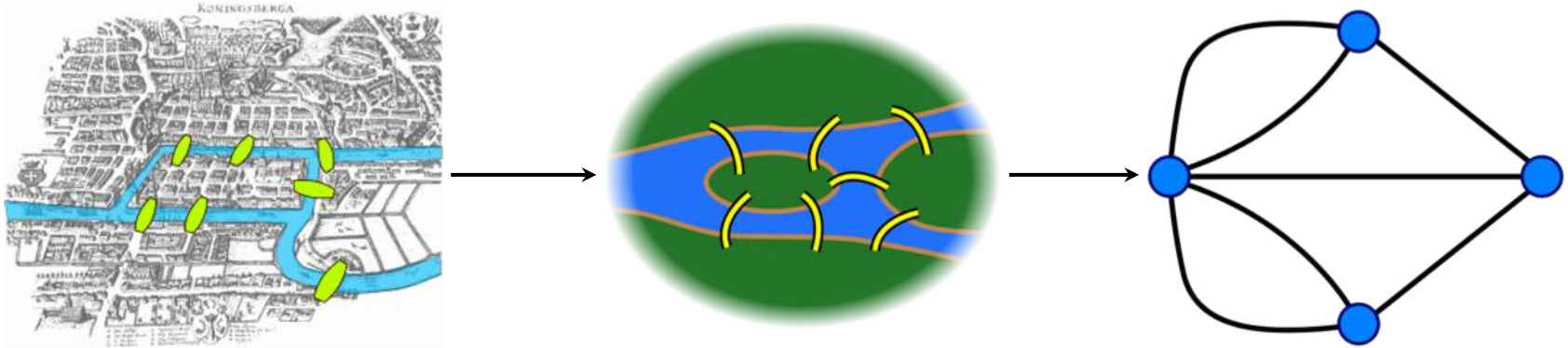


What is a graph?

neo4j

A graph is…

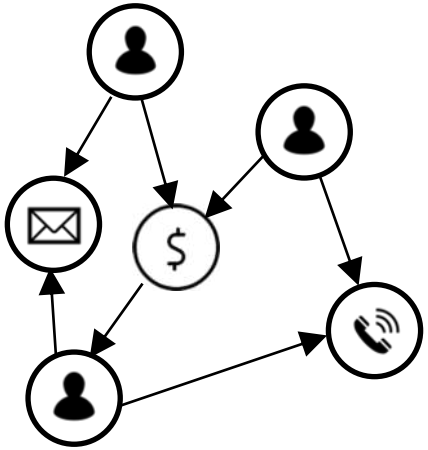
…a set of discrete objects, each of which has some set of relationships with the other objects



Seven Bridges of Konigsberg problem. Leonhard Euler, 1735

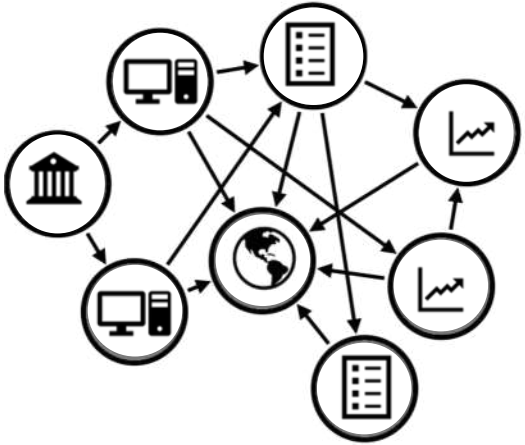
The Rise of Connections in Financial Services

Data connections are increasing as rapidly as data volumes



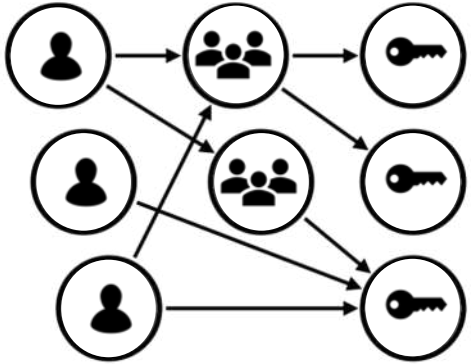
Networks of People/Payments

E.g. AML, Fraud Detection, Compliance Management, Customer 360



Network of Assets

E.g. Risk Management, Compliance Management, Business Insights, What-If Planning, Impact Analysis



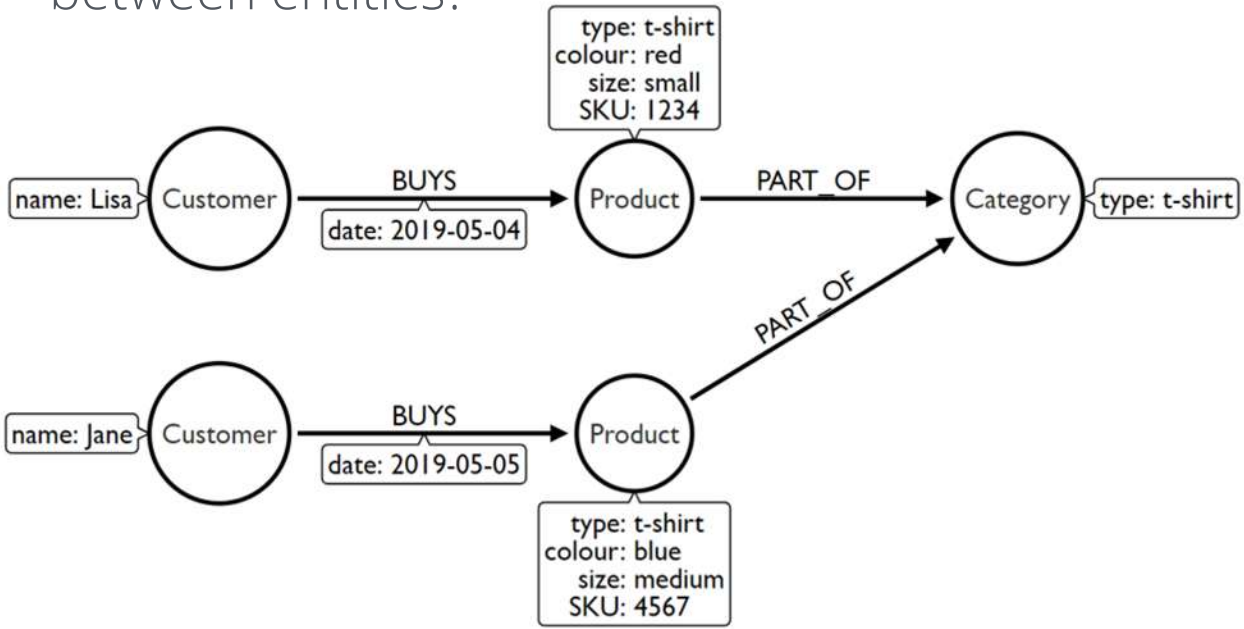
Network of Entitlements

E.g. Entitlement & Identity Management, Data Privacy & Breach Detection

Scenarios for identifying graph-shaped problems

Identifying graph-shaped problems

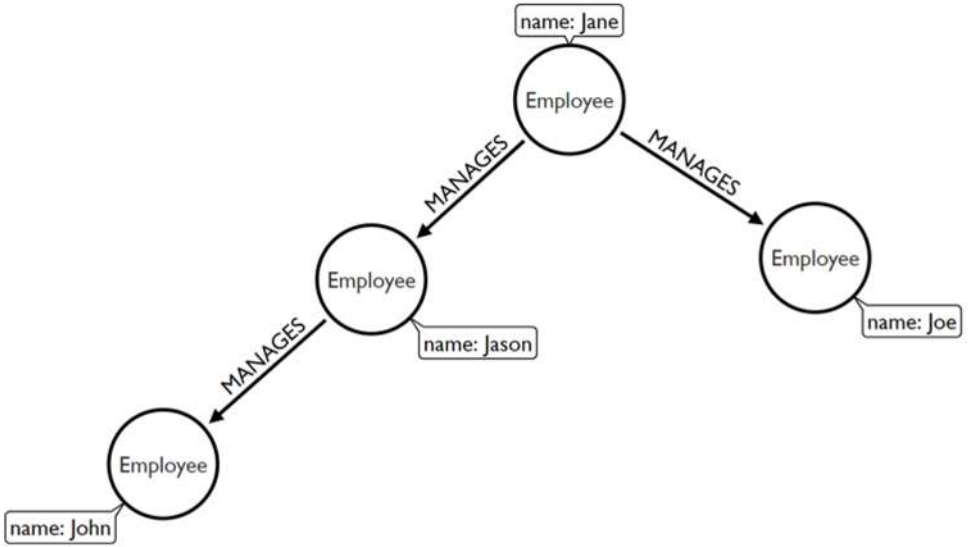
Scenario 1: Does our problem involve understanding relationships between entities?



- Recommendations
- Next best action
- Fraud detection
- Identity resolution
- Data lineage

Identifying graph-shaped problems

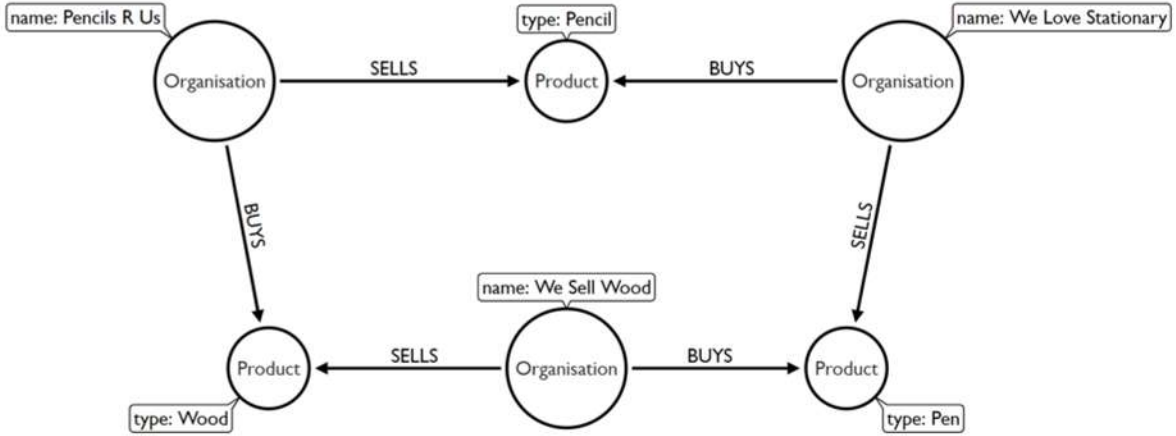
Scenario 2: Does the problem involve a lot of self-referencing to the same type of entity?



- Organisational hierarchies
- Social influencers
- Friends of friends
- Churn detection

Identifying graph-shaped problems

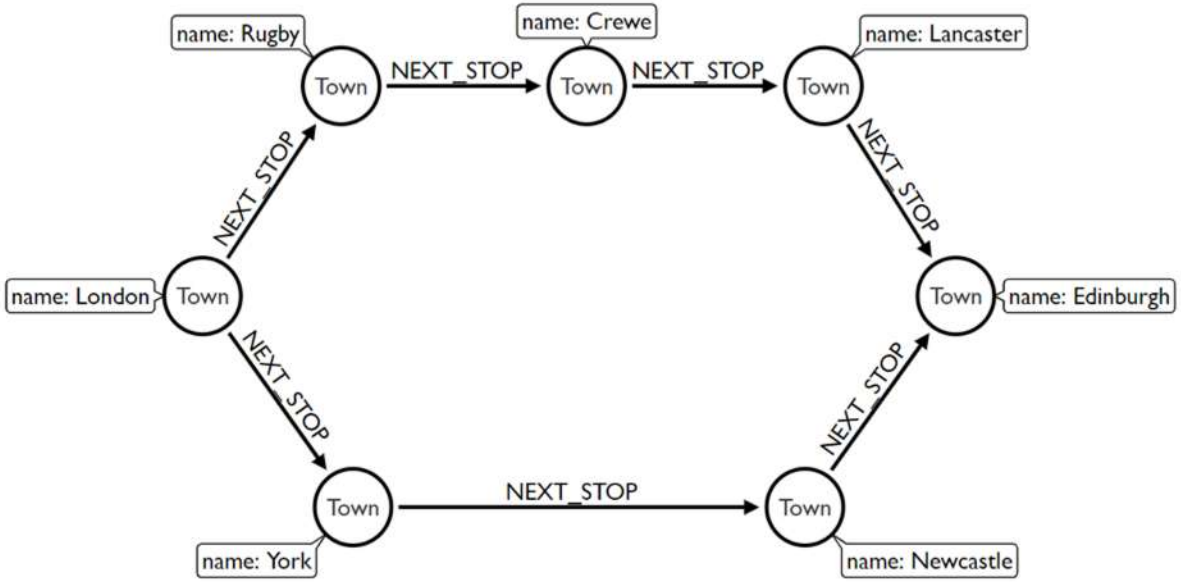
Scenario 3: Does the problem explore relationships of varying or unknown depth?



- Supply chain visibility
- Bill of Materials
- Network management

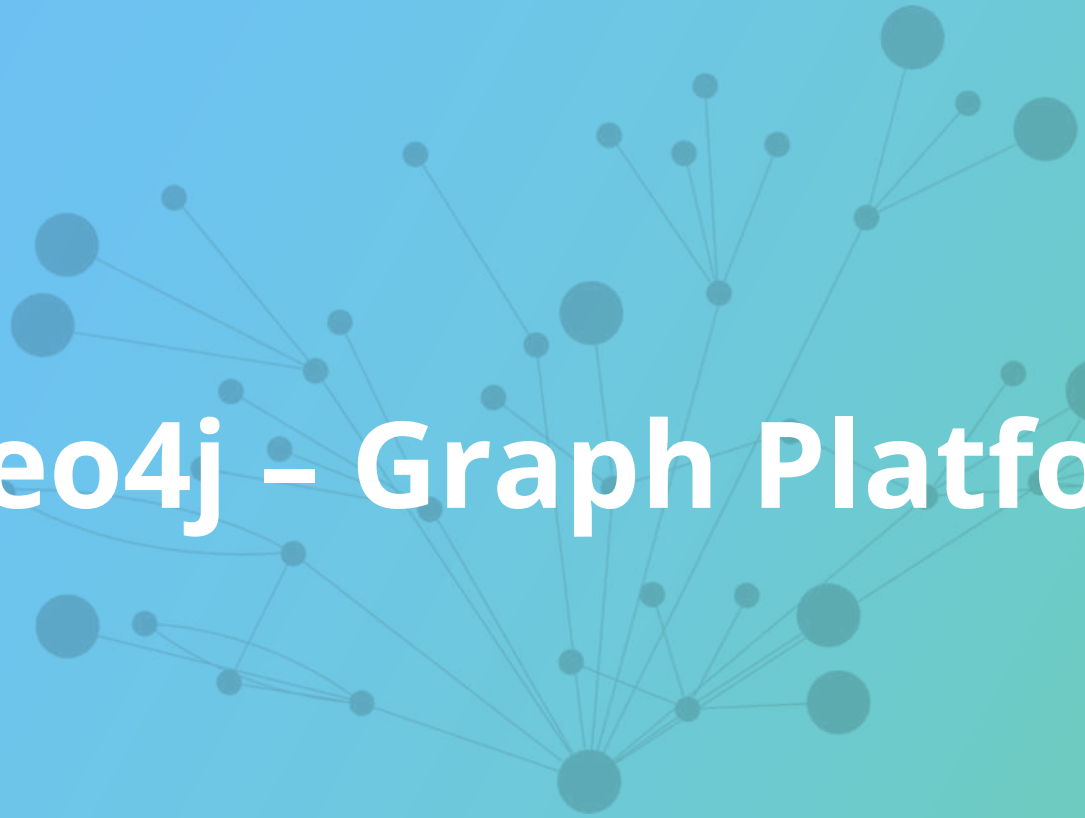
Identifying graph-shaped problems

Scenario 4: Does our problem involve discovering lots of different routes or paths?



- Logistics and routing
- Infrastructure management
- Dependency tracing

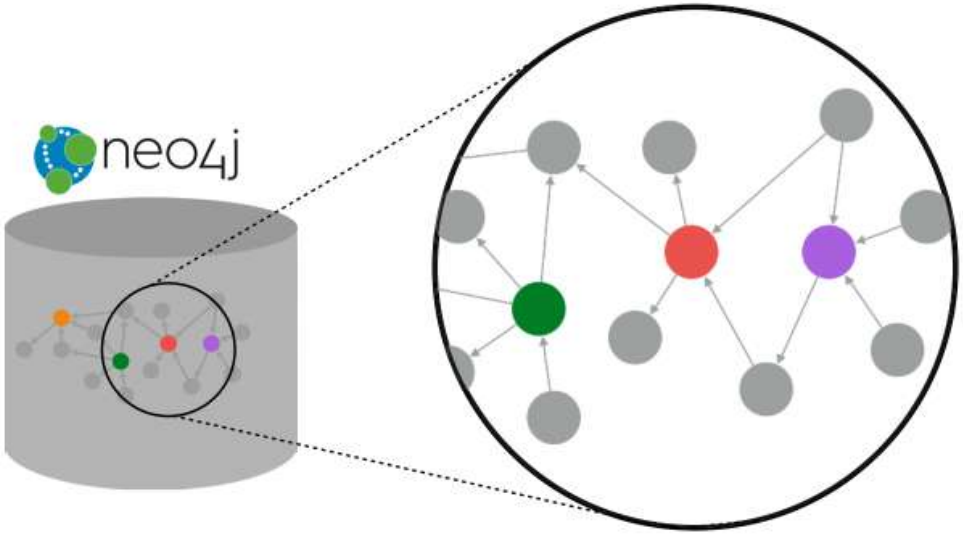
Neo4j – Graph Platform



neo4j

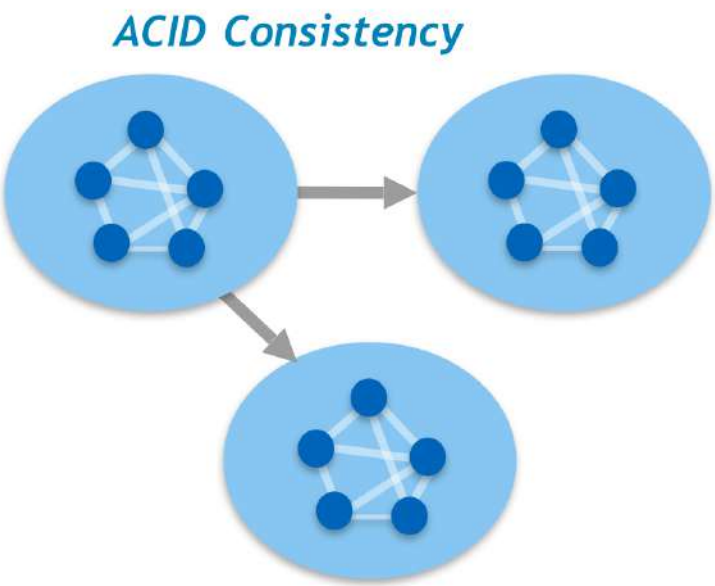
Neo4j Database: Index-free adjacency

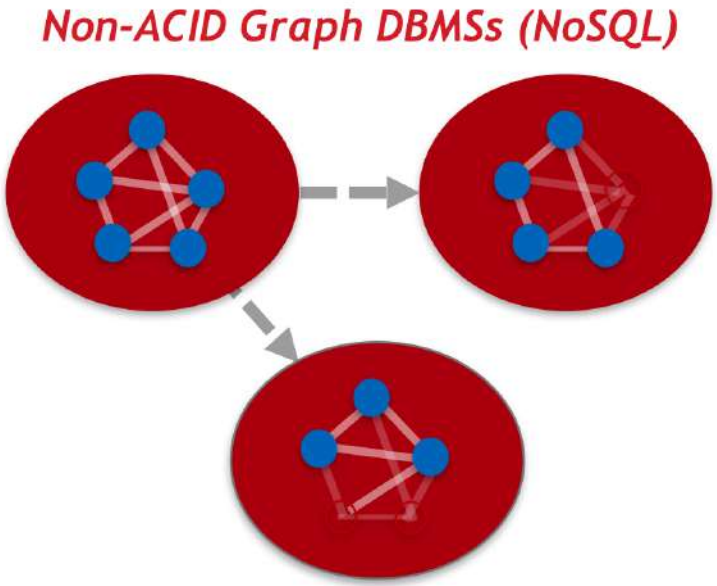
Nodes and relationships are stored on disk as a graph for fast navigational access using pointers.



Neo4j Database: ACID

Transactional consistency - all updates either succeed or fail.





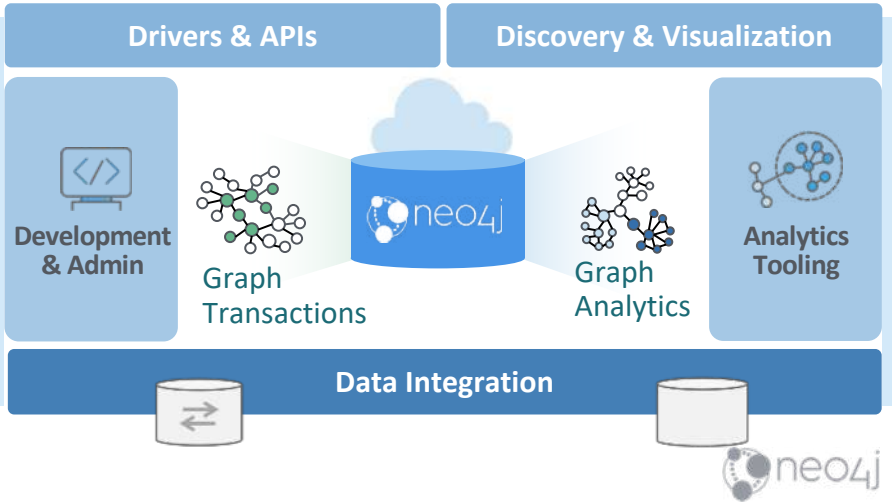
Native Graph Technology

Neo4j is an ***enterprise-grade native graph database and tools***

- Store, reveal and query data relationships
- Traverse and analyze highly connected data in real-time
- Add context and connect data to support emerging AI applications

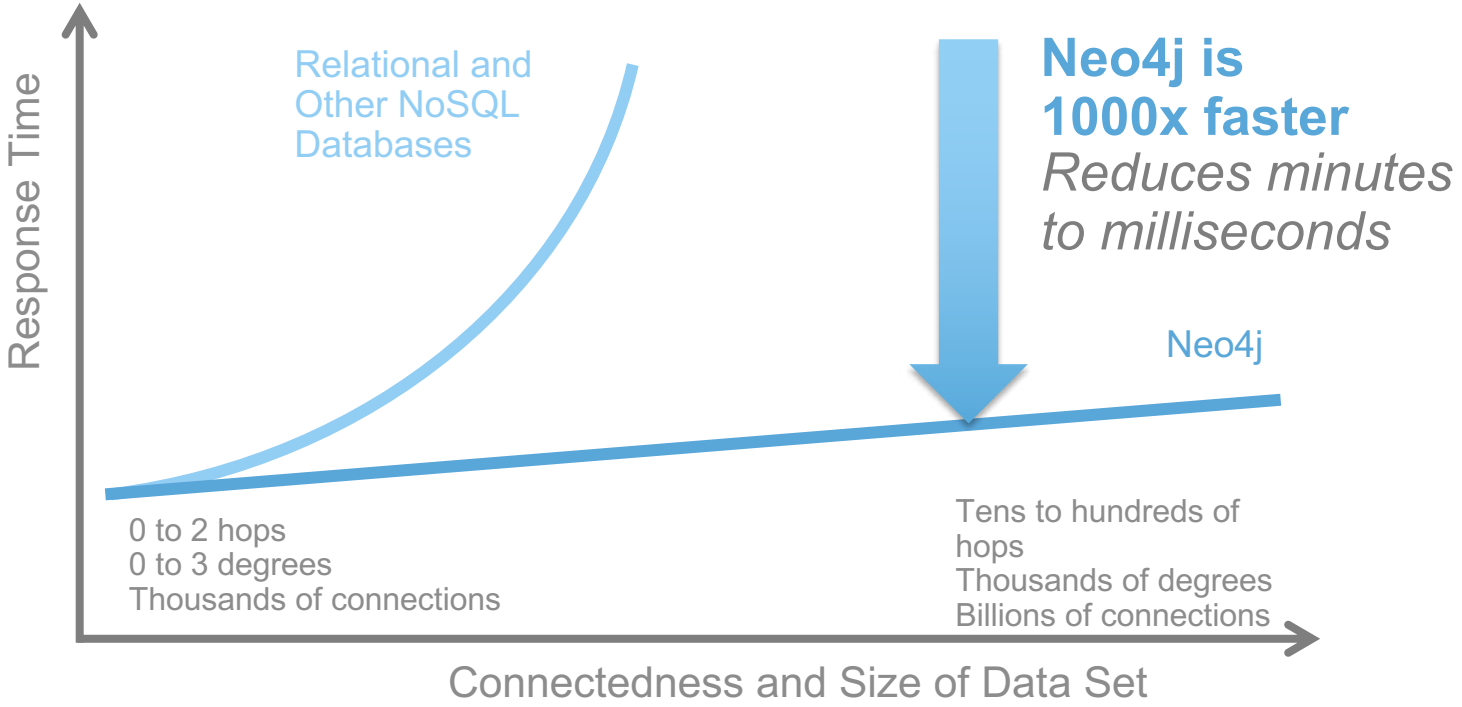
Neo4j Differentiators:

- Performance
- Visualization
- ACID Transactions
- Schema-free Agility
- Graph Data Science
- Global Scale
- Developer Productivity
- Deploy Anywhere

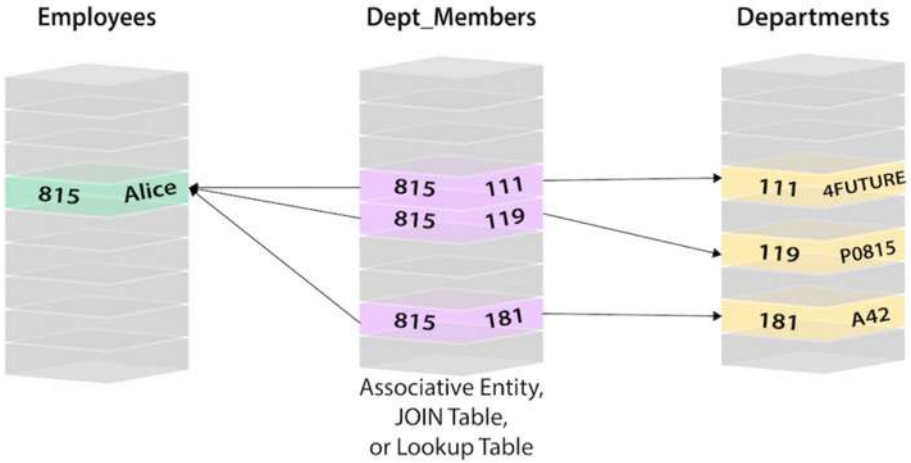


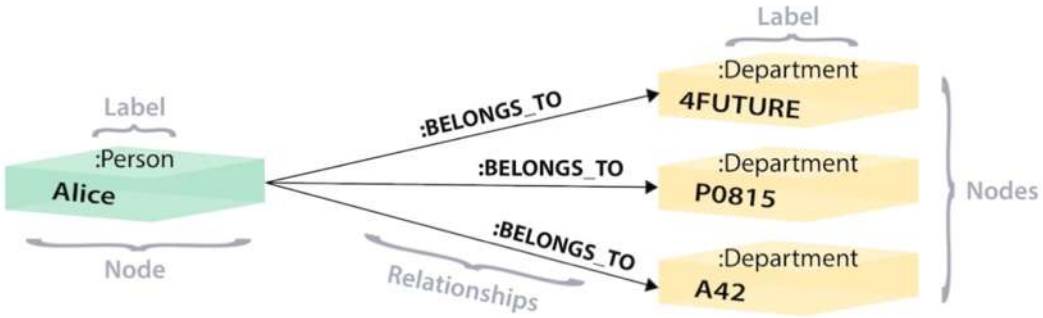
Real-Time Query Performance

Neo4j Versus Relational and Other NoSQL Databases



Relational to Graph






Real Time
Performance

Handling Large Graph Work Loads for Enterprises

Real-time promotion recommendations

- Record “Cyber Monday” sales
- About 35M daily transactions
- Each transaction is 3-22 hops
- Queries executed in 4ms or less
- Replaced IBM Websphere commerce

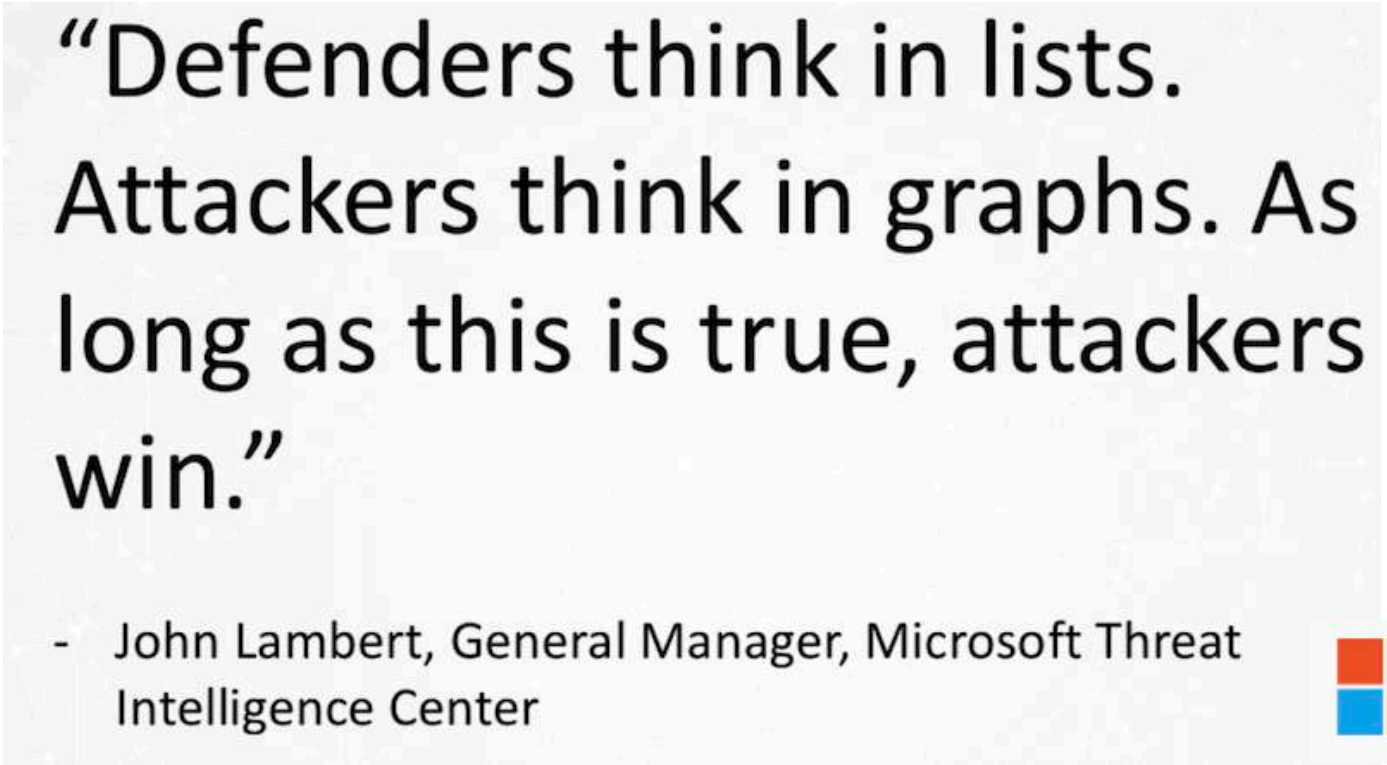
Marriott’s Real-time Pricing Engine

- 300M pricing operations per day
- 10x transaction throughput on half the hardware compared to Oracle
- Replaced Oracle database

Handling Package Routing in Real-Time

- Large postal service with over 500k employees
- Neo4j routes 7M+ packages daily at peak, with peaks of 5,000+ routing operations per second.

Pros use Neo4j for Cyber-security



https://neo4j.com/blog/bloodhound-how-graphs-changed-the-way-hackers-attack/

Productivity Gains with Cypher

The query asks: "Find all direct reports and how many people they manage, up to three levels down"

Example HR Query in SQL

```
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.pid AS directReportees, 0 AS count
  FROM person_reportee manager
  WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
  SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(L2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
SELECT reportee.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT depth1Reportees.pid AS directReportees,
count(depth2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
  FROM(
  SELECT reportee.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT L2Reportees.pid AS directReportees, count(L2Reportees.directly_manages)
AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT L2Reportees.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
)
```

The Same Query using Cypher

```
MATCH (boss)-[:MANAGES*0..3]->(sub),
      (sub)-[:MANAGES*1..3]->(report)
WHERE boss.name = "John Doe"
RETURN sub.name AS Subordinate,
  count(report) AS Total
```

Project Impact

Less time writing queries

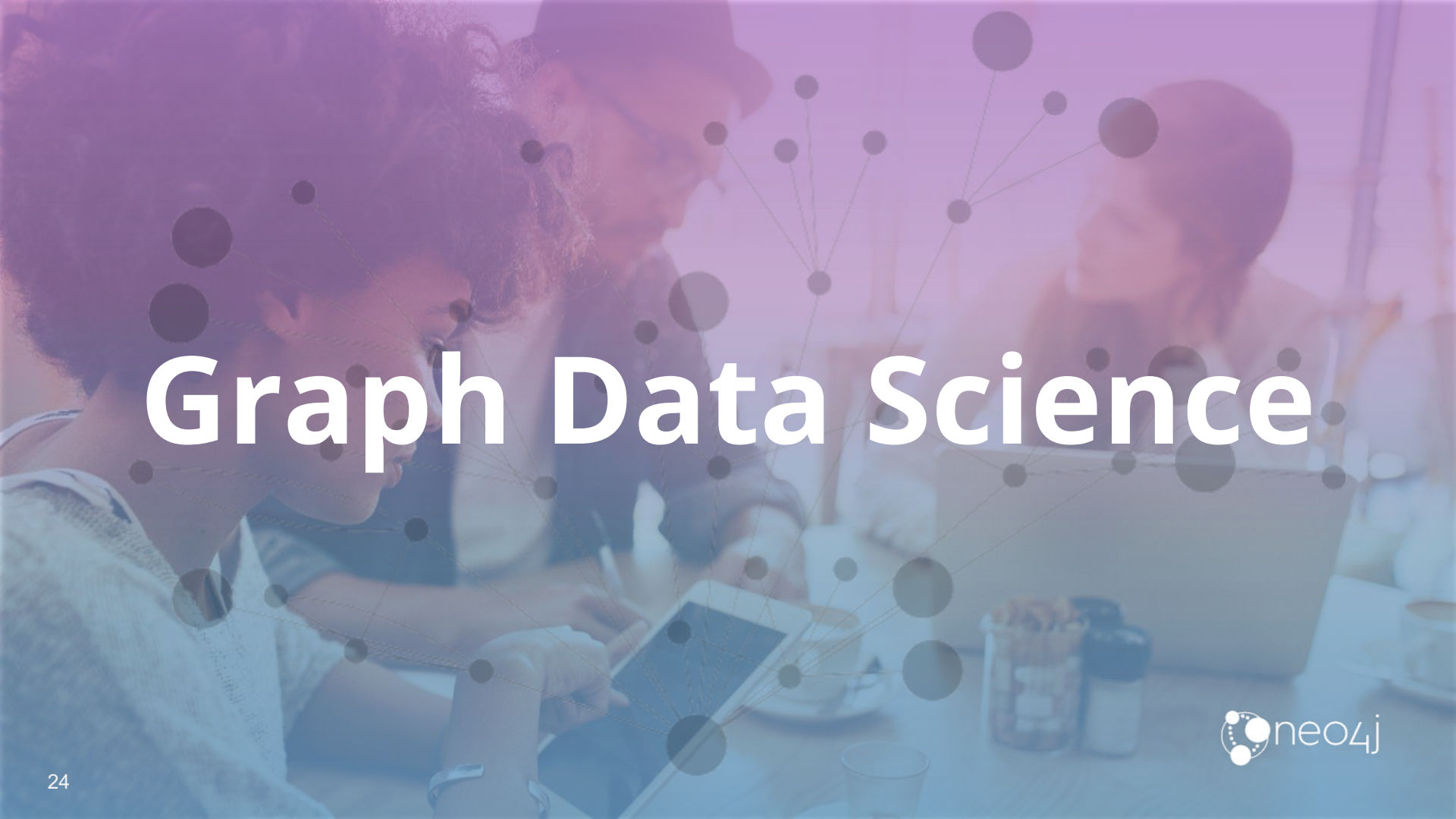
- More time understanding the answers
- Leaving time to ask the next question

Less time debugging queries:

- More time writing the next piece of code
- Improved quality of overall code base

Code that's easier to read:

- Faster ramp-up for new project members
- Improved maintainability & troubleshooting


Graph Data Science

Improving Analytics, ML & AI Across Industries

AstraZeneca Patient Journeys

- Early intervention project with 3 yrs of visits, tests & diagnosis with 10's of Bn of records
- Finding similarities in patient journeys
- Graph algorithms for identifying communities & best intervention points

Meredith Marketing to the Anonymous

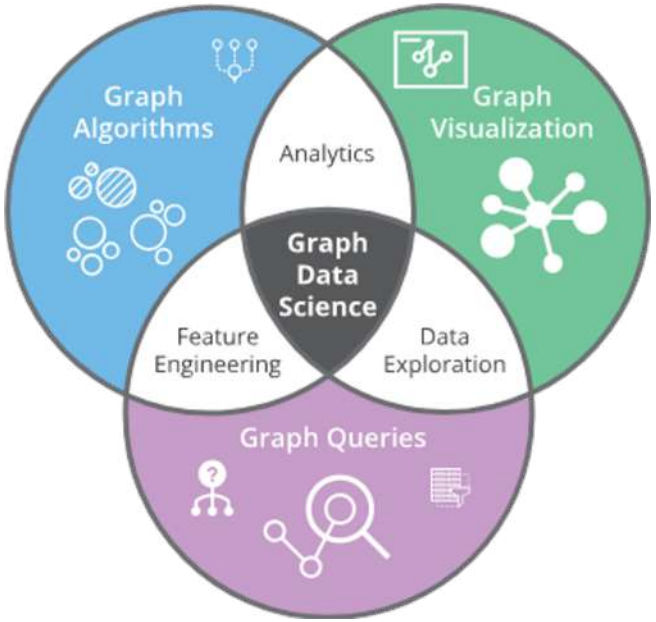
- Mostly anonymous users across devices and sites with ever changing cookies
- 4.4 TB: +14 Bn nodes +20Bn relationships
- +160 Mn rich, unique profiles created
- 612% Increase in visits per profile

Fraud Detection & Asset Recovery

Multiple Banks

- Majority of credit card fraud went undetected
- Millions of account with billions of transactions
- Graph analytics with queries & algorithms help find $10's of millions of fraud in 1st year

What is *Graph* data science?



Graph Data Science is a science-driven approach to gain knowledge from the relationships and structures in data, typically to power predictions.

Data scientists use relationships to answer questions.

Knowledge Graph Queries
e.g. Detecting Financial Fraud

Improving existing pipelines to identify fraud via heuristics

Deceptively Simple Queries

How many flagged accounts are in the applicant's network **4+ hops out**?

How many **login / account variables in common**?

Add these metrics to your approval process

Difficult for RDBMS systems over 3 hops

So, When Do I Need Graph Algorithms?

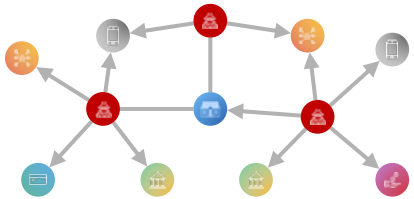
Query (e.g. Cypher/Python)

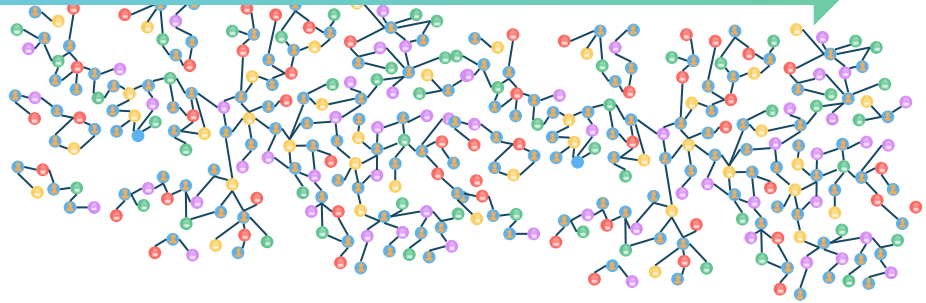
Real-time, local decisioning and pattern matching

Graph Algorithms

Global analysis and iterations

Local Patterns ⟵ ⟶ **Global Computation**





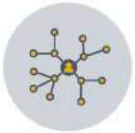
You know what you're looking for and making a decision

You're learning the overall structure of a network, updating data, and predicting

The Neo4j Graph Data Science Library

Pathfinding & Search

- Deep path analytics
- Optimal routing



Centrality / Importance

- Identifies node importance
- Influencer & Risk Identification

Community Detection

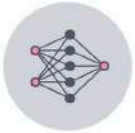
- Detects group clustering
- Partition options

Link Prediction

- Estimates likelihood of
- Estimate missing information

Similarity

- Evaluates how alike nodes are
- Construct graphs from data

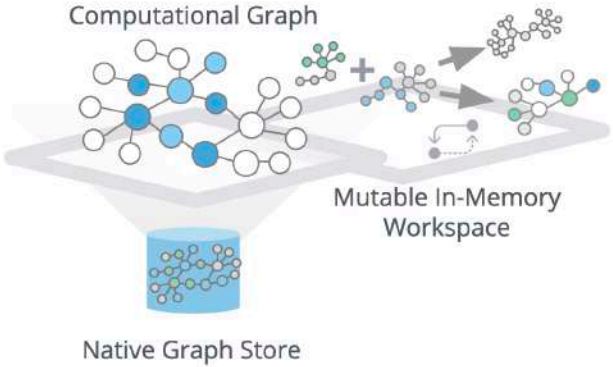


Graph Embeddings

- Learn your graph topology
- Use for dimensionality reduction

50+ Robust Algorithms

Flexible Analytics Workspace



Graph Algorithms
e.g. Detecting Financial Fraud

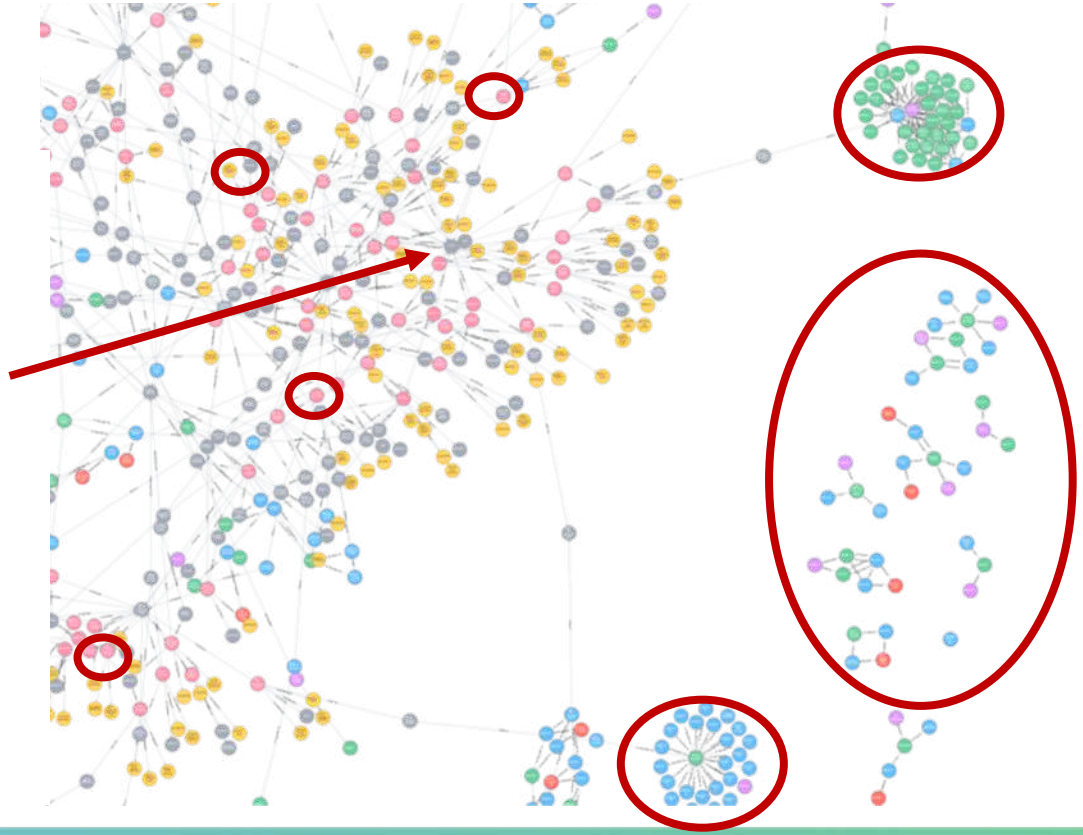
Graph algorithms enable reasoning about **network structure**

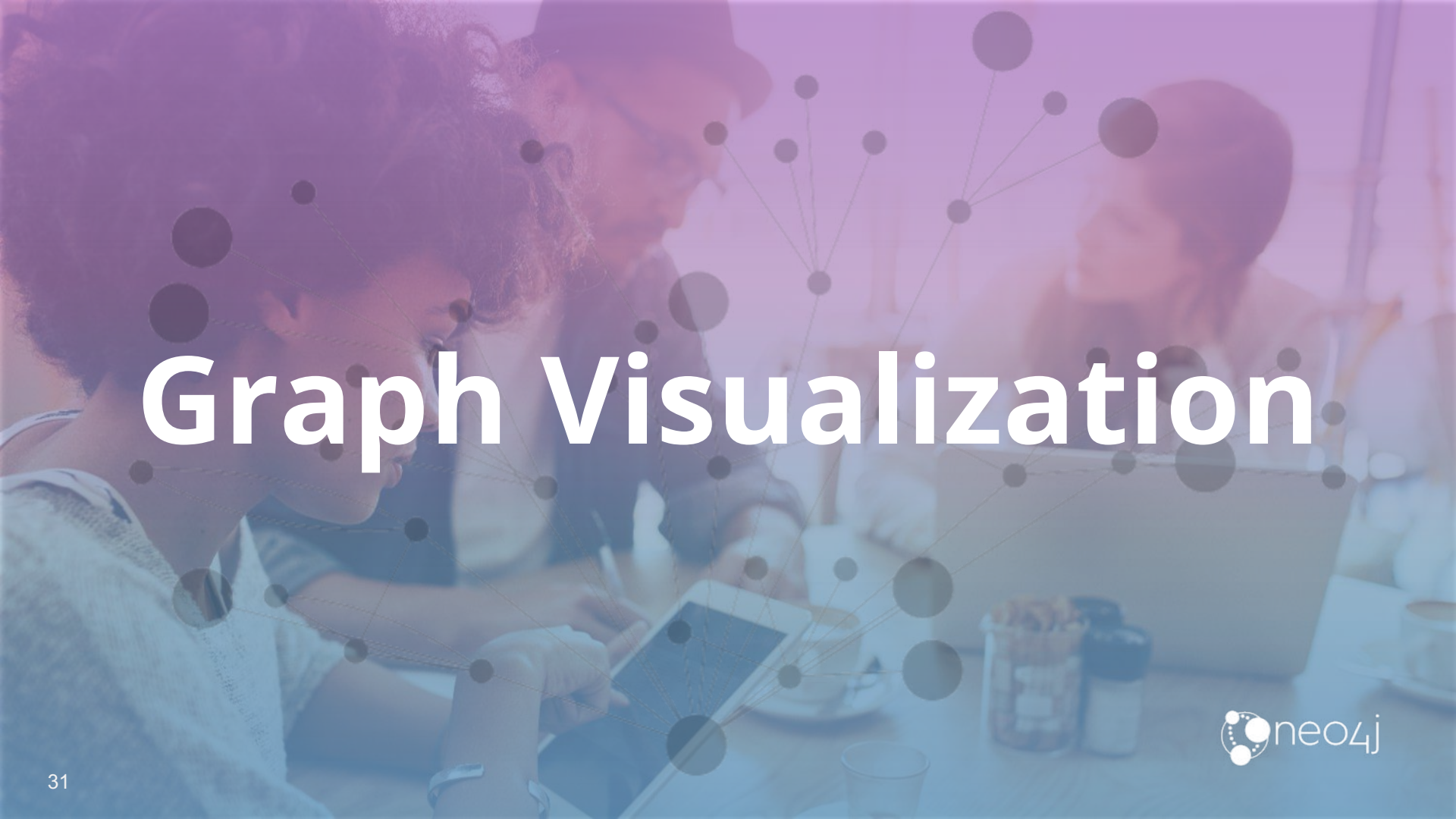
Connected components identify disjointed group sharing identifiers

PageRank to measure influence and transaction volumes

Louvain to identify communities that frequently interact

Jaccard to measure account similarity




Graph Visualization

Neo4j Bloom's Intuitive User Interface

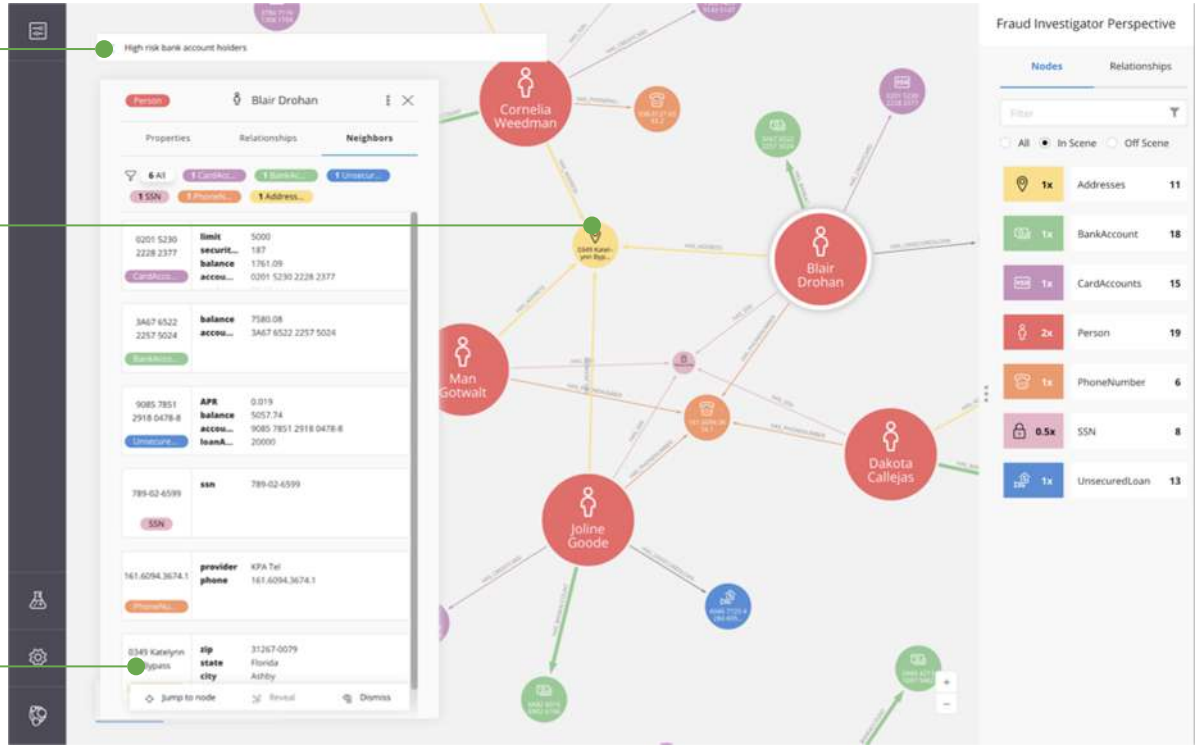
Search with type-ahead suggestions

Flexible Color, Size and Icon schemes

Visualize, Explore and Discover

Pan, Zoom and Select

Property Browser and editor




Neo4j Ecosystem

Native Graph Technology for Applications & Analytics

Applications

Business Users

Drivers & APIs

Discovery & Visualization

Developers

Admins

Development & Admin

Graph Transactions

neo4j

APOC/Stored Procedures

Graph Analytics

Analytics Tooling

Data Analysts

Data Scientists

BIConnector

Data Integration

kafka

Enterprise Data Hub

NEOSEMANTICS

neo4j

Neo4j Drivers

Languages

- Java, JavaScript, .NET, Python, Go, R, Ruby, PHP, Erlang, Perl

Driver modes

- Simple
- Asynchronous
- Reactive (back-pressure and flow control)

Transaction routing

- Route request to appropriate server based on server load and if read or write operation

Uses

- Cypher based graph queries
- Coding Environments
  - Juypter, Colab, RSuite
- Engine API custom procedures, functions
  - Traversals, injections, etc.
- Extension to Graph Data Science Library (e.g. Pregel)

Neo4j BI Connector

The most popular BI tools can now talk live to the world's most popular graph database

- Best live, seamless integration of graph data with your favorite BI tools
  - Familiar UI for end users
  - No development effort for IT
- Democratizes access to Neo4j data
- Free to adopt by BI teams of Enterprise Edition customers