



The logo features the word "graphsummit" in a large, white, sans-serif font. Above the "g", there is a white icon consisting of three stylized human figures connected by lines, forming a network. Below the main title, the words "ASIA PACIFIC" are written in a smaller, white, all-caps sans-serif font.

Building a Knowledge Centric Fraud Detection Graph using Neo4j and Python

Introduction



Neo4j Graph Data Platform





graphsummit

Neo4j Graph Data Platform

Native Graph Database

The core component of Neo4j platform

Graph Query Language

Cypher and GQL as the lingua franca for graphs

Data Science and Analytics

Tools, algorithms, and Integrated ML framework

Application Development Tools & Frameworks

Tools and APIs for rapid prototyping and development

Discovery & Visualisation

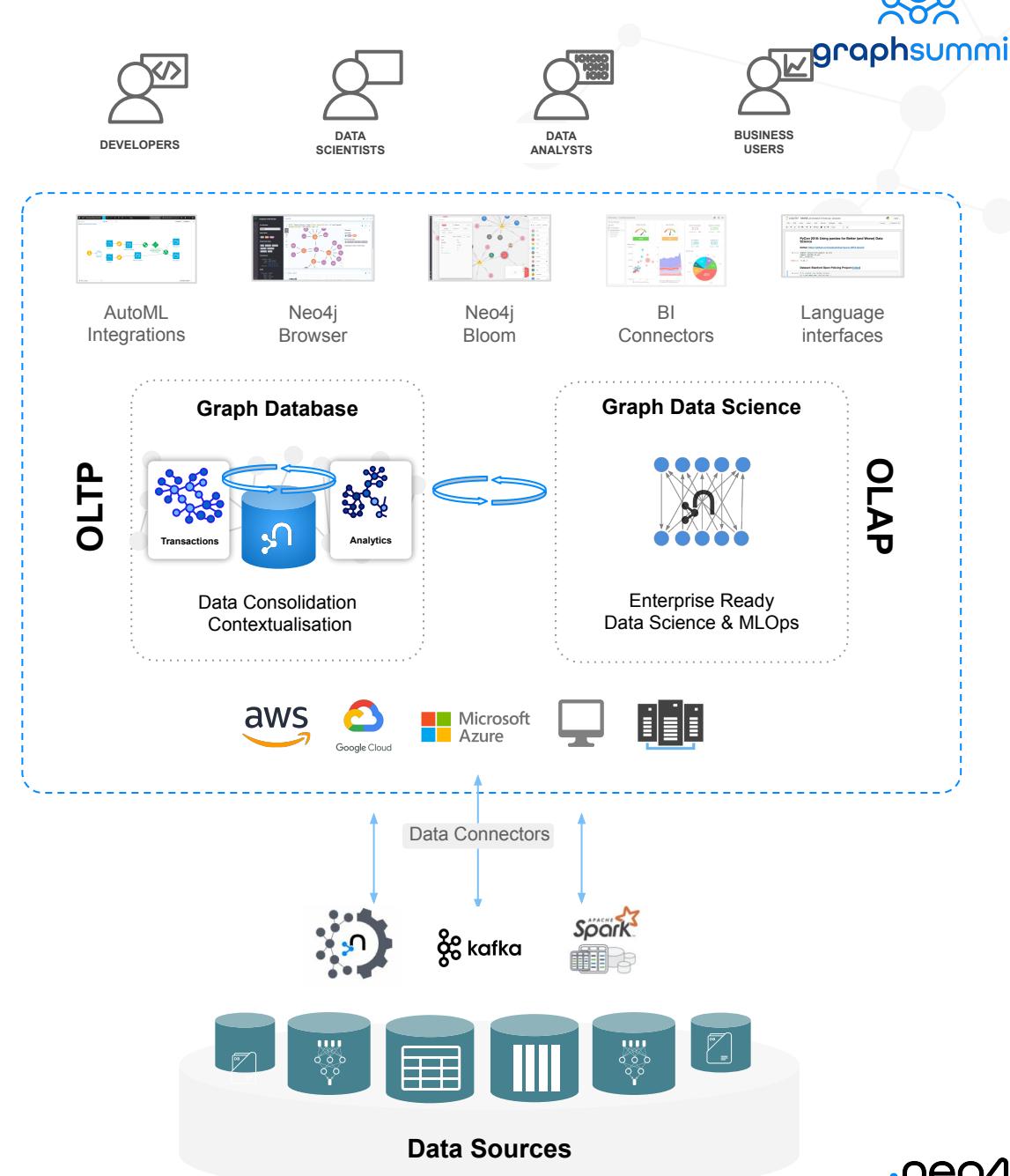
Low-code querying, data modeling and exploration tools

Ecosystem & Integrations

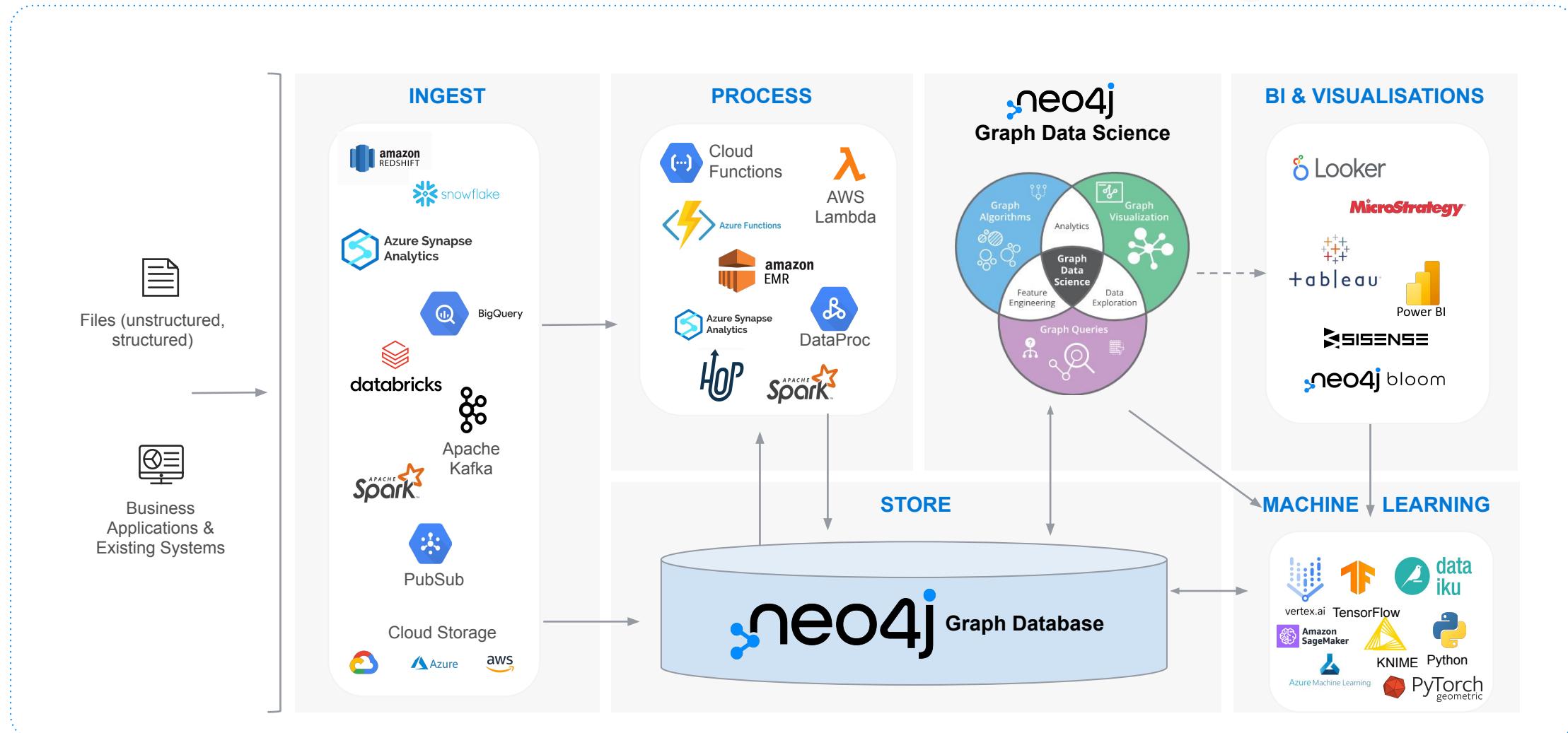
Rich set of connectors to plug into existing data ecosystems

Runs Anywhere

Run by yourself or as DBaaS by Neo4j, in the cloud or on premises



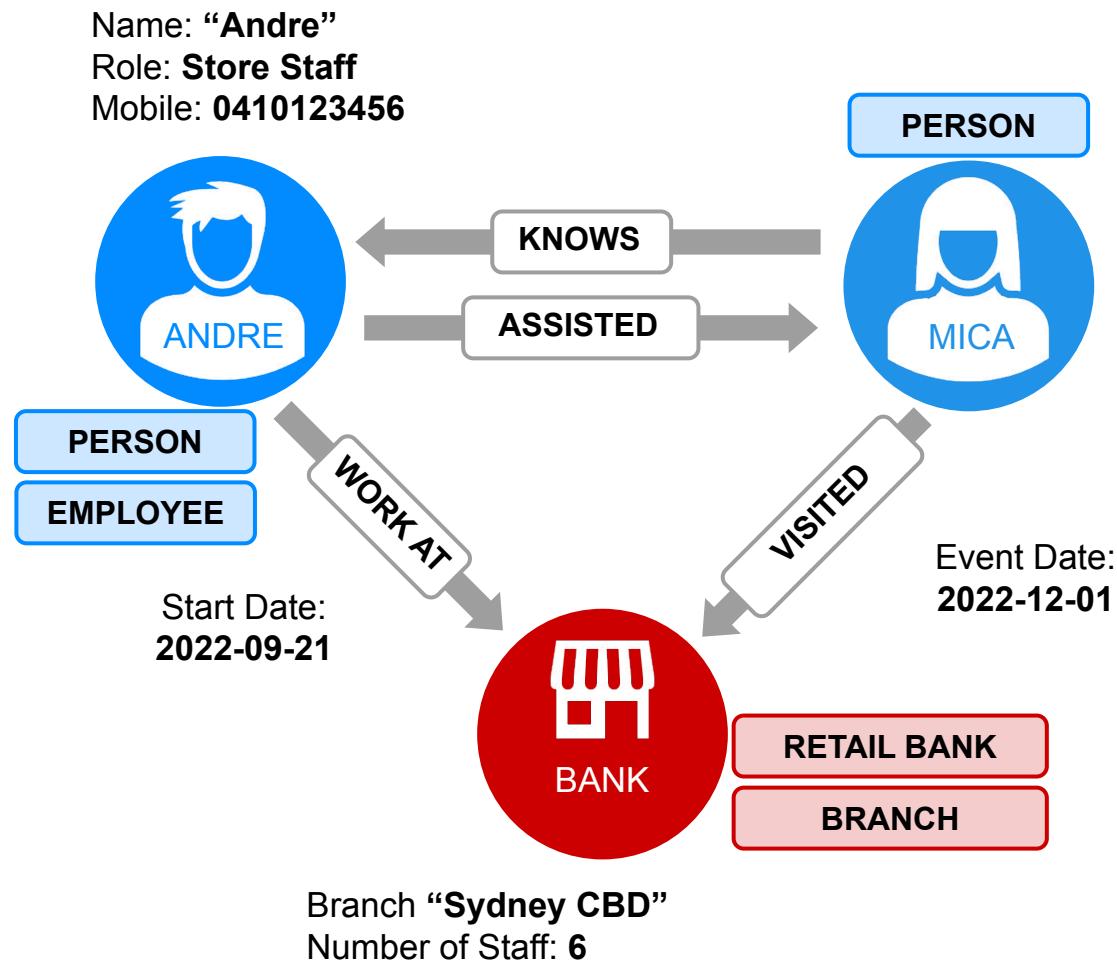
Neo4j Architecture and Deployment



Graph Fundamentals



Graph Building Blocks



Node

Represents an entity in the graph

Relationship

Connect nodes to each other

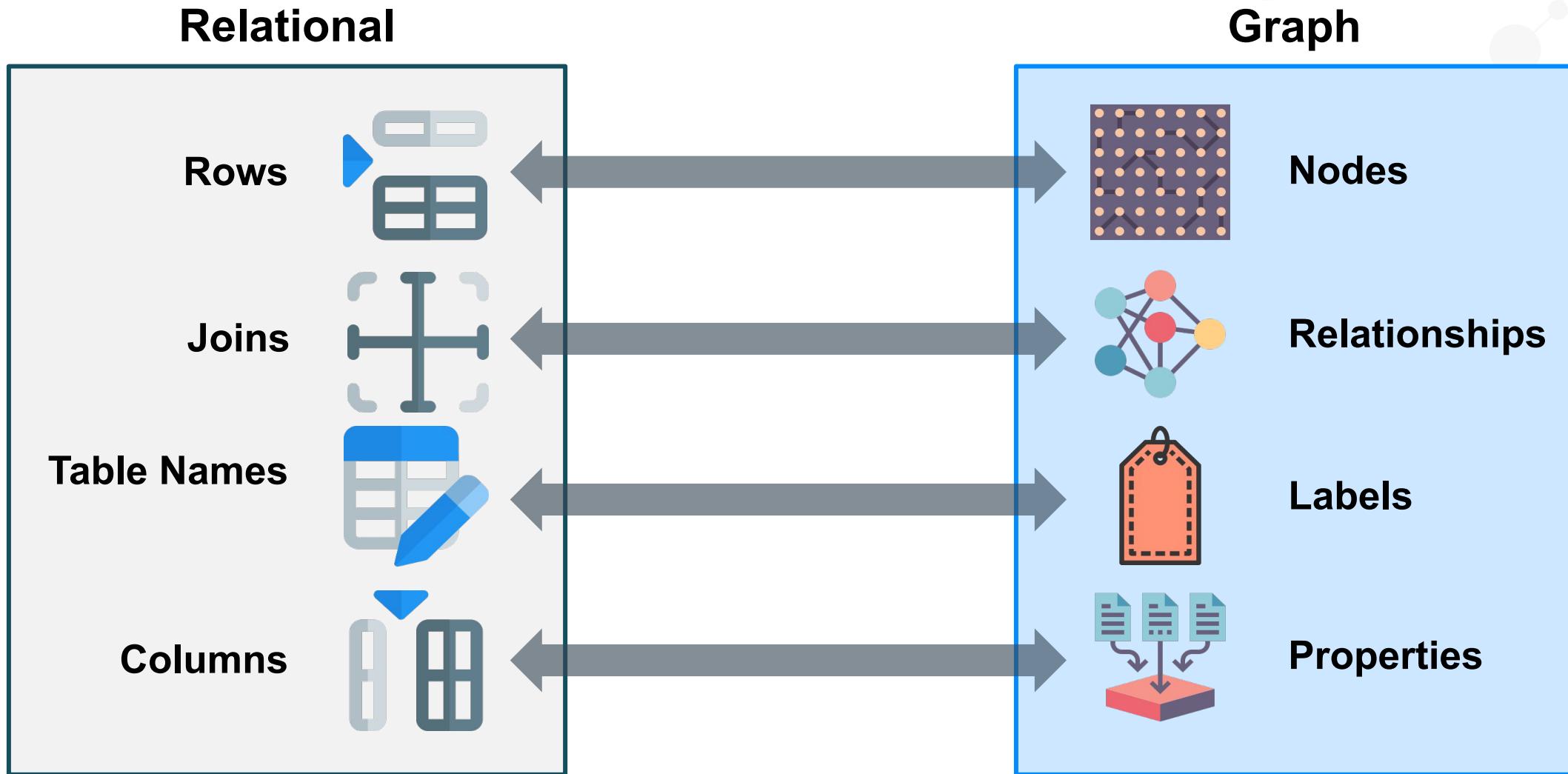
Label

Grouping of similar nodes

Property

Describes a node or relationship: e.g. name, age, address

Conceptual Mapping Relational → Graph



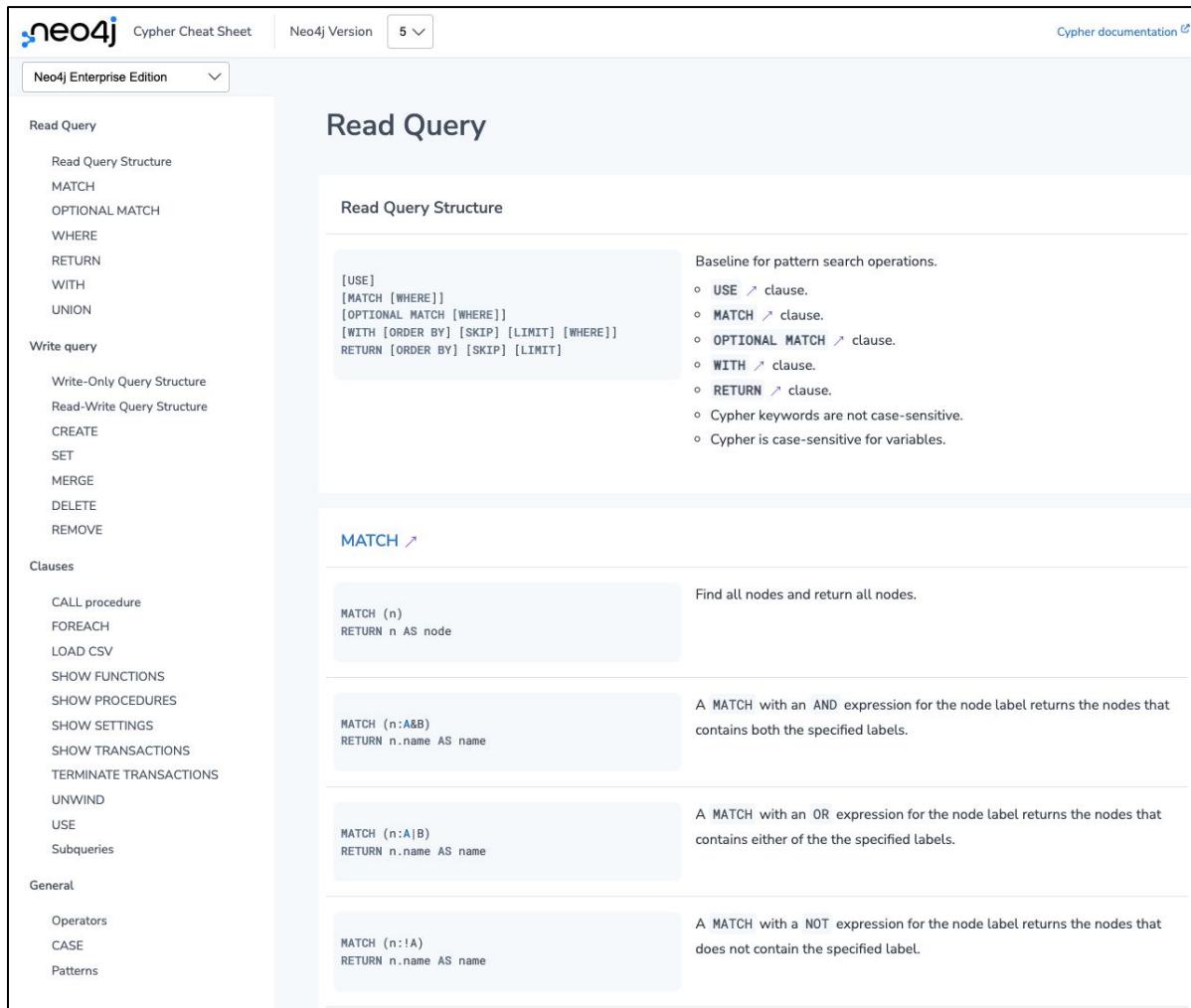
Graph (Cypher) Query Language

Fundamentals



Cypher Cheat Sheet

Cypher Cheat Sheet Link



The screenshot shows the Neo4j Cypher Cheat Sheet interface. The top navigation bar includes the Neo4j logo, "Cypher Cheat Sheet", "Neo4j Version 5", and a "Cypher documentation" link. A dropdown menu on the left indicates "Neo4j Enterprise Edition". The main content area is titled "Read Query" and contains a sidebar with sections like "Read Query Structure", "Match", "Optional Match", "Where", "Return", "With", "Union", "Write query", "Write-Only Query Structure", "Read-Write Query Structure", "Create", "Set", "Merge", "Delete", "Remove", "Clauses", "General", "Operators", "Case", and "Patterns". The "Match" section is expanded, showing examples of basic matching and more complex patterns involving AND, OR, and NOT operators.

Read Query

Read Query Structure

Baseline for pattern search operations.

- USE ↗ clause.
- MATCH ↗ clause.
- OPTIONAL MATCH ↗ clause.
- WITH ↗ clause.
- RETURN ↗ clause.
- Cypher keywords are not case-sensitive.
- Cypher is case-sensitive for variables.

MATCH ↗

Find all nodes and return all nodes.

```
MATCH (n)
RETURN n AS node
```

A MATCH with an AND expression for the node label returns the nodes that contains both the specified labels.

```
MATCH (n:A&B)
RETURN n.name AS name
```

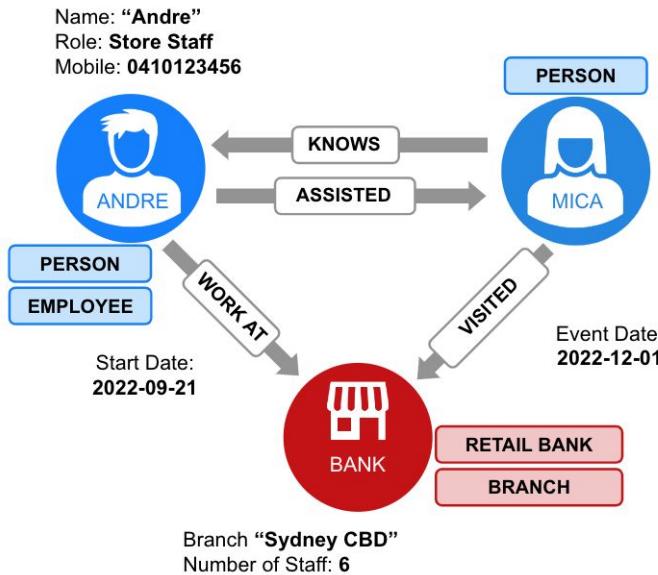
A MATCH with an OR expression for the node label returns the nodes that contains either of the the specified labels.

```
MATCH (n:A|B)
RETURN n.name AS name
```

A MATCH with a NOT expression for the node label returns the nodes that does not contain the specified label.

```
MATCH (n:!A)
RETURN n.name AS name
```

Basic Cypher (GQL) Syntax (1/3)



MATCH – nodes and relationship graph pattern matching (GPM)

WHERE – filter node or relationship properties, labels or GPM

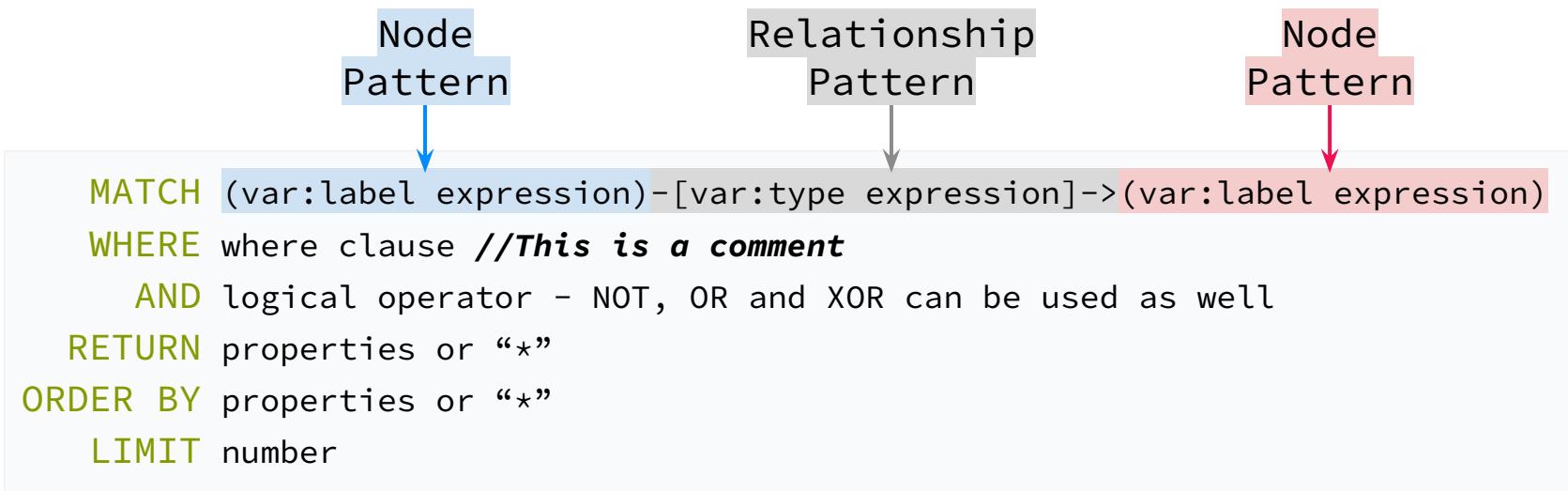
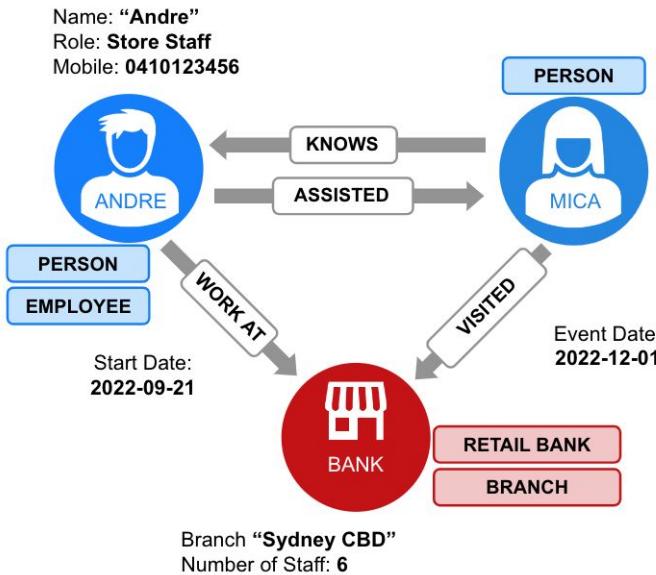
RETURN – return graph path, nodes, relationship properties or all (“*”)

ORDER BY – order set of properties in ascending or descending order

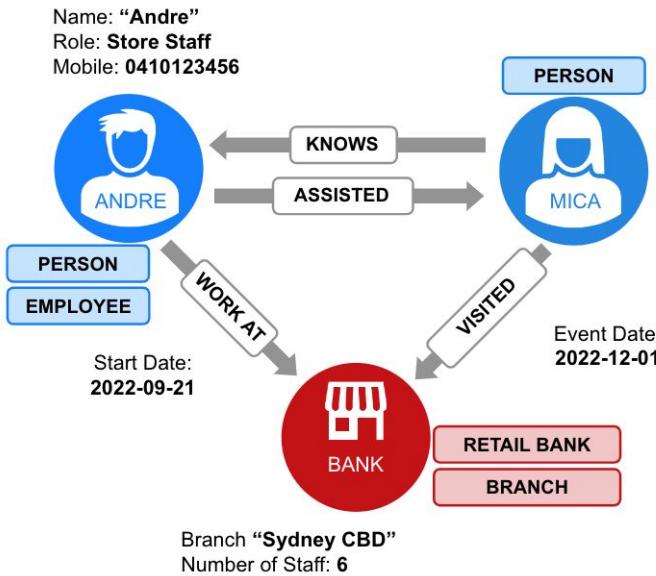
LIMIT – limit the number of nodes and relationships



Basic Cypher (GQL) Syntax (2/3)



Basic Cypher (GQL) Syntax (3/3)



Node Pattern: (p:Person)

Relationship Pattern: -(r:WORK_AT)->(b:Branch)

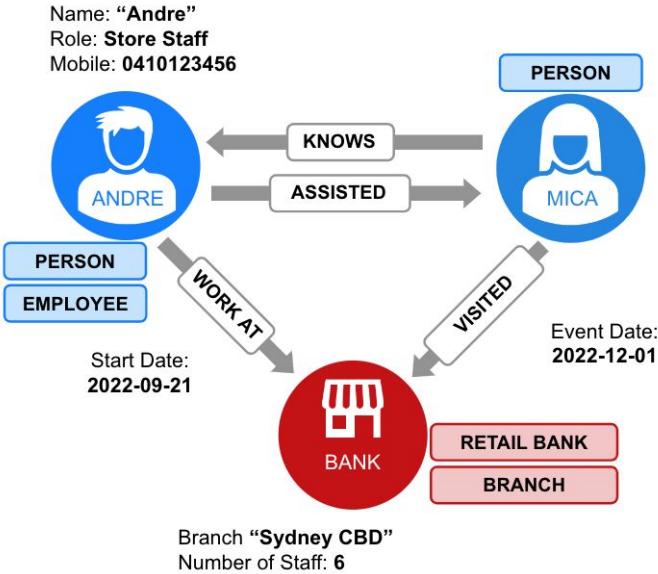
Node Pattern: (b:Branch)

```

    MATCH (p:Person)-[r:WORK_AT]->(b:Branch)
    WHERE p.name = "Andre" // This is a comment
        AND r.startDate = '2022-09-21'
    RETURN *
    ORDER BY b.branch
    LIMIT 1
  
```

WITH Clause Syntax

The WITH clause is similar to RETURN. It allows query parts to be chained together, piping the results from one to be used as starting points or criteria in the next.



```

MATCH (p:Person)-[r:WORK_AT]->(b:Branch)

// Only variables declared in the WITH can be referenced later in the query
WITH b, COUNT(p) AS numEmployees

WHERE 1 < numEmployees < 4
OR b.branch STARTS WITH "Sydney"

RETURN *
  
```

UNWIND Clause Syntax

The UNWIND clause **expands a list into a sequence of rows**. Multiple UNWIND clauses can be chained to unwind nested list elements.

Inputs		UNWIND Syntax	Output	
id	roles		id	role
1	[“READ”, “WRITE”, “DELETE”]	<code>WITH \$Inputs AS input</code> <code>// Expand list of roles into sequence of role.</code>	1	READ
2	[“READ”]	<code>UNWIND input.roles AS role</code>	1	WRITE
3	[“READ”, “WRITE”]	<code>RETURN input.id AS id</code> <code>,input.role AS role</code>	1	DELETE
4	[“READ”]		2	READ
5	[“READ”, “WRITE”, “DELETE”]		3	READ
6	[“READ”, “WRITE”]	

CONSTRAINT Syntax

Ensures that node and relationship property value are **unique for specific node label or relationship type**. It **automatically adds an index** on that property.

CONSTRAINT Syntax

// Unique node property constraints.

```
CREATE CONSTRAINT constraint_name FOR (p:Client) REQUIRE p.id IS UNIQUE;
```

// Unique relationship property constraints.

```
CREATE CONSTRAINT constraint_name FOR ()-[r:LIKED]-() REQUIRE r.date IS UNIQUE;
```

// Node key constraints.

```
CREATE CONSTRAINT constraint_name FOR (p:Client) REQUIRE (p.id, p.name) IS NODE KEY;
```

LOAD CSV Syntax

To import data from a CSV file into Neo4j, you can use **LOAD CSV** to get the data into your query. Then, you write it to your database using the normal updating clauses of Cypher.

```
LOAD CSV Syntax
// Load CSV with headers based on the file URL.
LOAD CSV WITH HEADERS

// Each line the csv file will be assigned in the variable "row".
FROM "https://url/clients.csv" AS row
WITH row

// CSV fields can be access using the variable.<field name> e.g. row.ID
MERGE (c:Client { id: row.ID })
  SET c.name = row.NAME
  ,c.isFraud = toBoolean(row.ISFRAUD)
```

MERGE Clause Syntax (1/3)

The MERGE clause **create (if it doesn't exist) or update nodes, relationships or properties** based on node key (primary key). It is similar to SQL UPSERT clause.

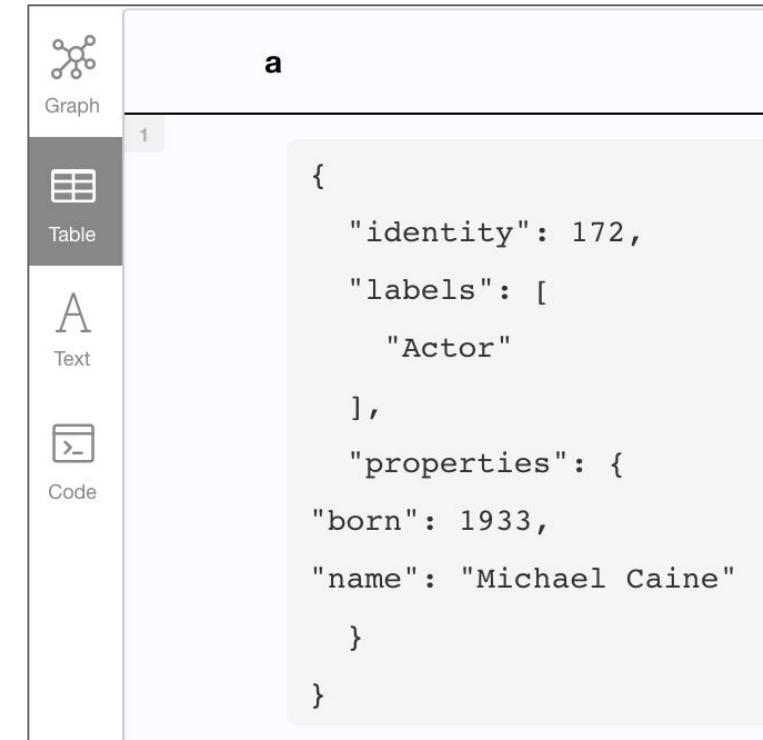
MERGE Syntax

```

MERGE (a:Actor {name: 'Michael Caine'})

// Create 'Michael Caine' node and set born
// property if it doesn't exists.
// Else, just set the born property.
  SET a.born = 1933

RETURN a
  
```



MERGE Clause Syntax (2/3)

The MERGE clause allows you to **specify CREATE or MATCH behavior when merging**.

MERGE Syntax

```

MERGE (a:Person {name: 'Sir Michael Caine'})

// Properties to be set when a new node is
// created.
ON CREATE SET a.born = 1934
      ,a.birthPlace = 'UK'

// Properties to be set when a existing node
// matches.
ON MATCH SET a.birthPlace = 'UK2'

RETURN a
  
```



Output when the node already exists

	a.name	a.born	a.birthPlace
Table			
A			
1	"Sir Michael Caine"	1934	"UK2"
Text			

MERGE Clause Syntax (3/3)

The MERGE clause **applies to relationships** as well.

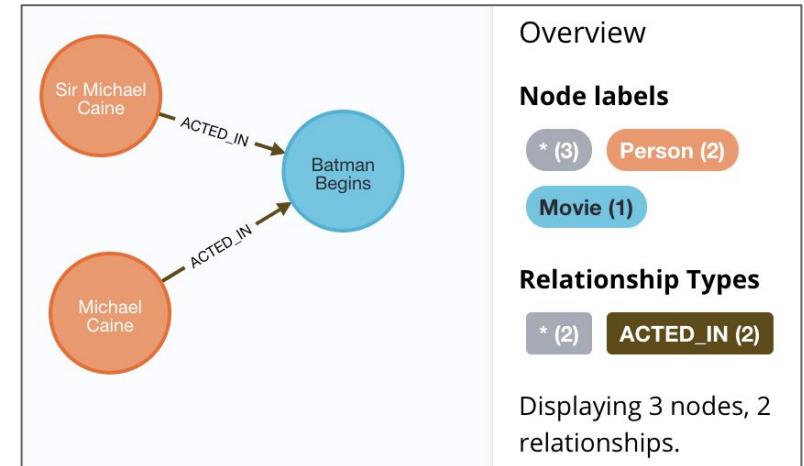
MERGE Syntax

```

MATCH (p:Person), (m:Movie)
WHERE m.title = 'Batman Begins'
  AND p.name ENDS WITH 'Caine'
MERGE (p)-[r:ACTED_IN]->(m)
ON CREATE SET r.role = 'New'
ON MATCH SET r.role = 'Existing'
RETURN p, m
  
```



Output when the relationship doesn't exist



Neo4j APOC Library

APOC stands for Awesome Procedures on Cypher. It is standard utility library for common procedures and functions.

APOC Syntax

```
// General APOC Syntax
CALL <package>.<subpackage>.<procedure>(<argument1>,<argument2>, ...);

// Returns the information stored in the transactional database statistics
CALL apoc.meta.stats()
YIELD nodeCount, labels;

// Creates a list of lists of adjacent elements in a list, skipping the last item.
// Returns [[1,2],[2,3]].
CALL apoc.coll.pairsMin([1,2,3]) YIELD value;
```

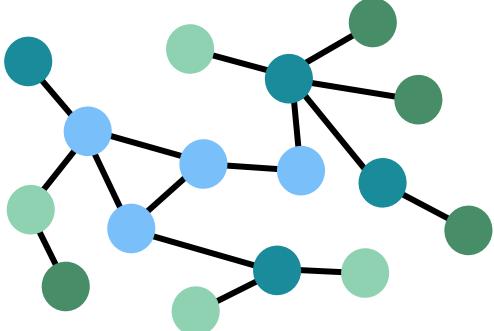
Neo4j Graph Data Science (GDS)



Graph and Data Science

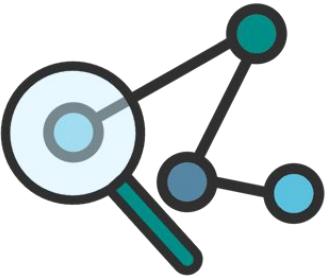
Graph Native Machine Learning

Knowledge Graphs

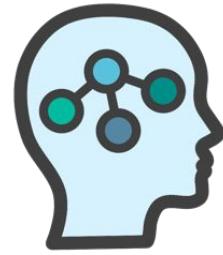


Find the patterns you're looking for in connected data

Graph Algorithms



Use unsupervised machine learning techniques to identify associations, anomalies, and trends.



Use embeddings to learn the features in your graph that you don't even know are important yet.

Train in-graph supervised ML models to predict links, labels, and missing data.

Graph Data Science (GDS) Family of Algorithms

Graph algorithms are a set of instructions that visit the nodes of a graph to analyse the relationships in connected data.



**Pathfinding &
Search**



**Centrality &
Importance**



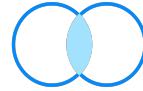
**Community
Detection**



**Supervised
Machine Learning**



**Heuristic Link
Prediction**



Similarity



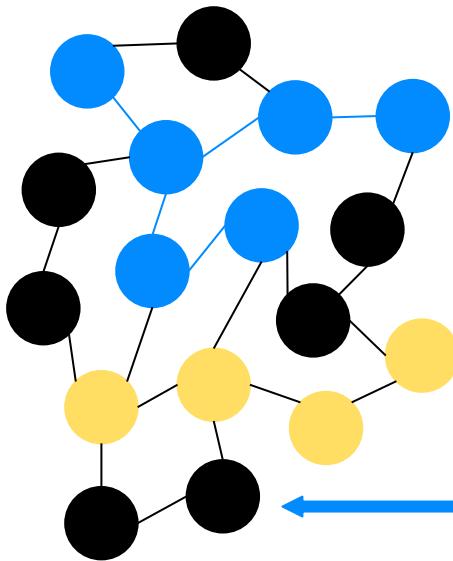
**Graph
Embeddings**



...and more

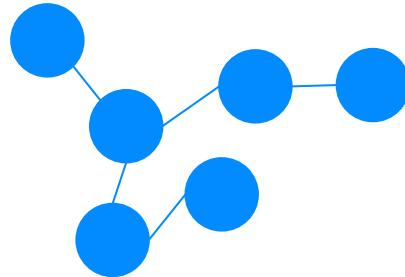
GDS Process in Neo4j

Build Knowledge Graph



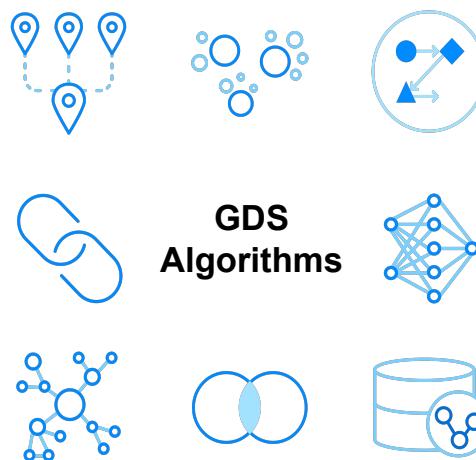
Ingest and model data into the Knowledge Graph. Then, it will be stored into the **disk**.

Graph Projection



Specific portion of the **graph** will be **projected in-memory** depending on what the algorithm requires e.g. monopartite, bipartite or multipartite graph.

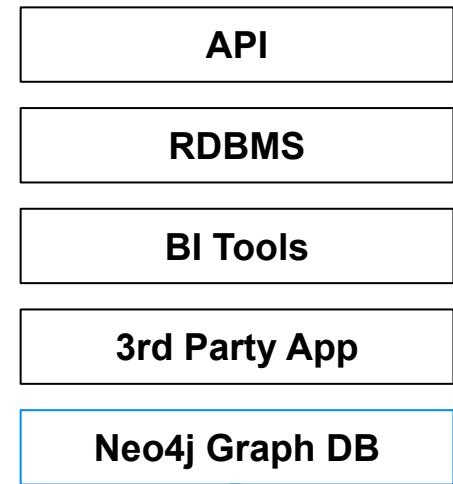
Execute Algorithm



GDS Algorithms

Graph algorithm and machine learning will be executed within Neo4j.

Use the Graph Insights



Outputs of the GDS algorithms and ML is **feed back** into **Neo4j** or any of the **downstream systems**.

Graph Data Science (GDS)

Using Python



Neo4j GDS Python Client Library

Graph Data Science (GDS) Python Client package called **graphdatascience** to interact with Neo4j Graph Data Science Library.

#	Syntax	What does it do?
1	<code>pip install graphdatascience</code>	Install Graph Data Science (GDS) Python client
2	<code>gds = GraphDataScience(neo4j_url, auth=(user, password), aura_ds= False)</code>	Instantiate GDS client session
3	<code>gds.set_database("neo4j")</code>	Set specific Neo4j database
4	<code>gds.run_cypher(<Cypher query>, params={"data": data})</code>	Run Cypher query with input data
5	<code>gds.graph.project()</code>	Create graph projection
6	<code>gds.graph.drop()</code>	Drop existing graph projection
7	<code>gds.<algo>.<mode>()</code>	Execute a graph algorithm
8	<code>gds.close()</code>	Closing a GDS client session

Documentation: [Neo4j Graph Data Science Python Client Manual](#)

Workshop Scenario



Workshop Environment(s)



colab +



neo4j
sandbox



jupyter



neo4j
desktop

The dataset

☰ README.md

 Java CI passing

A Fork of PaySim - Simulating Mobile Money Networks

This is a fork of [PaySim 2.0](#) designed for use as a library while maintaining some ability to run standalone (if desired).

It's first usage is in conjunction with [Neo4j](#) to create a network graph, facilitating application of graph algorithms for detecting fraud characteristics.

Why Fork?

Numerous high-level enhancements from the original include:

- Implementation of First and Third Party fraudsters
 - 1st Party Fraudsters steal or use fake identities to open new accounts and drain their value
 - 3rd Party Fraudsters prey upon normal clients via "compromised" merchants (like via card skimming) and drain accounts
- Incorporation of "identities" tied to clients to help facilitate First Party Fraud simulation including SSN, Email, and Phone Numbers

<https://github.com/voutilad/PaySim>

The dataset



Transaction	Description
CashIn	A Client moves money into the network via a Merchant
CashOut	A Client moves money out of the network via a Merchant
Debit	A Client moves money into a Bank
Transfer	A Client sends money to another Client
Payment	A Client exchanges money for something from a Merchant

We are starting with lots of this....

```

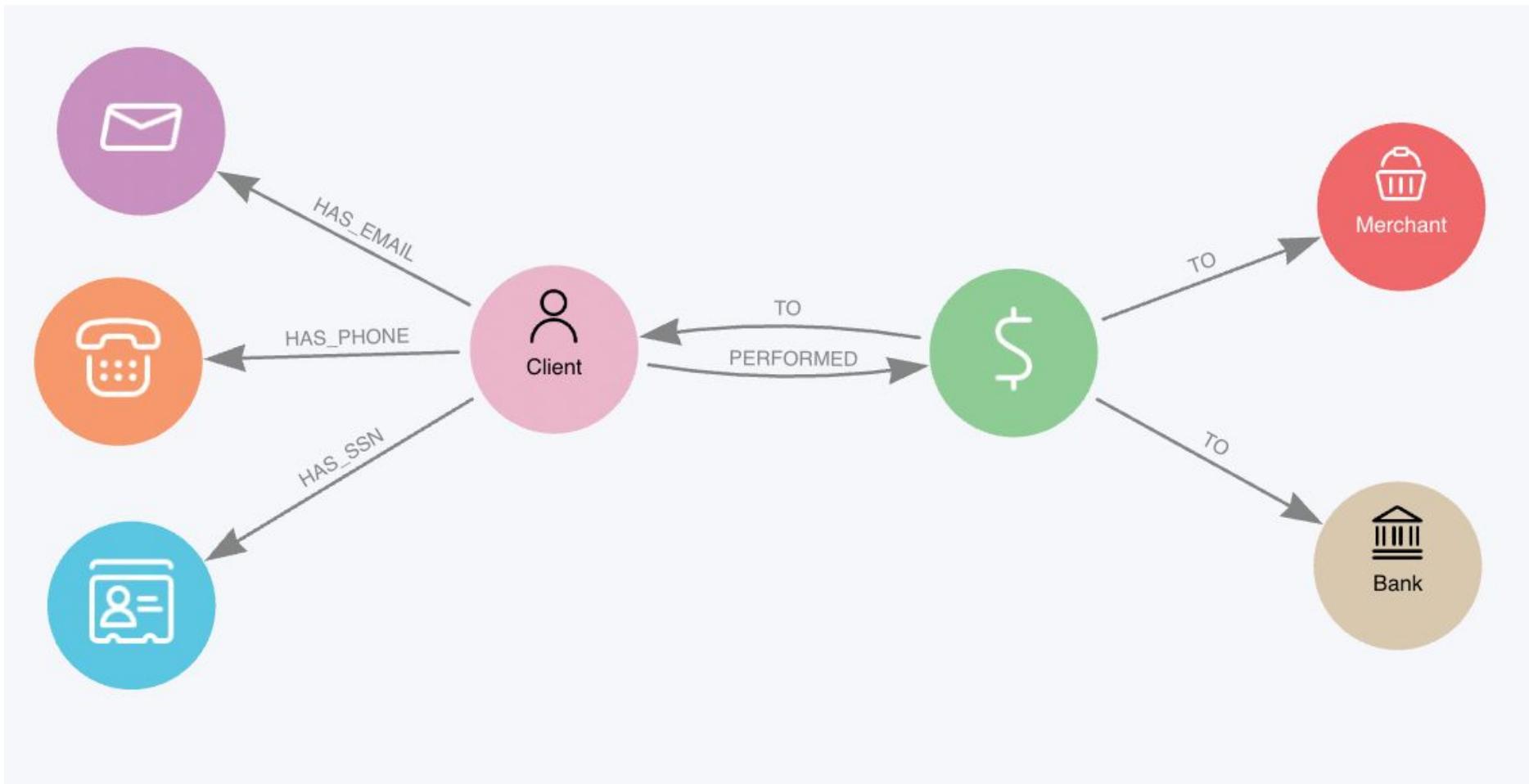
"CASH_IN", "237116.77070964957", "20", "23-0004491", "4498460370531508", "false", "false", "Morsem Company", "Emily Kennedy", "MERCHANT", "CLIENT"
"CASH_IN", "179365.2082069302", "21", "15-0003005", "4241749339258264", "false", "false", "Morsem Industries", "Alex Vincent", "MERCHANT", "CLIENT"
"CASH_IN", "76752.68374051453", "22", "21-0009346", "4080865025714388", "false", "false", "Alist", "Brayden Oliver", "MERCHANT", "CLIENT"
"CASH_IN", "149276.16857622805", "23", "22-0008851", "4502784777097761", "false", "false", "Datastore Industries", "Ava Abbott", "MERCHANT", "CLIENT"
"CASH_IN", "235372.5933713698", "24", "39-002150", "4011728151648289", "false", "false", "Gualas", "Autumn Bowers", "MERCHANT", "CLIENT"
"CASH_IN", "106590.67072335906", "25", "82-0007635", "4794986630113852", "false", "false", "Robutenia", "Harper Duke", "MERCHANT", "CLIENT"
"CASH_IN", "188255.7546070336", "26", "23-0001463", "4893589937977753", "false", "false", "FlyHigh", "Colton Reilly", "MERCHANT", "CLIENT"
"CASH_IN", "23001.785424514455", "27", "56-0006095", "4132980803492624", "false", "false", "Woods", "Stella McLaughlin", "MERCHANT", "CLIENT"
"CASH_IN", "47481.905640228404", "28", "31-0000111", "4348182805684841", "false", "false", "Beans", "Chase Gibson", "MERCHANT", "CLIENT"
"CASH_IN", "65192.65955373068", "29", "21-0009346", "4080223082232961", "false", "false", "Alist", "Juan Estes", "MERCHANT", "CLIENT"
"CASH_IN", "126238.14804415256", "30", "92-0008906", "4872545349272873", "false", "false", "Quicker Industries", "Amelia Ellis", "MERCHANT", "CLIENT"
"CASH_IN", "327311.6222781761", "31", "92-0008906", "4872545349272873", "false", "false", "Quicker Industries", "Amelia Ellis", "MERCHANT", "CLIENT"
"CASH_IN", "136107.09527000843", "32", "92-0008906", "4872545349272873", "false", "false", "Quicker Industries", "Amelia Ellis", "MERCHANT", "CLIENT"
"CASH_IN", "191827.07112804684", "33", "92-0008906", "4872545349272873", "false", "false", "Quicker Industries", "Amelia Ellis", "MERCHANT", "CLIENT"
"CASH_IN", "117966.8078601925", "34", "92-0008906", "4872545349272873", "false", "false", "Quicker Industries", "Amelia Ellis", "MERCHANT", "CLIENT"
"CASH_IN", "170995.93742837547", "35", "59-0006985", "4892844970450163", "false", "false", "Interdem Company", "Amelia Winters", "MERCHANT", "CLIENT"
"CASH_IN", "80116.96737040897", "36", "36-0007378", "4127130248030491", "false", "false", "Datastore", "Justin Peters", "MERCHANT", "CLIENT"
"CASH_IN", "95795.11338322057", "37", "24-0002737", "4600611278590971", "false", "false", "Woods Associates", "Jason Taylor", "MERCHANT", "CLIENT"
"CASH_IN", "124094.62877092097", "38", "73-0000254", "418916438564525", "false", "false", "Nonos", "Henry Whitney", "MERCHANT", "CLIENT"
"CASH_IN", "126236.86980934851", "39", "73-0000254", "4189164385634525", "false", "false", "Nonos", "Henry Whitney", "MERCHANT", "CLIENT"
"CASH_IN", "autumnlittle592@yahoo.com", "4139396596344943", "true", "2", "126236.86980934851", "39", "73-0000254", "4189164385634525", "false", "false", "Nonos", "Henry Whitney", "MERCHANT", "CLIENT"
"CASH_IN", "398703.6251183568", "40", "46-0003505", "4583007980881892", "false", "false", "MemorTech Consulting", "Christopher Garrison", "MERCHANT", "CLIENT"
"CASH_IN", "129569.38179422423", "41", "24-0002938", "4373961024118824", "false", "false", "Quicker", "Julia Hoffman", "MERCHANT", "CLIENT"
"CASH_IN", "62870.15542699712", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "averymathis567@gmail.com", "4768167410334172", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "christian.sims774@mail.com", "4122563683829958", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "burt288@mail.com", "4122103098728012", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "kaylee.battle180@yahoo.com", "4138529467491382", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "caldwell1698@gmail.com", "4806443672380185", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "avery.mccormick727@mail.com", "46831767348883328", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "juan.williams606@gmail.com", "41113147778042310", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "tristansosa502@mail.com", "4221203098728012", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "jose.robbins647@yahoo.com", "4713875011658379", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "mueller722@yahoo.com", "4530290273653129", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "addisc@gmail.com", "46849.28639214253", "47", "36-0007858", "4559665405902773", "false", "false", "Adapt", "Carlos Kemp", "MERCHANT", "CLIENT"
"CASH_IN", "wyattcohen831@gmail.com", "4560248735199209", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Datastore", "Reagan Becker", "MERCHANT", "CLIENT"
"CASH_IN", "byers204@mail.com", "4882234888667276", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Erntogra", "Alyssa Baldwin", "MERCHANT", "CLIENT"
"CASH_IN", "andrewadkins277@yahoo.com", "497806199279401", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Erntogra", "Alyssa Baldwin", "MERCHANT", "CLIENT"
"CASH_IN", "grace.dickerson481@gmail.com", "4703706549747693", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "aguilar570@gmail.com", "4145736547359086", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "khloe.bishop063@gmail.com", "4528302092557334", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "lillian.elliott678@gmail.com", "4740673726551766", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "jose.wright4318@gmail.com", "455199301424019", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "rollins847@yahoo.com", "4257432936434825", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "jonathan.palmer154@yahoo.com", "4723679131725953", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "spence804@mail.com", "4175048645840406", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "sebastian.stein0698@yahoo.com", "4034620484775900", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "jacob.weiss549@gmail.com", "4365887814447273", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "blakeconley123@mail.com", "4969980201223208", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "miller758@yahoo.com", "4584980964633993", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "brayden.richardson150@mail.com", "4709391052055802", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "haney332@mail.com", "4800973552748776", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "spence804@mail.com", "4175048645840406", "true", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "ryder.bryant525@gmail.com", "4012972056822354", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "curtis101@mail.com", "4050554188919036", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "chloe.michael209@yahoo.com", "4446503711408614", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "wolf103@mail.com", "4126628014879075", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "carolinemerriett415@mail.com", "4571374539472217", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "justin.snyder645@yahoo.com", "4556641339267313", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "peyton.mccoy244@gmail.com", "4751316150101353", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "merritt570@mail.com", "4359066643426828", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "piper.brady889@yahoo.com", "4809475630218328", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "kaylaaguirre17@yahoo.com", "4085939032240804", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "lydiafranks3518@gmail.com", "446508711694111", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "thomas.walls656@gmail.com", "4161981667647182", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "webster203@mail.com", "4614848240759474", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"
"CASH_IN", "lel91@yahoo.com", "4401527017466037", "false", "2", "129569.38179422423", "42", "52-0001990", "4901646199659307", "false", "false", "Felics Ltd", "Henry Long", "MERCHANT", "CLIENT"

```

46 "TRANSFER", "72053.4599965948", "6621", "4389696512811454", "4758208726940773", "false", "false", "Connor Houston", "Victoria Pearson", "CLIENT", "CLIENT"

47 https://neo4j.com/graphdb/import/neo4j-importer.html#neo4j-importer-importing-a-graph

...and making sense of it like this



We are just scratching the surface....

WHERE DO I GO NEXT?

Curated Learning Paths

Once you have learned the fundamentals, you are free to diverge into one of our curated Learning Paths to become an expert in your chosen subject area.

Cypher Data Scientist Developer

[View all courses →](#)

Data Scientist

6 courses

This path includes everything a Data Scientist needs to know, from loading data to applied tutorials on the Graph Data Science library.

[Follow the Data Scientist path →](#)

Beginners

Introduction to Neo4j Graph Data Science

Gain a high-level technical understanding of the Neo4j Graph Data Science (GDS) library

~ 30 minutes

[View Course →](#)

Beginners

Neo4j Graph Data Science Fundamentals

Learn all you need to know about Graph Algorithms and Machine Learning Pipelines

~ 1 hour

[View Course →](#)

Begin

Path with

Learn how between p

~ 1 h

< >

<https://graphacademy.neo4j.com/>

Lets get started !



<https://bit.ly/graph-summit-apac-2023>



Thank You!
Coming up...
Graph Summit Meetup