

Star-Wars® Jedi Workshop

Version 0.8

Mark Quinsland
Senior Field Engineer
February 8, 2023



Neo4j Star-Wars® Jedi Workshop Agenda

- Quick review of Neo4j
- Creating a Jedi data model using Arrows
- Using AuraDB - Neo4j's Cloud DB
- Creating a Star-Wars® graph using Importer
- Exploring the graph visually using Bloom
- Cypher Queries to read, write, update and delete
- Loading Jedi data into Neo4j using Python
- Creating Jedi DataFrames using Jupyter notebooks
- Analyzing characters using Graph Data Science algorithms

The World's Most Popular Property Graph DB

100m+ Neo4j Downloads
250k+ Community Members

7 Retailers
of the Top 10



7 Telcos
of the Top 10



8 Automakers
of the Top 10



5 Pharmaceuticals
of the Top 5

8 Insurance
of the Top 10



3 Hotels
of the Top 5

20 Banks
of the Top 20
North American



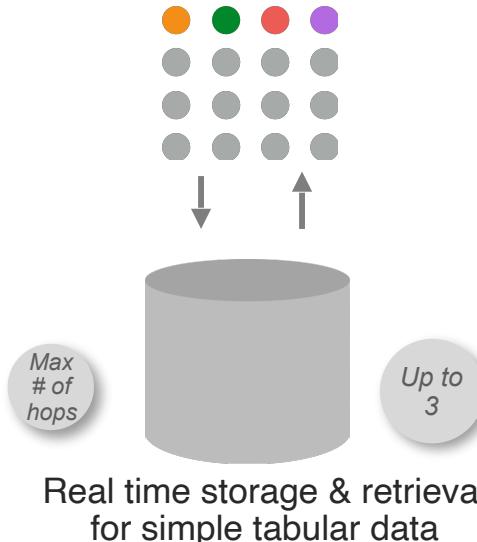
3 Aircraft
Manufacturers
of the Top 5



Data Platform Evolution

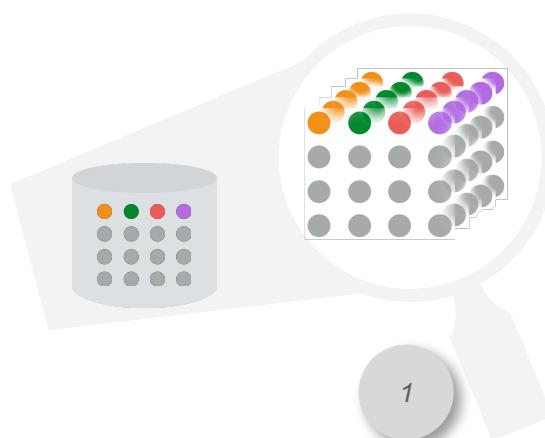
TRADITIONAL DATABASES

Store and retrieve data

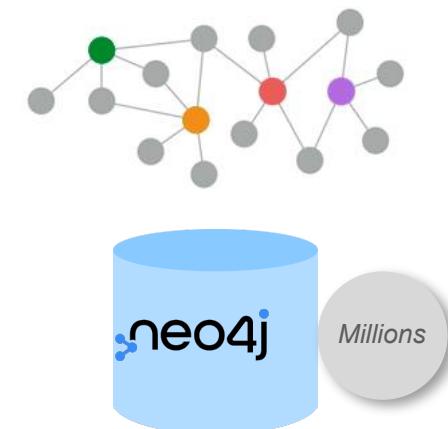


BIG DATA TECHNOLOGY

Aggregate and filter data



Exploit connections in data



Real-time graph queries & algorithms for contextual insights

"Our **Neo4j** solution is literally **thousands of times faster** than the prior MySQL solution, with queries that require **10-100 times less code**"
Volker Pacher, Senior Developer



the Labeled Property Graph Model

Nodes

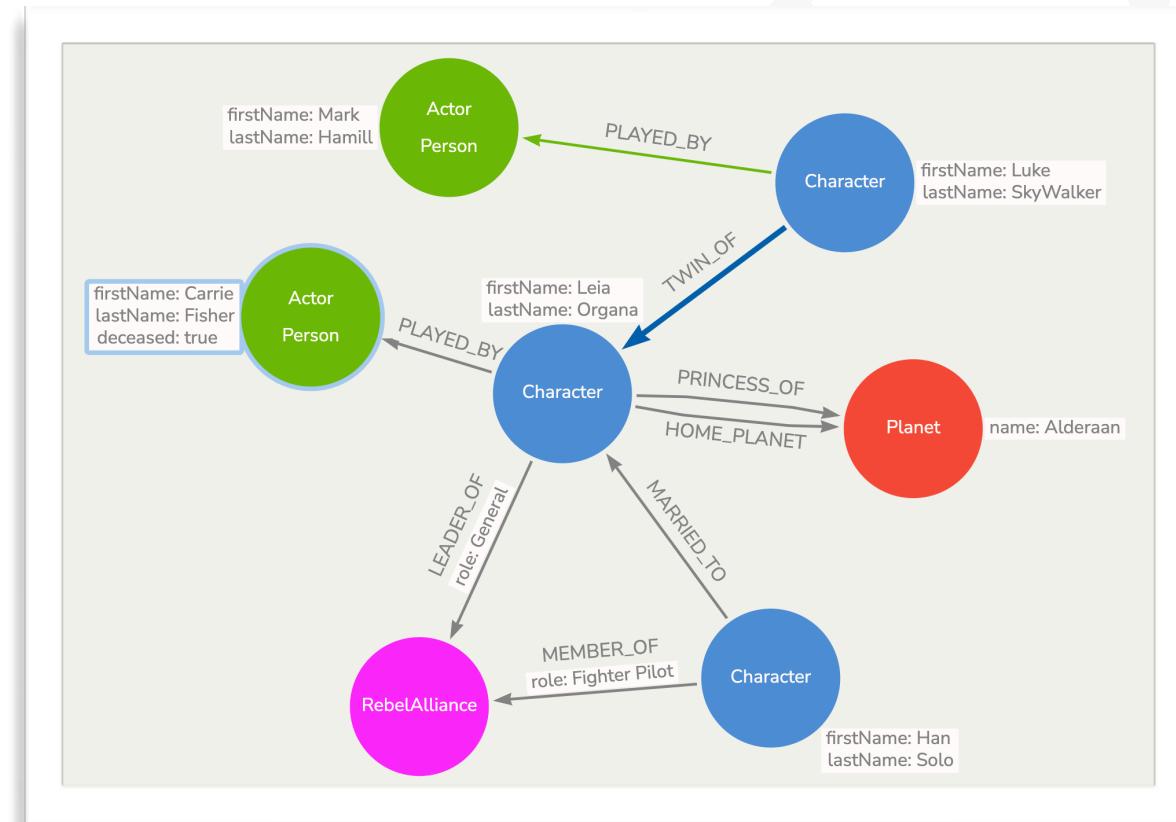
- Define the nouns/entities
- Can have multiple labels

Relationships

- Verbs that relate nodes by type and direction
- Nodes can have multiple relationships

Properties

- Attributes of Nodes and Relationships
- Stored as name/value pairs



- New labels, relationship types and properties can be added w/o schema change

Complex SQL Queries vs Cypher



Find all direct reports and how many people they manage, up to three levels down

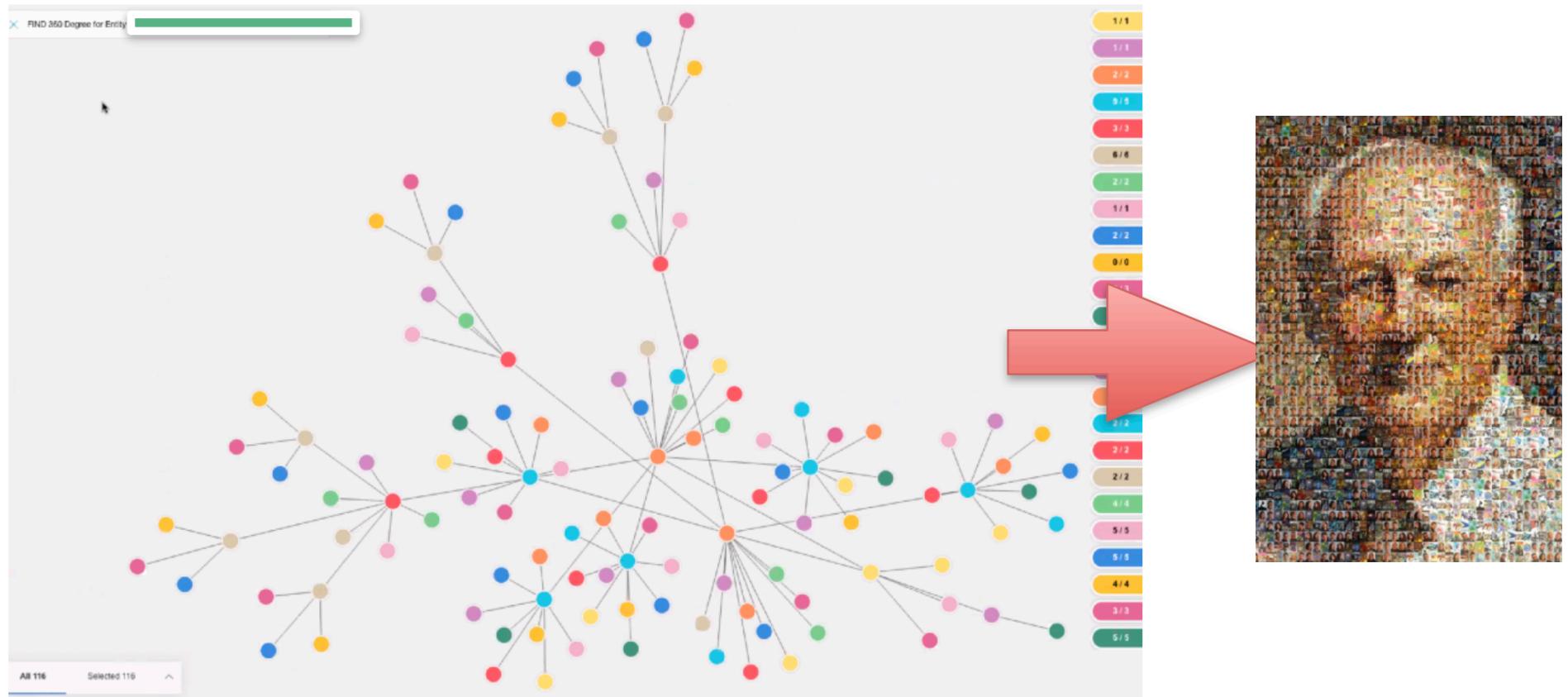
```
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
    SELECT manager.pid AS directReportees, 0 AS count
    FROM person_reportee_manager
    WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
    UNION
    SELECT manager.pid AS directReportees, count(manager.directly_manages) 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
    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 reportee.pid AS directReportees, count(L2Reportees.directly_manages) AS count
    FROM person_reportee_manager
    JOIN person_reportee L1Reportees
    ON manager.directly_manages = L1Reportees.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 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
    UNION
    SELECT L2Reportees.pid AS directReportees, count(L3Reportees.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 L3Reportees.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"))
)
```

Cypher Query

```
MATCH (boss)-[ :MANAGES*0..6 ]->(sub),  
      (sub)-[ :MANAGES*1..3 ]->(report)  
WHERE boss.name = "John Doe"  
RETURN sub.name AS Subordinate,  
      count(report) AS Total
```

- Less time writing queries
 - Less time debugging queries
 - Code that's easier to read

Real-time Joins of Data From Myriad Sources to Provide 360° View



Hybrid Scoring-Based Approach is More Contextual

Graph technology enables you to make recommendations that weight multiple methods



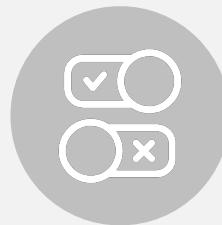
Collaborative Filtering

Based on similar users or products



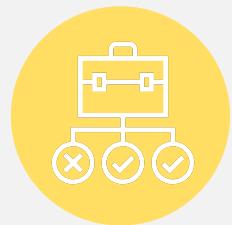
Content Filtering

Based on user history and profile



Rules-Based Filtering

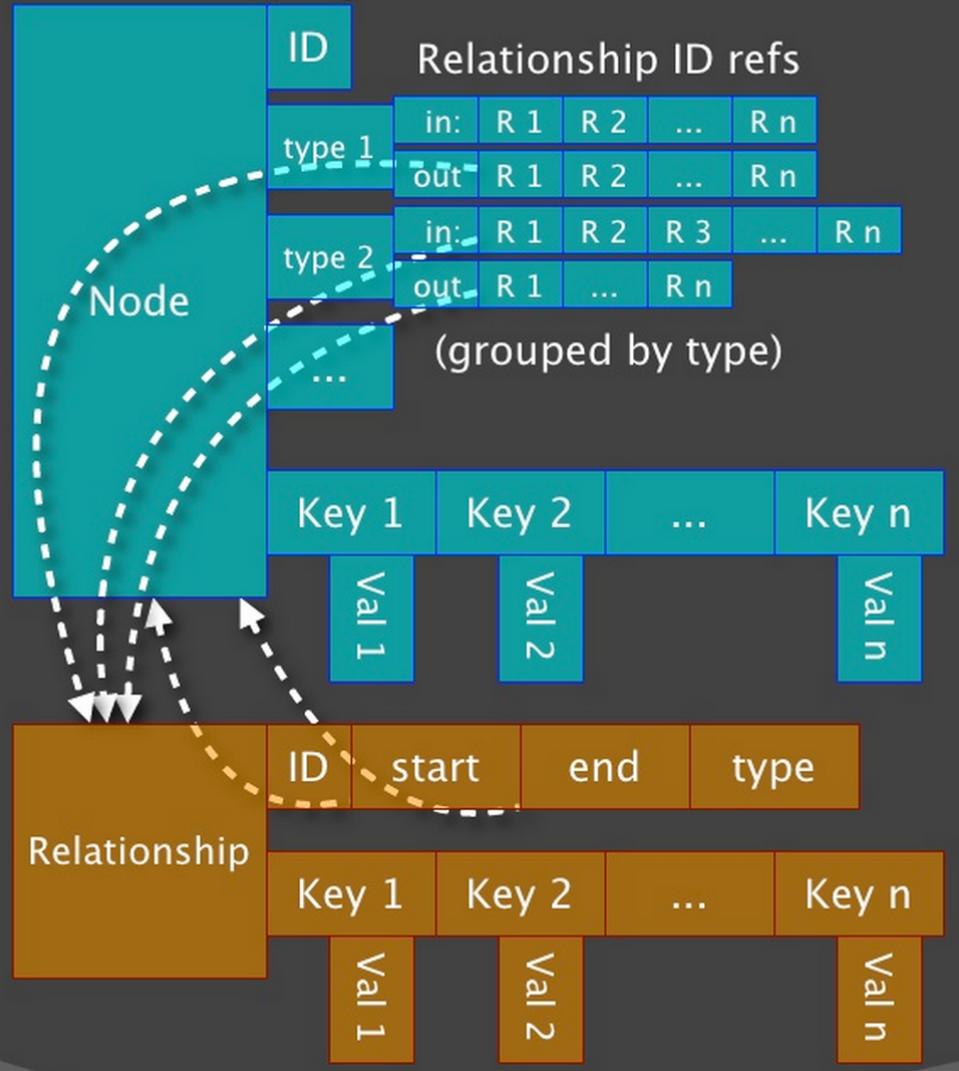
Based on predefined rules and criteria



Business Strategy

Based on promotions, margins, inventory

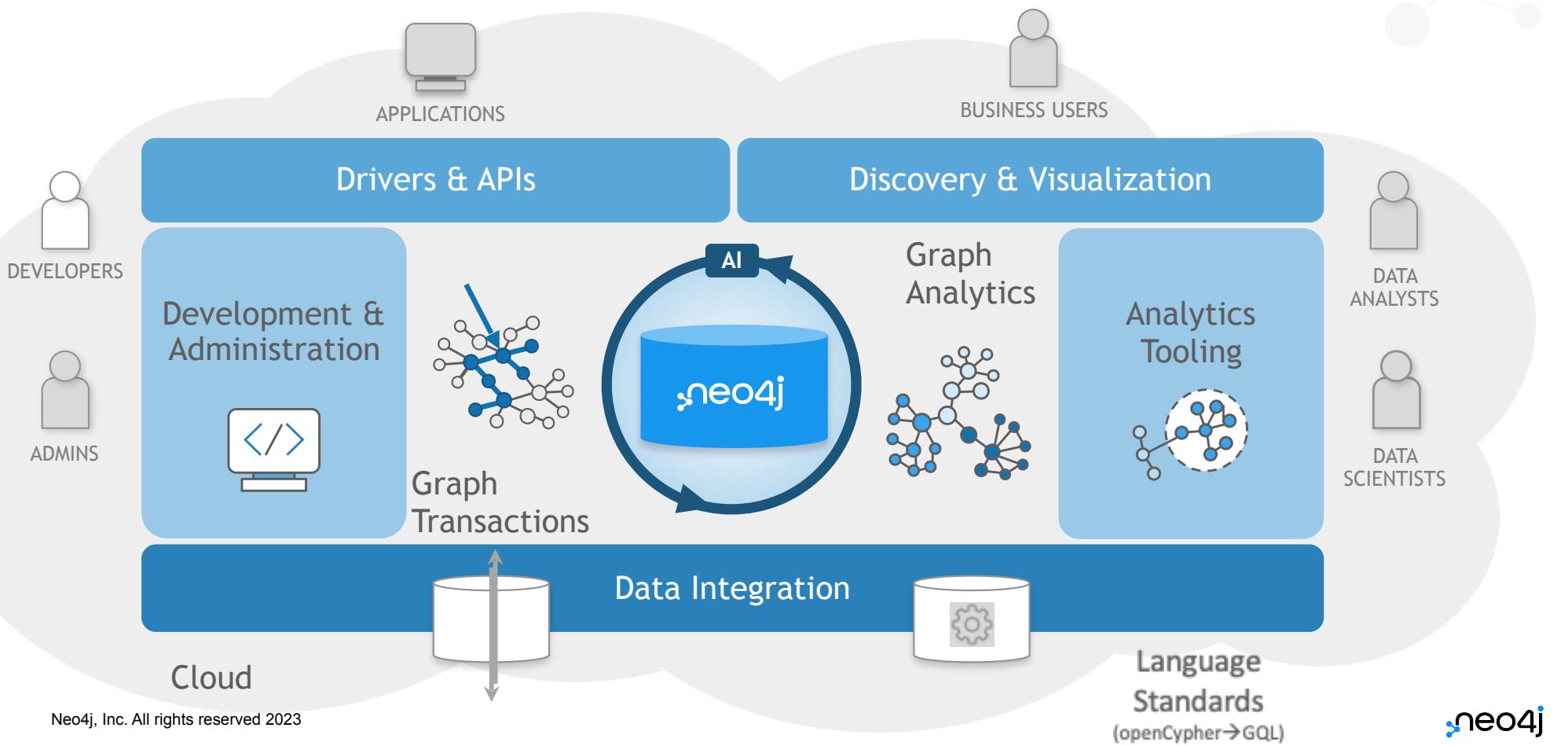
What we put in cache



Neo4j Secret Sauce

- 1** Pointers instead of Lookups
- 2** Fixed Sized Records
- 3** “Joins” on Creation
- 4** Spin Spin Spin through this data structure

Neo4j Graph Platform



Neo4j in the Cloud



The world's most popular graph database, as a cloud service

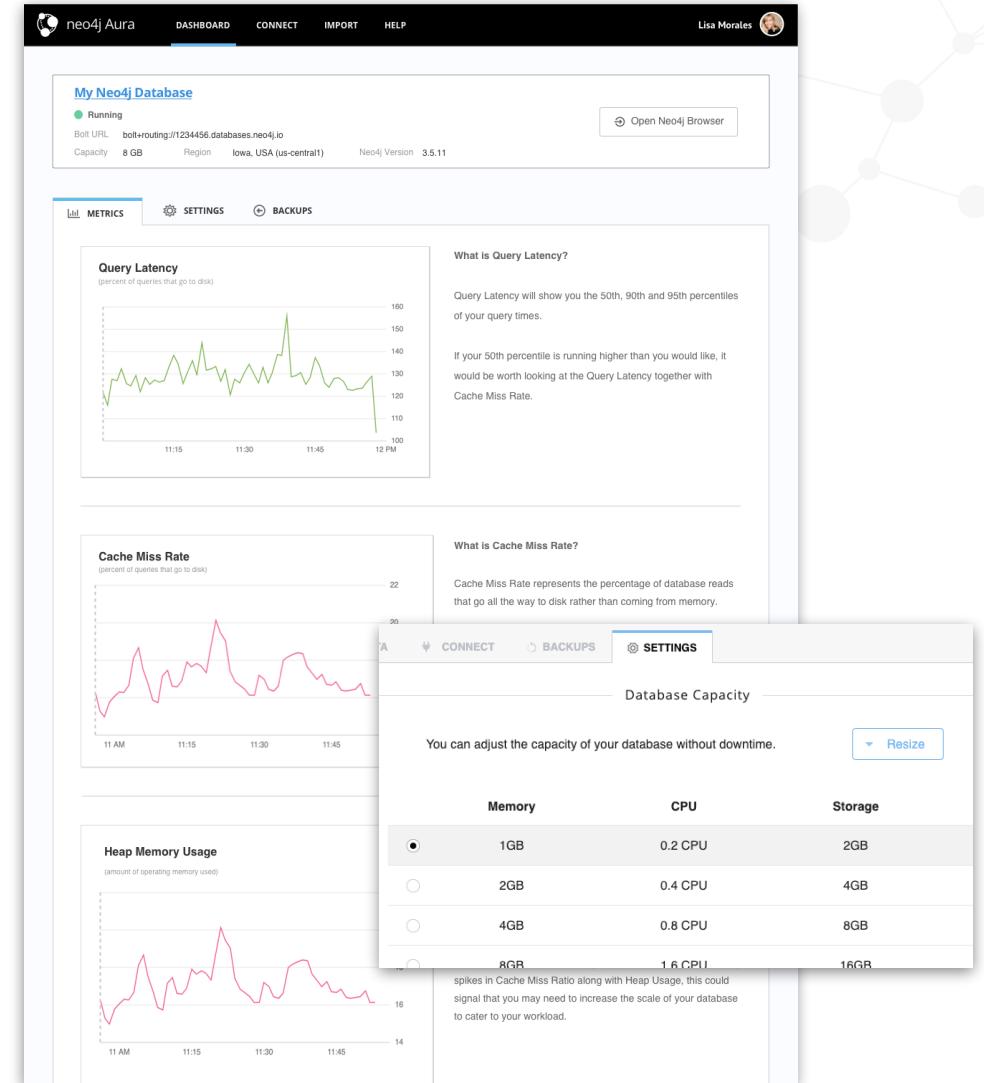
Fully automated with zero administration

Faster innovation with the power of graphs

Scalable on-demand dynamically

Worry-free security and reliability

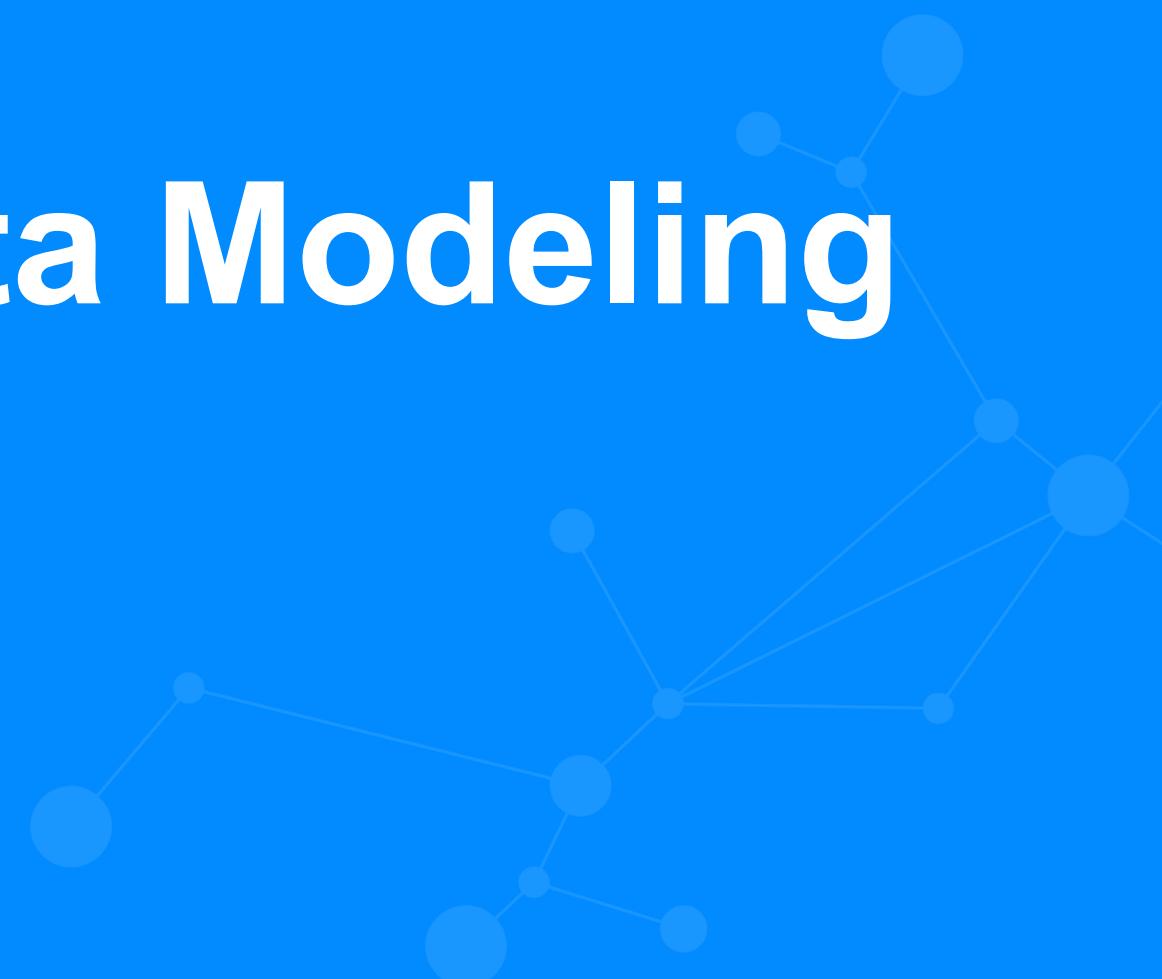
Simple pay-as-you-go pricing



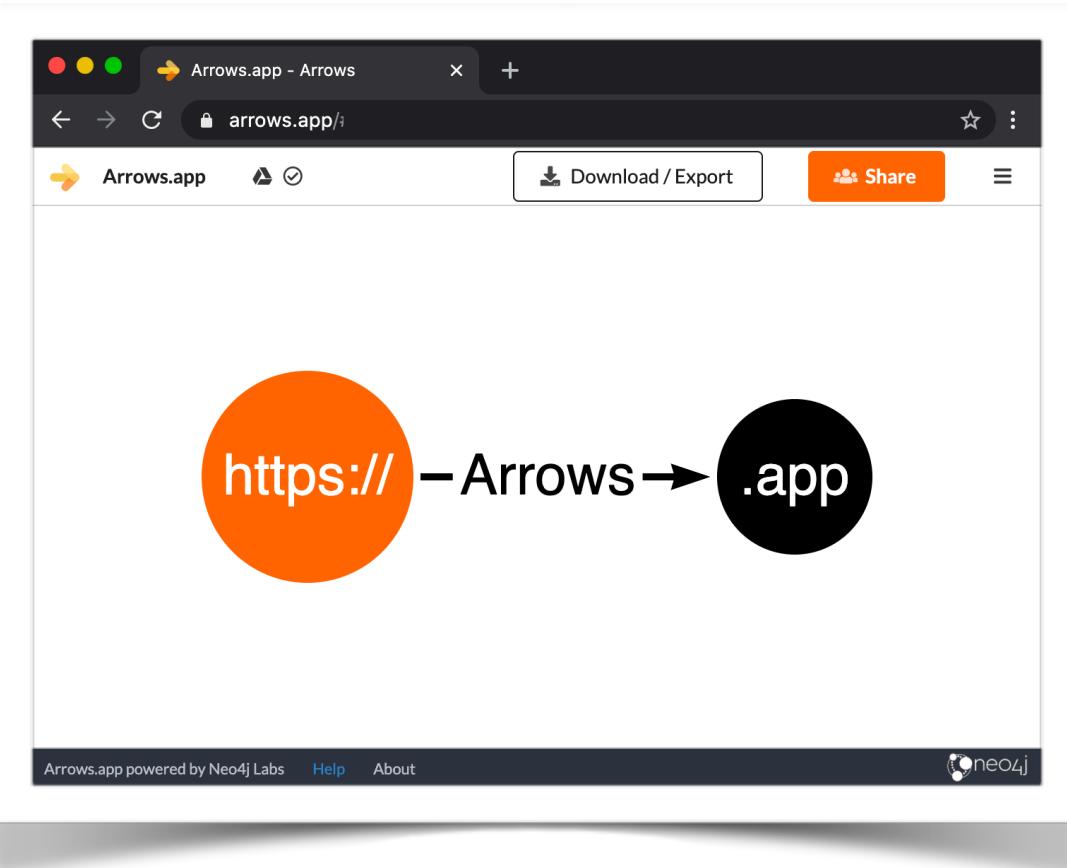
Lab 0 - Connecting to Aura



Neo4j Data Modeling



Visually Create Data Models with Arrows



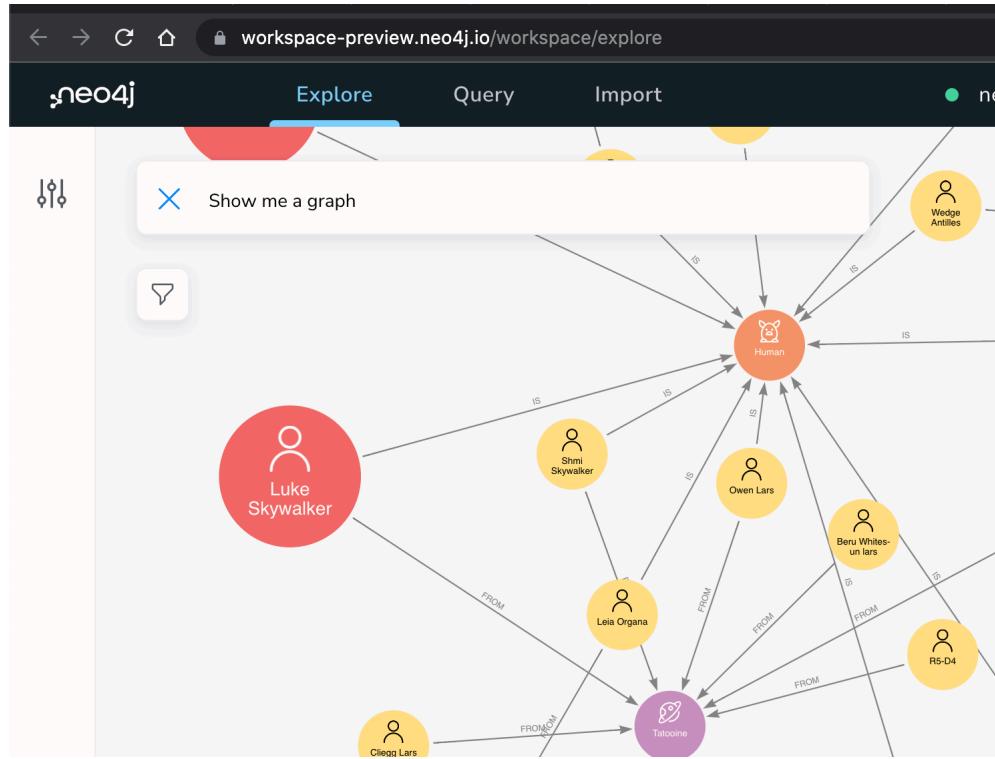
- Create Graphs Visually
- Export to Cypher
- Import / Export models
- Simple to learn
- Youtube demos available

Lab 1 - Data Modeling with Arrows

Neo4j Workspace



Explore, Query, Import with Neo4j Workspace



- Easy-to-use work surface (beta)
- Import CSV files
- Edit/Run Cypher Commands
- Visually explore the graph without using Cypher

Run Cypher Commands -Workspace - Query

neo4j Explore Query Import neo4j+s://d44b75fb.databases.neo4j.io:7687 Send feedback ?

Database Information

Nodes (255) Character Planet Species User

Relationships (657) FROM HOMeworld INTERACTS_WITH IS

Property keys average_height average_lifespan betweenness birthYear classification climate closeness count data designation diameter eye_colors eyecolor gender gravity hair_colors hairColor height homePlanet homeworld

Show all (19 more)

1 MATCH (c3po:Character {name:"C-3PO"}),
(r2:Character {name:"R2-D2"})
2 MATCH p=(c3po)-[i:INTERACTS_WITH]-(others)
3 WHERE NOT (others)-[:INTERACTS_WITH]-(r2)
4 return p

Graph Table RAW

Results overview

Nodes (15) Character (15)

Relationship (14) INTERACTS_WITH (14)

Started streaming 14 records after 3ms and completed after 13ms.

Quick Demo



Export Arrows Model - Import using Workstation

The screenshot shows two windows side-by-side. On the left is the Neo4j Export interface, and on the right is the Neo4j Workstation interface.

Export Window:

- Header: Export
- Format: Cypher (selected)
- Cypher Clause: MERGE (radio button selected)
- Message: "MERGE query behaviour depends on what data is already present in the database. You may need to edit the query to achieve exactly the behaviour you are looking for. Please see MERGE documentation for guidance."
- Buttons: Copy to clipboard, Run in Neo4j Browser
- Code area:

```
MERGE (n1:Character {name: "Leia Organa", birthYear: "aby25"})-[:IMPRISONED]-(n0:Character {name: "Jabba the Hutt", wt: 1000})-[:WORKED_FOR {year: "aby50"}]-(n0:Character {name: "Han Solo", birthYear: "aby20", pilot: "true", occupation: "smuggler"})-[:MARRIED_TO {year: "aby55"}]-(n1)-[:TWIN_OF]-(n0)-[:OWNS]-(n4:Starship {name: "Millennium Falcon", length: 77})-[:PILOT_OF]-(n0)-[:FRIEND_OF]-(n2:Character {name: "Chewbacca", birthYear: "bby5", pilot: "true"})-[:CO_PILOT_OF]-(n4)
```

Workstation Window:

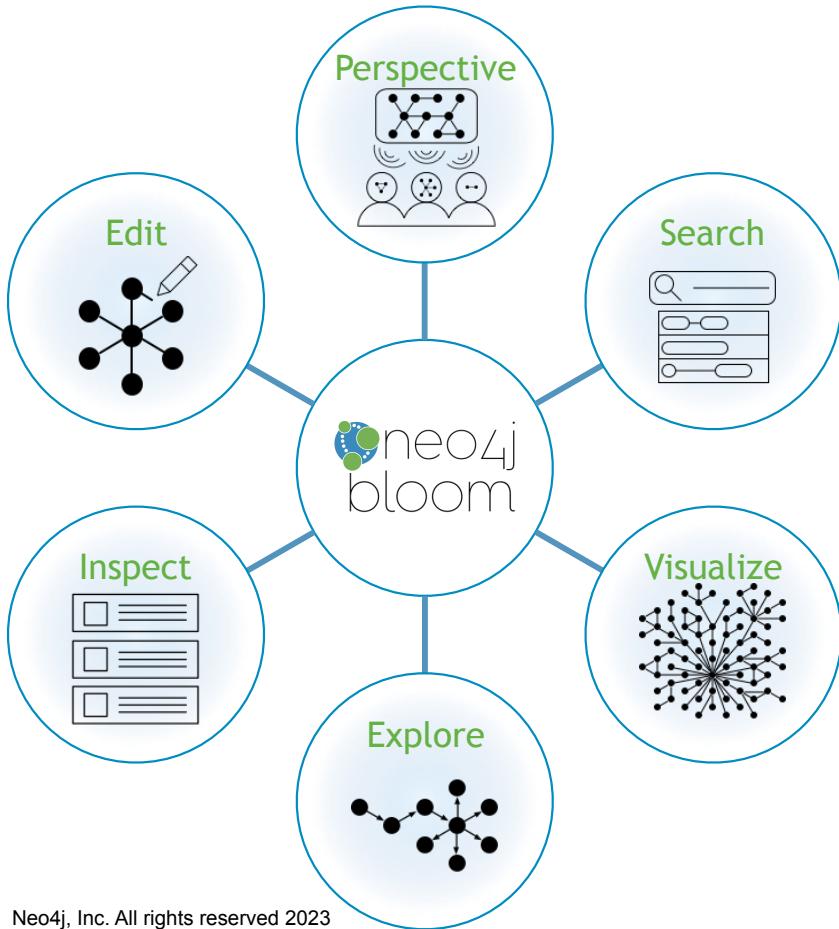
- Header: neo4j
- Tab: Query (selected)
- Import: neo4j+s://d44b75fb.databases.neo4j.com:7687
- Send feedback
- Code area:

```
1 MERGE (n1:Character {name: "Leia Organa", birthYear: "aby25"})-[:IMPRISONED]->(:Character {name: "Jabba the Hutt", wt: 1000})-[:WORKED_FOR {year: "aby50"}]->(:Character {name: "Han Solo", birthYear: "aby20", pilot: "true", occupation: "smuggler"})  
2 -[:MARRIED_TO {year: "aby55"}]->(n1)-[:TWIN_OF]->(:Character {name: "Luke Skywalker", birthYear: "aby25", jedi: "true"})  
3 MERGE (n0)-[:OWNS]->(n4:Starship {name: "Millennium Falcon", length: 77})  
4 -[:PILOT_OF]->(:Character {name: "Chewbacca", birthYear: "bby5", pilot: "true"})-[:CO_PILOT_OF]->(n4)
```

Neo4j Bloom



Explore & Collaborate with Neo4j Bloom



Explore Graphs Visually

Prototype Concepts Faster

Collaborate Across Teams

Neo4j Bloom's Intuitive User Interface

Search with type-ahead suggestions

Flexible Color, Size and Icon schemes

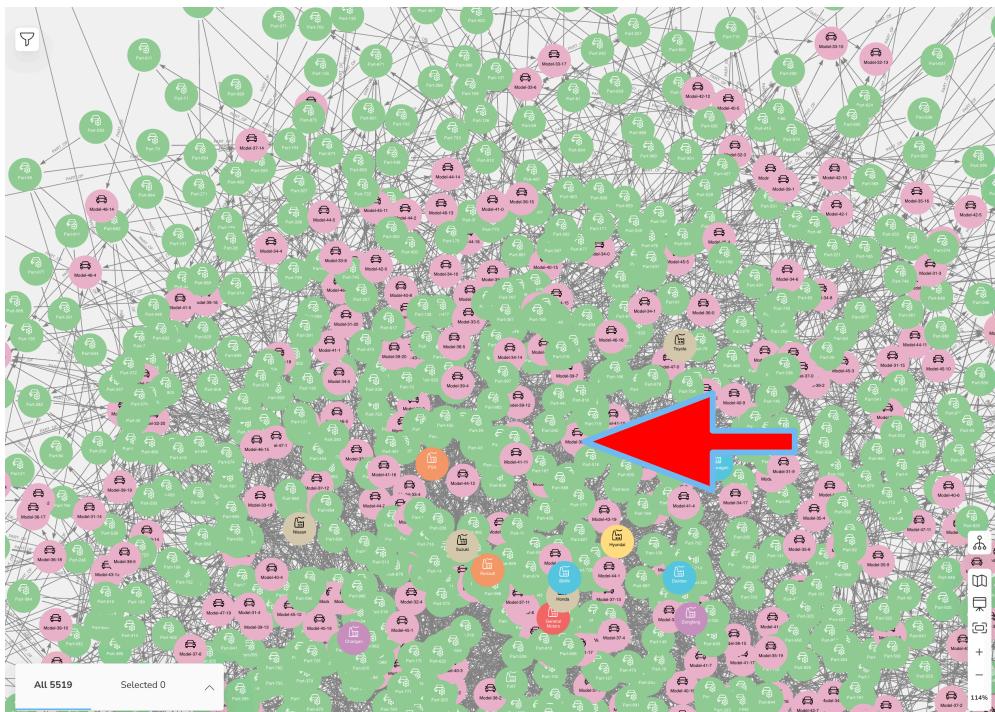
Visualize, Explore and Discover

Pan, Zoom and Select

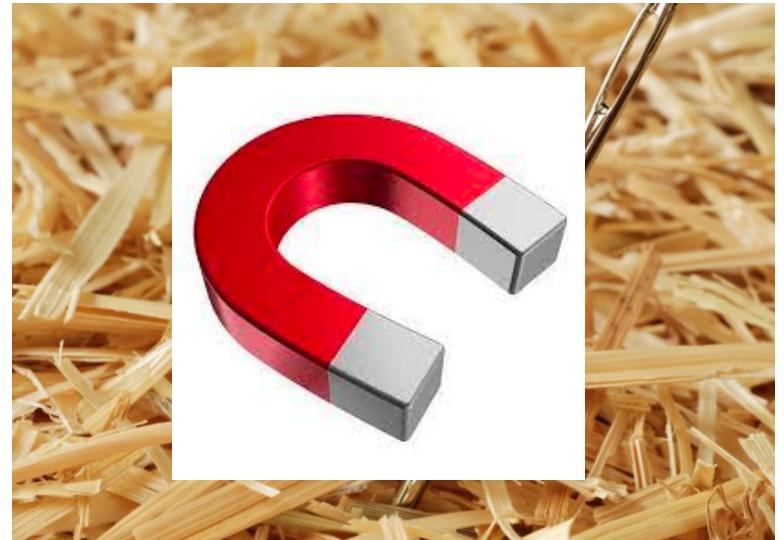
Property Browser and editor



Finding Key Objects



Similar to Finding a Needle in a Haystack



Easier to do with the right tools!

Graph Centrality Algorithms Assist in Analysis

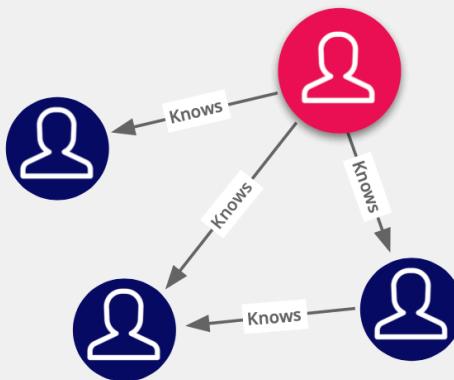


- Closeness algorithms identify:
 - Highly-connected nodes
 - Built-in redundancy
- Betweenness algorithms identify:
 - Strategically-connected nodes
 - Single points of failure
 - Potentially vulnerable nodes

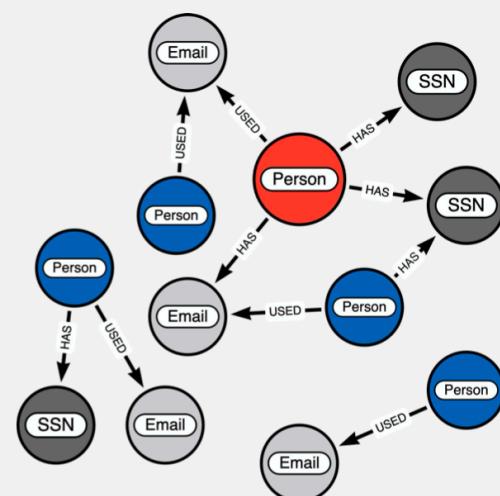
Neo4j Graph Data Science



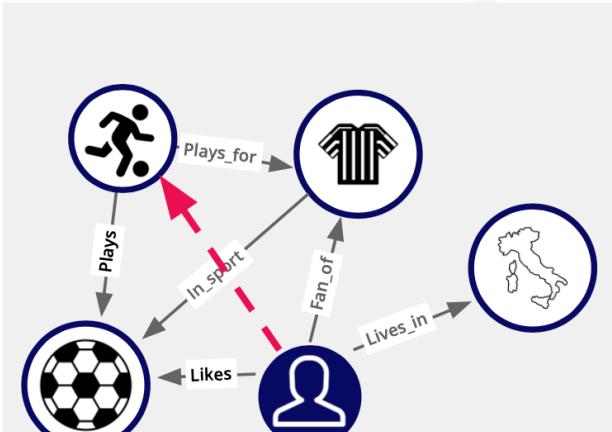
Use Graph Data Science to Answer:



What's important?



What's unusual?



What's next?

Let's Do Something Amazing Together

 neo4j

