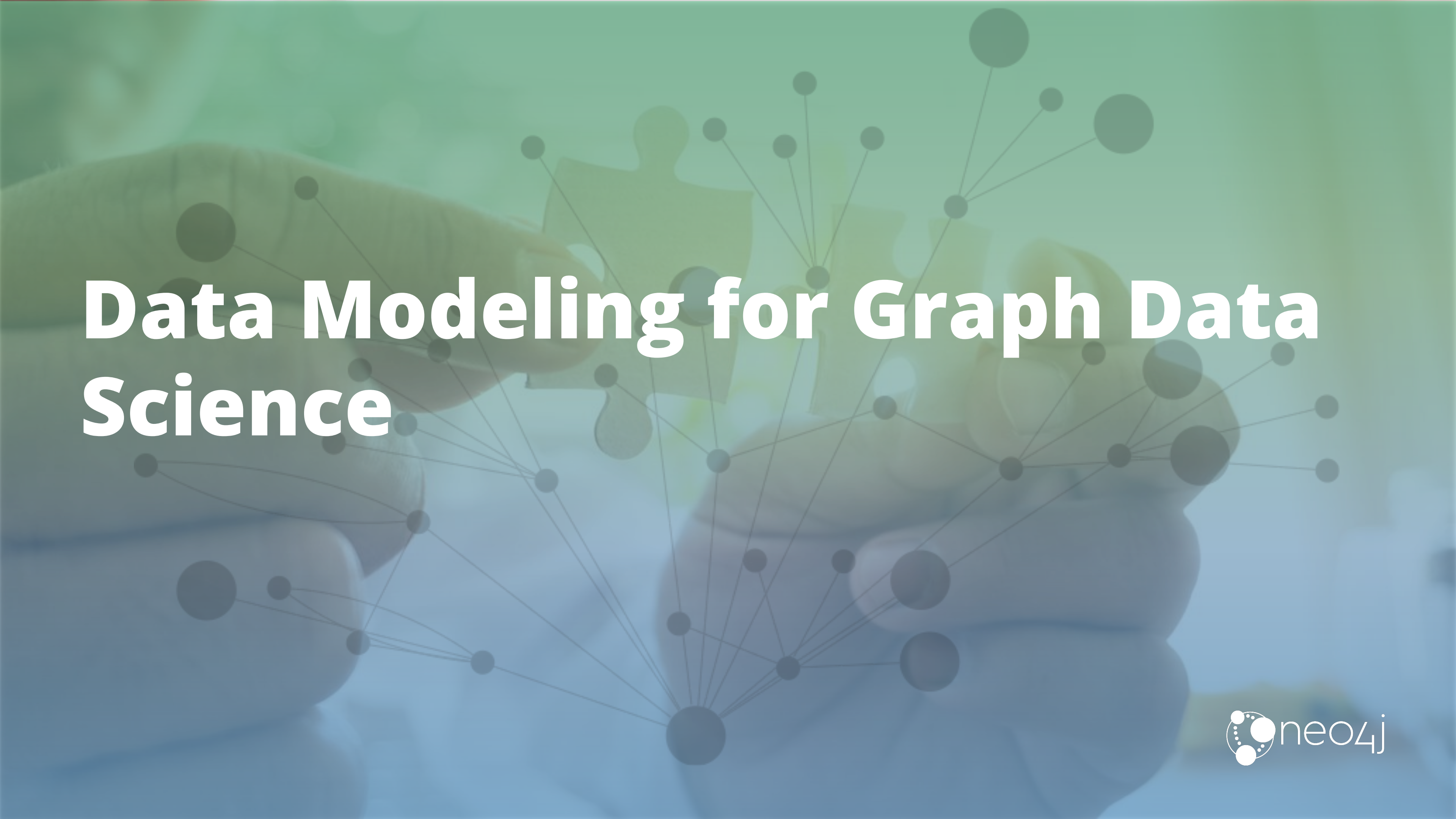
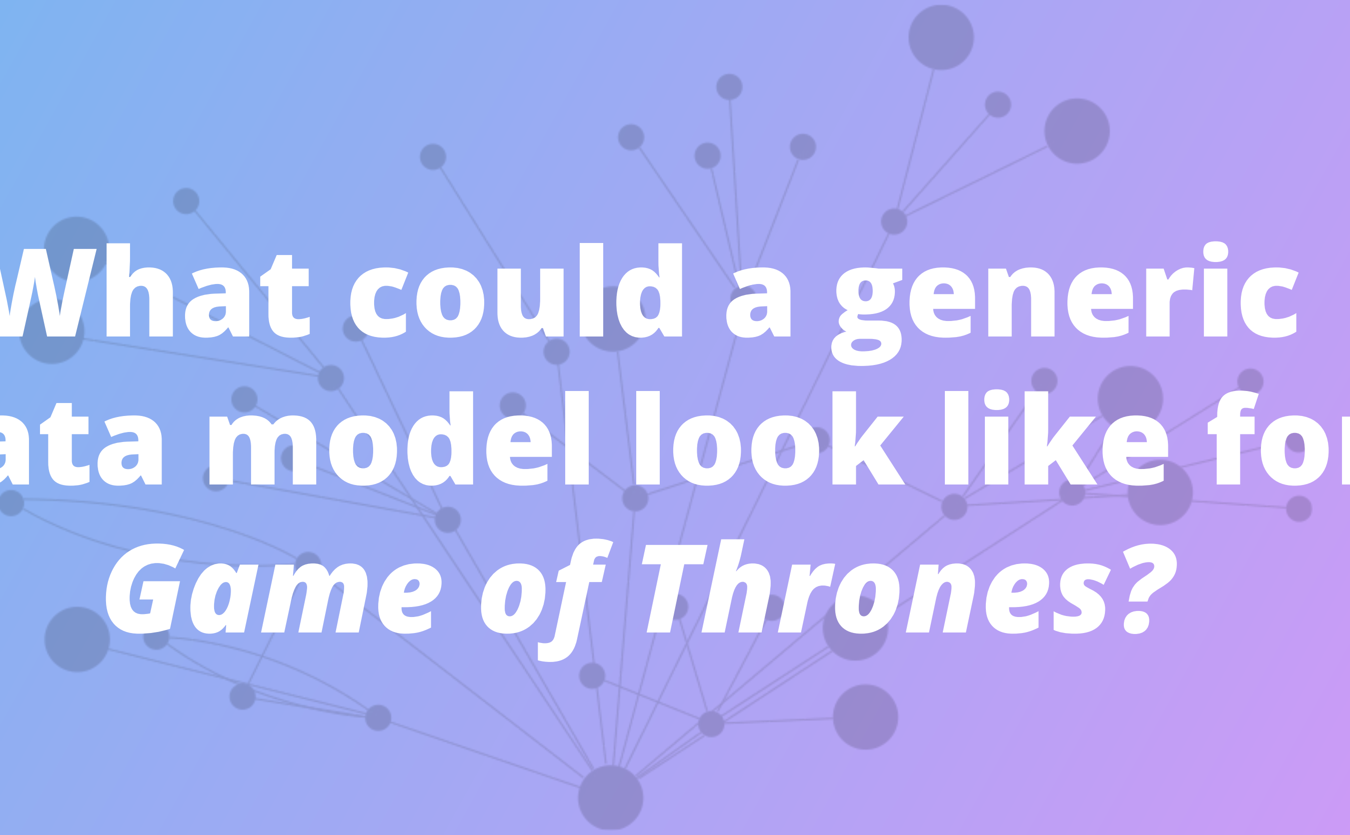


Data Modeling for Graph Data Science

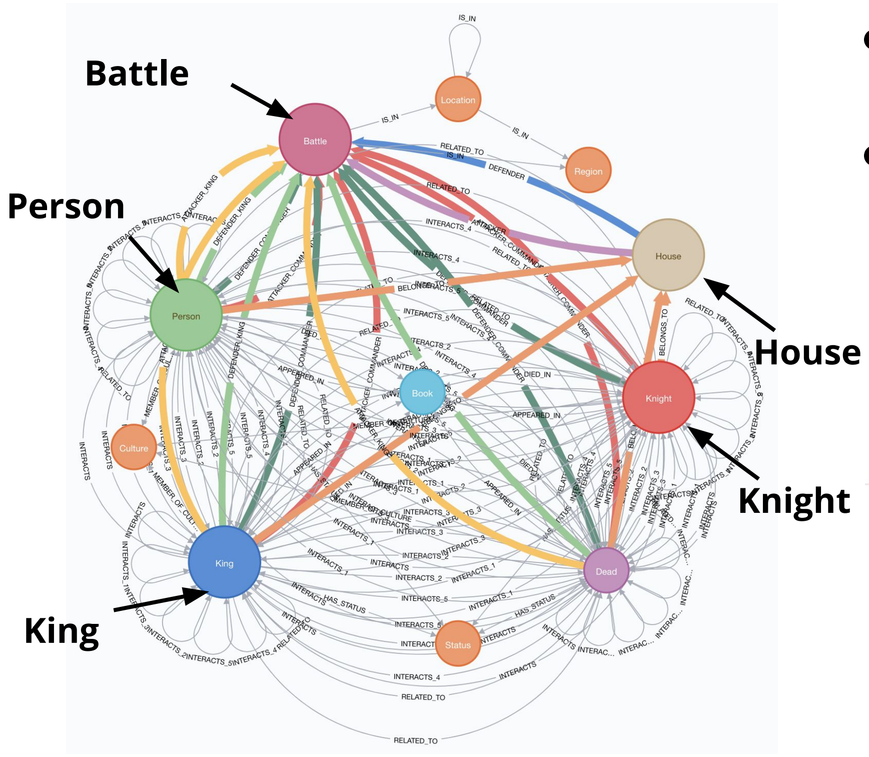
From Previous Session:
Developing the initial graph data model

1. Define high-level domain requirements
2. Create sample data for modeling purposes
3. Define the questions for the domain
4. Identify entities
5. Identify connections between entities
6. Test the model against the questions
7. Test scalability

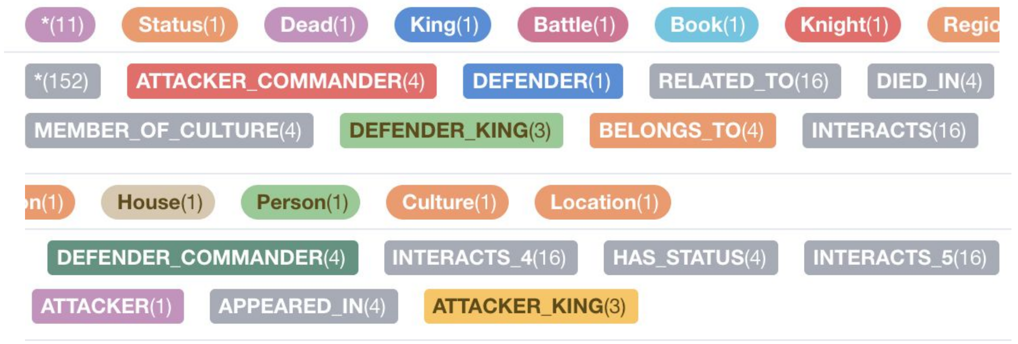


What could a generic data model look like for *Game of Thrones?*

Game Of Thrones Data Model



- Based on Network of Thrones, A song of Math and Westeros
- Information from 5 books. Represents characters, houses they belong to, interactions between them and battles etc..


*(11) Status(1) Dead(1) King(1) Battle(1) Book(1) Knight(1) Regio

*(152) ATTACKER_COMMANDER(4) DEFENDER(1) RELATED_TO(16) DIED_IN(4)

MEMBER_OF_CULTURE(4) DEFENDER_KING(3) BELONGS_TO(4) INTERACTS(16)

n(1) House(1) Person(1) Culture(1) Location(1)

DEFENDER_COMMANDER(4) INTERACTS_4(16) HAS_STATUS(4) INTERACTS_5(16)

ATTACKER(1) APPEARED_IN(4) ATTACKER_KING(3)

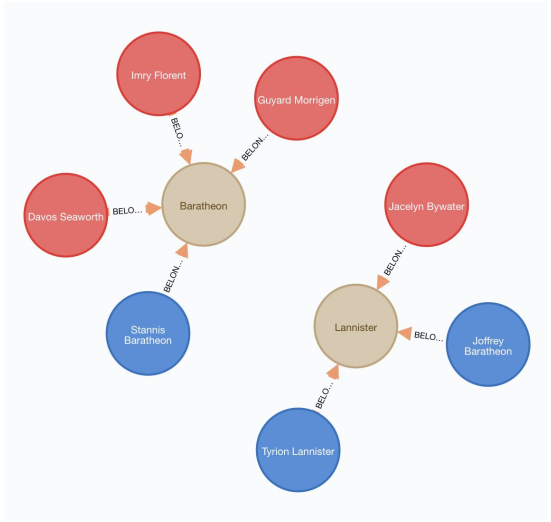
Let's look at some queries..

Find the Houses attacking or defending in "Battle of the Blackwater Bay" and return names of Kings and Knights in those houses

```
MATCH (k) -
[:ATTACKER_KING |:DEFENDER_KING |:ATTACKER_COMMANDER
|:DEFENDER_COMMANDER]
-> (b:Battle {name: 'Battle of the Blackwater'})
WHERE 'King' in labels(k) or 'Knight' in labels(k)

MATCH (h:House) - [:ATTACKER |:DEFENDER] -> (b:Battle
{name: 'Battle of the Blackwater'})

RETURN (k) -[:BELONGS_TO] - (h)
```



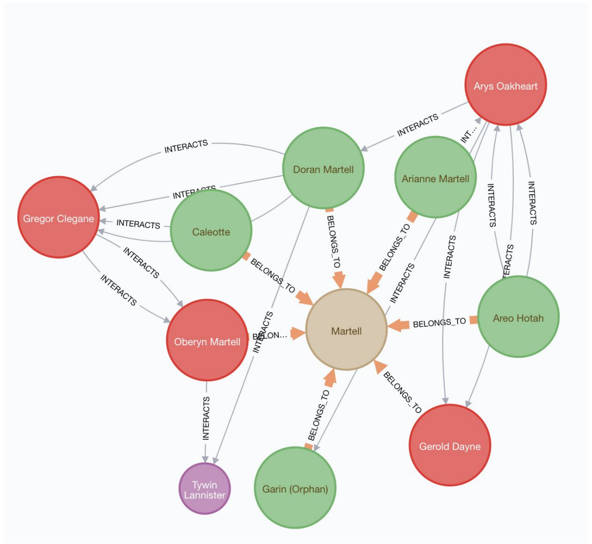
Let’s look at some queries..

Find all people who belonged to “House Lannister” and died between 300 and 600 and whom they interacted with from “House Martell”

```
MATCH (p1:Person) - [:BELONGS_TO] ->
(h:House {name: 'Lannister'})
WHERE 300 <=p1.death_year <=600

MATCH p = (p1) -[:INTERACTS] - (:Person) -
[:BELONGS_TO] - (:House {name : 'Martell'})

RETURN p
```



Let’s look at some queries..

Cluster or group battles based on Kings and Knights participated in them?

…. this isn’t easy to answer with Cypher, but we’ve got **Graph Algorithms (Community detection)** for that!

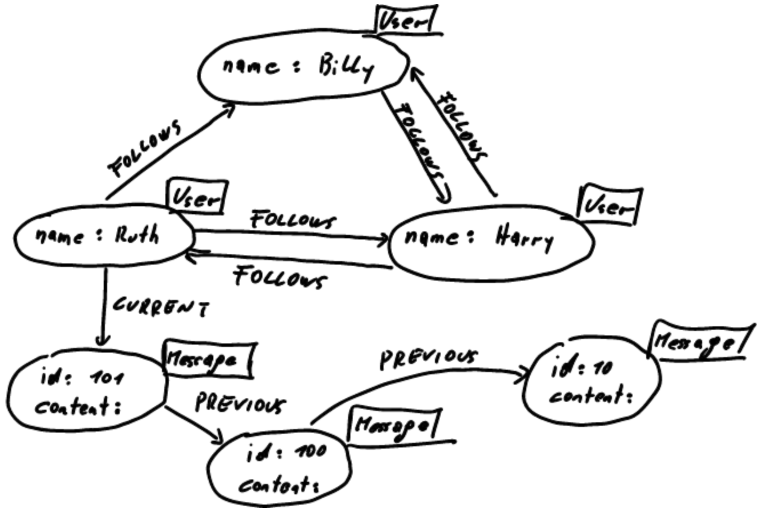
In This Session..

Objective: Understand data modeling for graph data science.

- Considerations prior to schema design
- Monopartite, bipartite & multipartite graphs

The fundamentals don’t change

The white board model is still your starting point



Things that break Neo4j still break Neo4j for data science

- Putting tons of attributes on nodes and relationships
- Making all of your attributes into nodes
- Labels that aren’t sufficiently specific
- Super densely connected nodes

Considerations prior to designing schema

- **Define your questions** and objectives up front
    - Are you running algorithms to answer a specific question? ***or***
    - Are you generating features for an ML pipeline?
- **Which algorithms do you want to run?**
    - Different algorithms require differently shaped graphs
- **How often** do you want to run your workload
    - Is the graph for one offs and experimentation?
    - Is this part of a pipeline?
    - Do you need to do data refreshes? Incremental updates?
    - Are there any time constraints?

What is your objective?

CONSIDERATIONS

OBJECTIVES	Workload Type	Performance Expectation	Data	Data Model
Experimental	Mixed	Variable, no strict requirements	The more the merrier	Domain relevant, easy to refactor
Model building	Primarily Algorithms	Variable, usually <24 hours	Reproducible, timestamps, test/train splits	Follow Best practices, Easy for running algos and updating data
Production ML	Primarily Algorithms	< a few hours	Only the data relevant to your workload	Mono- or bipartite graphs, seeding
Production Analytics	Mixed	< 24 hours	Only the data relevant to your workload	Optimized for algorithms (mono- or bipartite graphs, seeding) + key queries

What are Graph Algorithms?

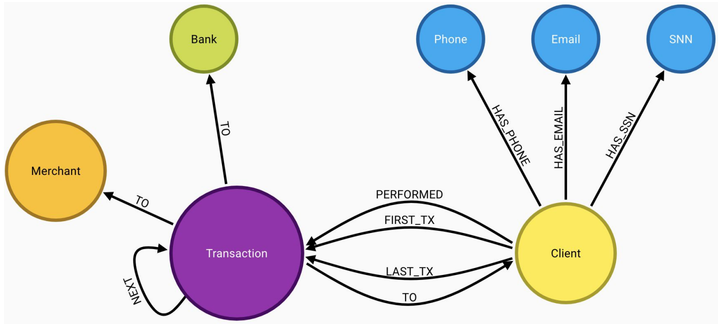
- Graph Algorithms (#Algorithms) solve a wide variety of problems (#Problems) and are applicable in many areas of science and technology (#Applications)

- Algorithms require differently shaped graphs. For example, spanning trees, shortest path and community detection algorithms work on **homogeneous networks** represented by **monopartite** graphs

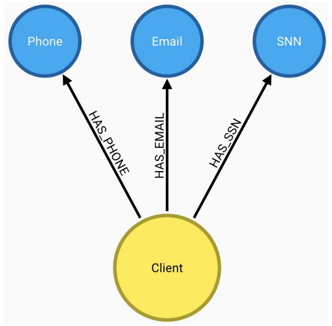
- Similarity algorithms can work on **heterogeneous networks** represented by **bipartite or multipartite** graphs with some strong assumptions

Real world networks are heterogeneous!

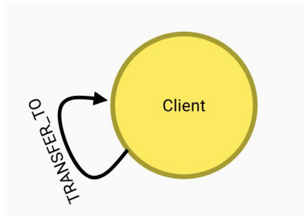
Monopartite, bipartite & multipartite graphs



Most networks contain data with multiple node and relationship types. (**multipartite**)



A **bipartite** graph contains nodes that can be divided into two sets, such that relationships only exists between sets but not within each set.
Node Similarity relies on this type of graph.

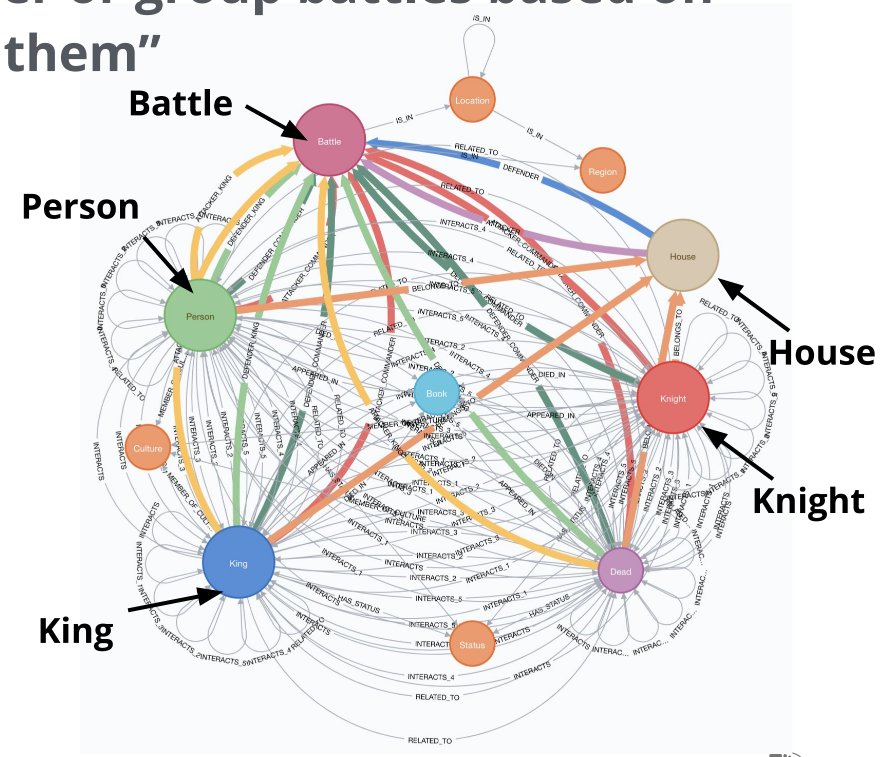


A **monopartite** graph contains one node label and relationship. Most algorithms rely on this type of graph.

Derived relationships and *k-partite* graphs

Let's revisit this question: "Cluster or group battles based on Kings and Knights participated in them"

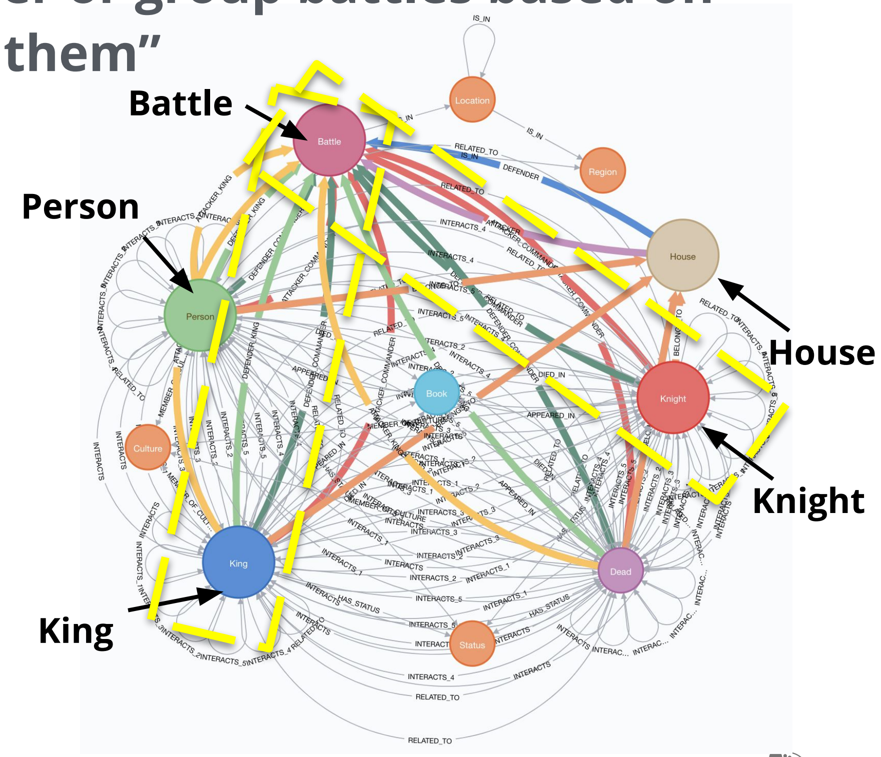
- One class of algorithm -- **community detection** is made to answer this question, but it is intended to work on monopartite graphs
- Can we use **cypher queries** to construct a subgraph with only battles in it? That has information about Kings/Knights that participated in the battles?



Derived relationships and *k-partite* graphs

Let's revisit this question: "Cluster or group battles based on Kings and Knights participated in them"

- One class of algorithm -- **community detection** is made to answer this question, but it is intended to work on monopartite graphs
- Can we use **cypher queries** to construct a subgraph with only battles in it? That has information about Kings/Knights that participated in the battles?

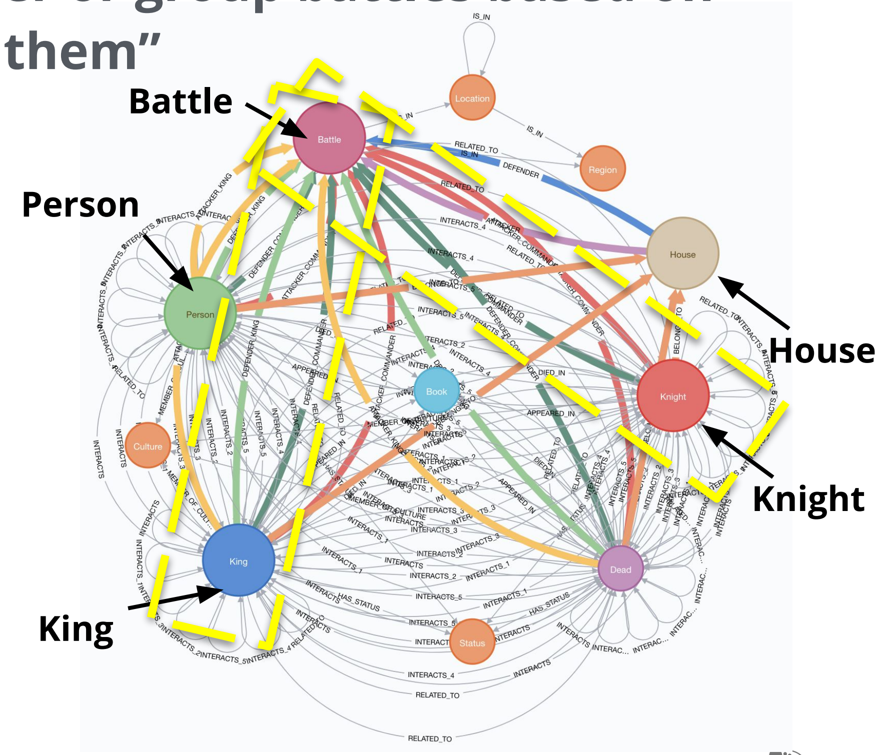


Derived relationships and *k-partite* graphs

Let's revisit this question: "Cluster or group battles based on Kings and Knights participated in them"

Monopartite = only 1 node label

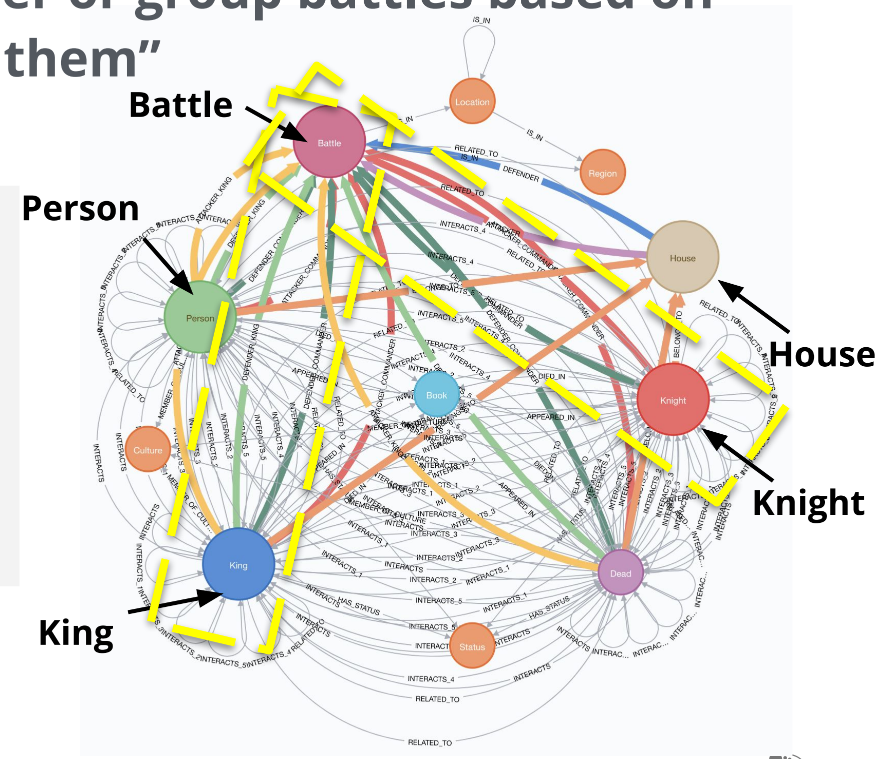
- **Connect battles to each other**
- Based on how many Knights and Kings fought in the ***same*** battle



Derived relationships and *k-partite* graphs

Let's revisit this question: "Cluster or group battles based on Kings and Knights participated in them"

```
MATCH
(b1:Battle)<-[]-(k)-[]->(b2:Battle)
WHERE k:King or k:Knight
RETURN b1.name AS source, b2.name
AS target, count(distinct k) AS
num_participants
```



This gives you:

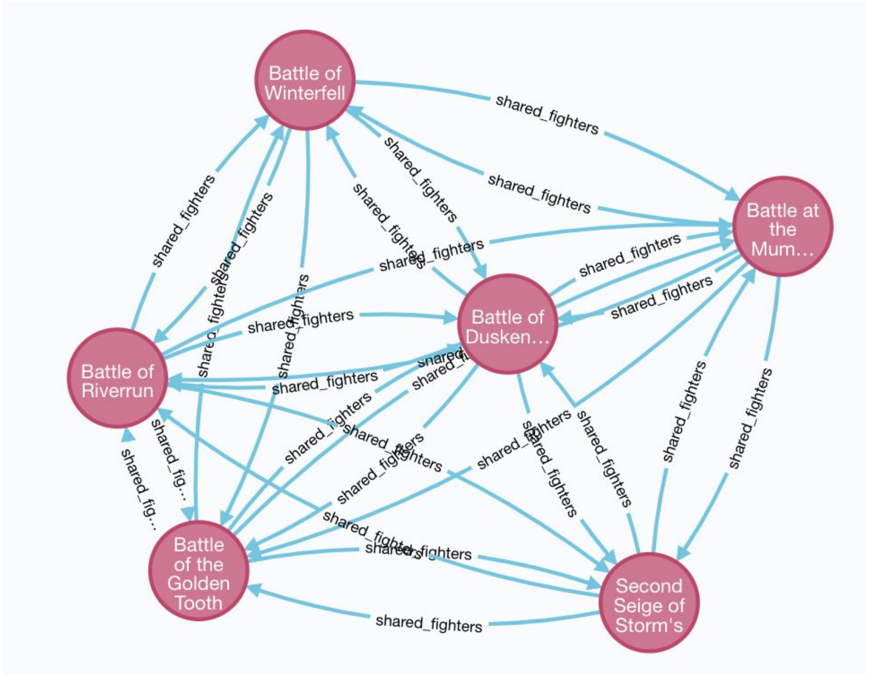
- for every pair of battles,
- how many kings or knights fought in both

Derived relationships and *k-partite* graphs

Let's revisit this question: "Cluster or group battles based on Kings and Knights participated in them"

```
MATCH
(b1:Battle)<-[]-(k)-[]->(b2:Battle)
WHERE k:King or k:Knight
WITH b1, b2 count(distinct k) AS
num_participants
MERGE (b1)-[r:shared_fighters
{count:num_participants}]->(b2)
RETURN b1,r,b2
```

We can add the new *derived* relationship to the graph and extract the relevant *subgraph*

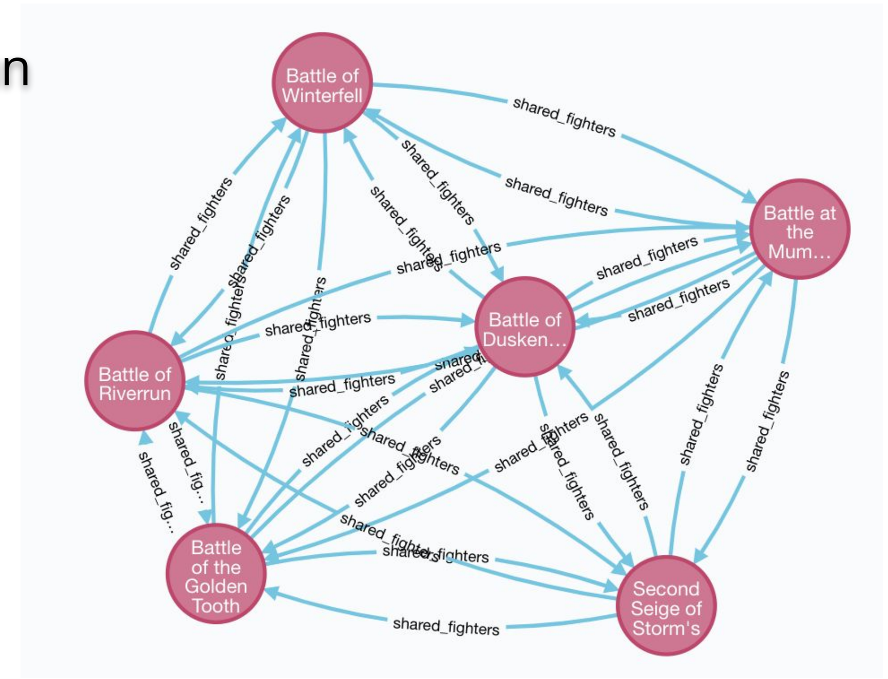


Derived relationships and *k-partite* graphs

Let's revisit this question: "Cluster or group battles based on Kings and Knights participated in them"

Now that we have the graph we can run our algorithms on ....

we can use one of the **Community detection algorithms** to cluster battles.



Why can’t I run my algorithm on a multipartite graph?

Algorithms learn based on the structure of a graph, and make assumptions about graph topology:

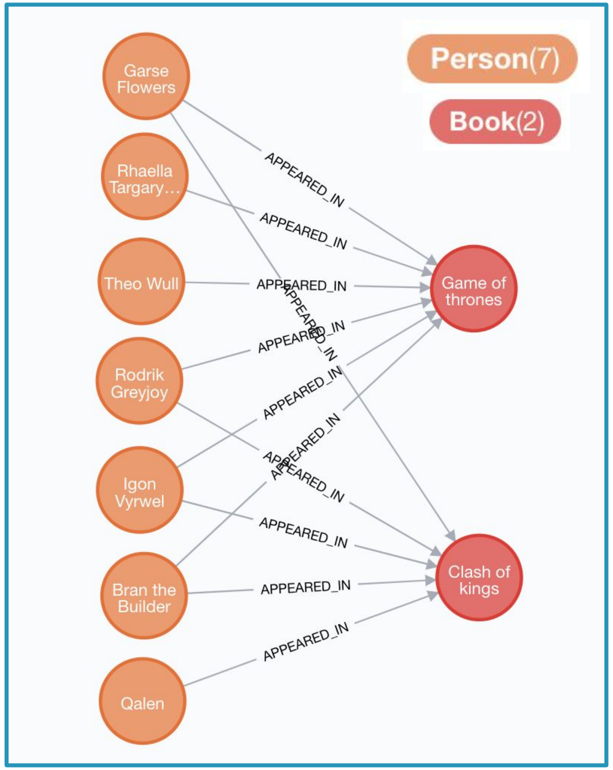
- Any variation in the distribution of relationships for nodes can tell us some underlying property (how important is a node? what community is it in?)
- It’s not possible to *iterate* heuristics over directed graphs
- There’s no external information relevant to the algorithm (like node labels or relationship type)

huh?

Why can’t I run my algorithm on a multipartite graph?

What if I try to an algorithm on this graph?

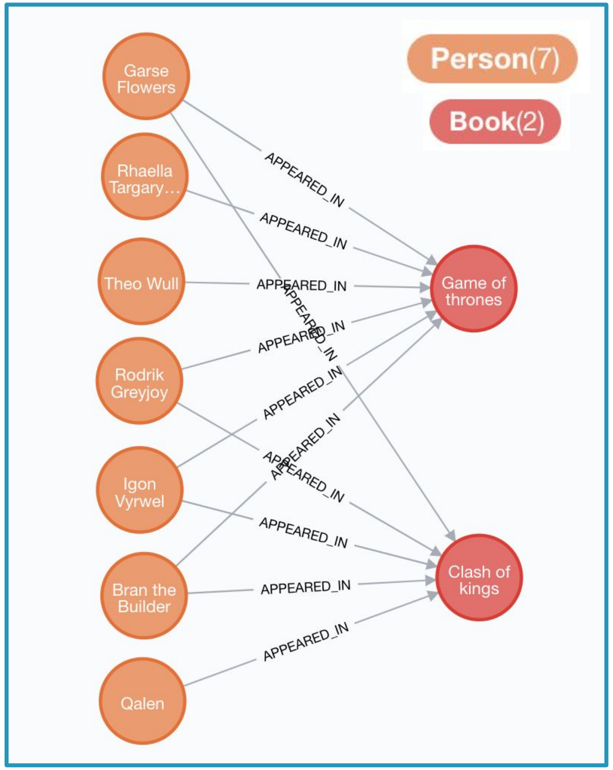
- How many relationships does each **person** have?
- How many relationships does each **book** have?
- What is the ***direction*** of the relationships in this graph?
- Can I reach a person node from another person node, following the directed relationships?



Why can't I run my algorithm on a multipartite graph?

What if I try to an algorithm on this graph?

- How many relationships does each **person** have? **1 or 2**
- How many relationships does each **book** have? **5 or 6**
- What is the ***direction*** of the relationships in this graph? `Person-[:APPEARED_IN]->Book`
- Can I reach a person node from another person node, following the directed relationships? **No!**

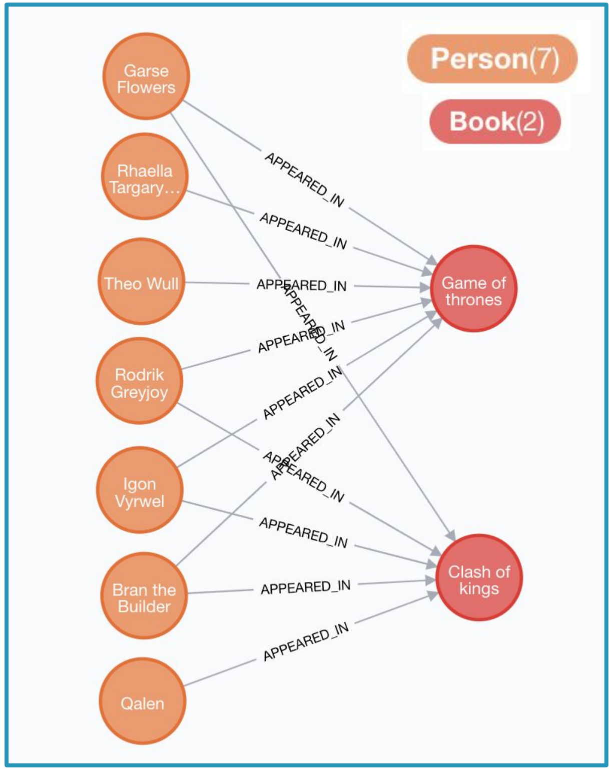


Why can't I run my algorithm on a multipartite graph?

What if I try to an algorithm on this graph?

- What would an algorithm that used the number of edges each node has to calculate *centrality* conclude?

- What would an algorithm that followed directed relationships to find *communities* conclude?



Why can’t I run my algorithm on a multipartite graph?

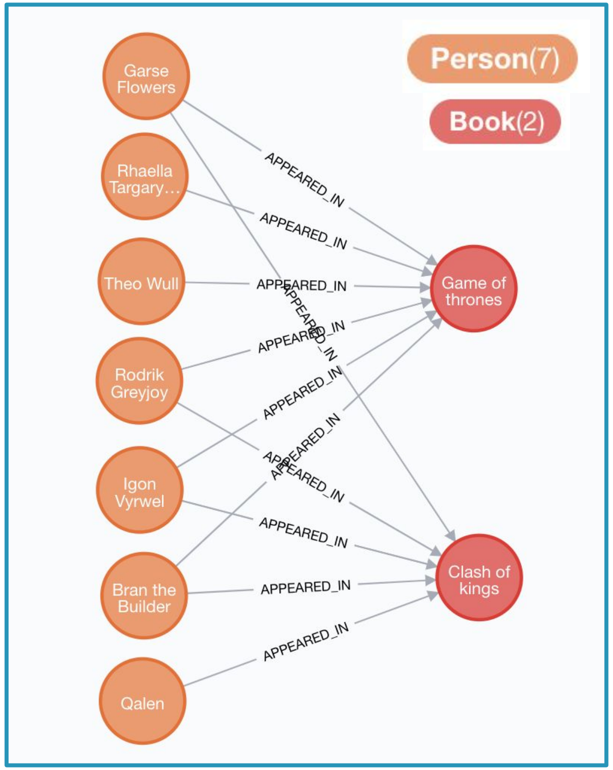
What if I try to an algorithm on this graph?

- What would an algorithm that used the number of edges each node has to calculate *centrality* conclude?

  Books are more important than people!

- What would an algorithm that followed directed relationships to find *communities* conclude?

  There seven communities?



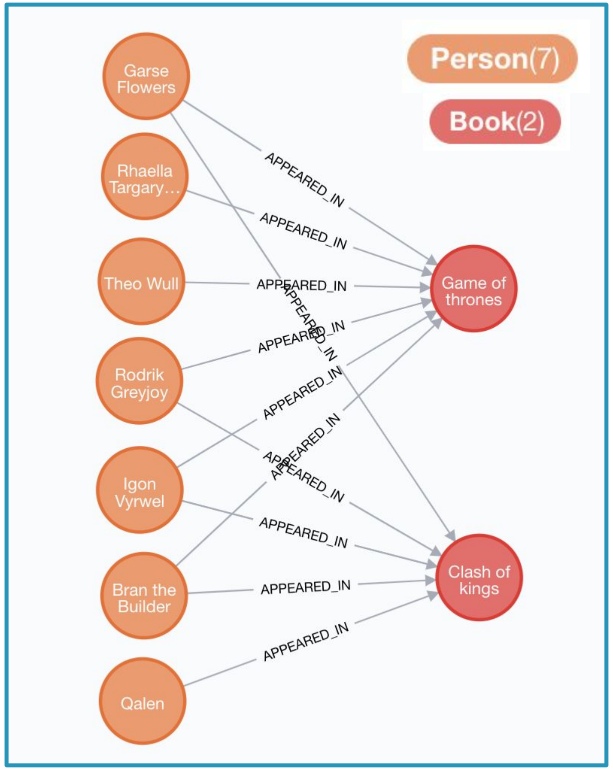
Why can't I run my algorithm on a multipartite graph?

If you want to find out:

- What person is the most important

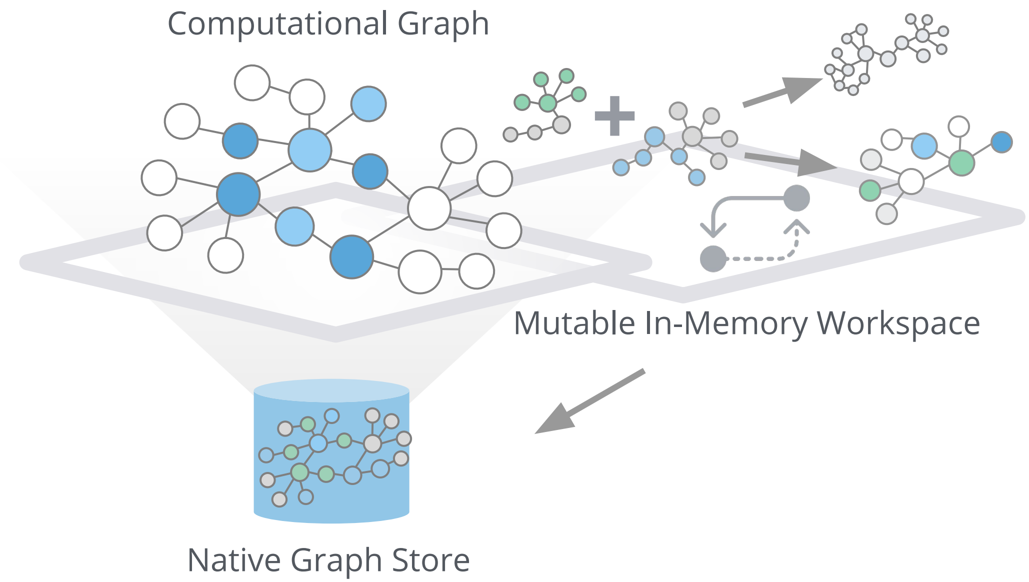
- How many communities of people are there, across all the books?

You need to ***reshape your graph!***



Neo4j's Graph Catalog solves this problem!

We have procedures to let you reshape and subset your transactional graph so you have the right data in the right shape.

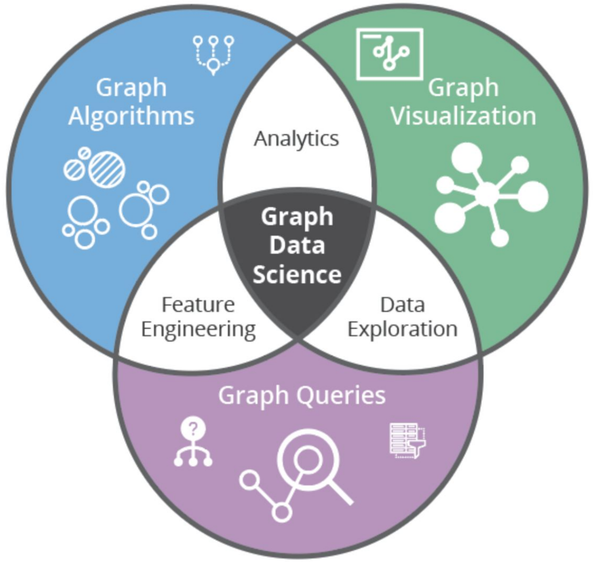


- Neo4j automates data transformations
- Fast iterations & layering
- Production ready features, parallelization & enterprise support

Next Session..

Deep dive into Graph Data Science Library

- Graph Catalog
- Graph Algorithms


Graph Algorithms
Graph Visualization
Analytics
Graph Data Science
Feature Engineering
Data Exploration
Graph Queries