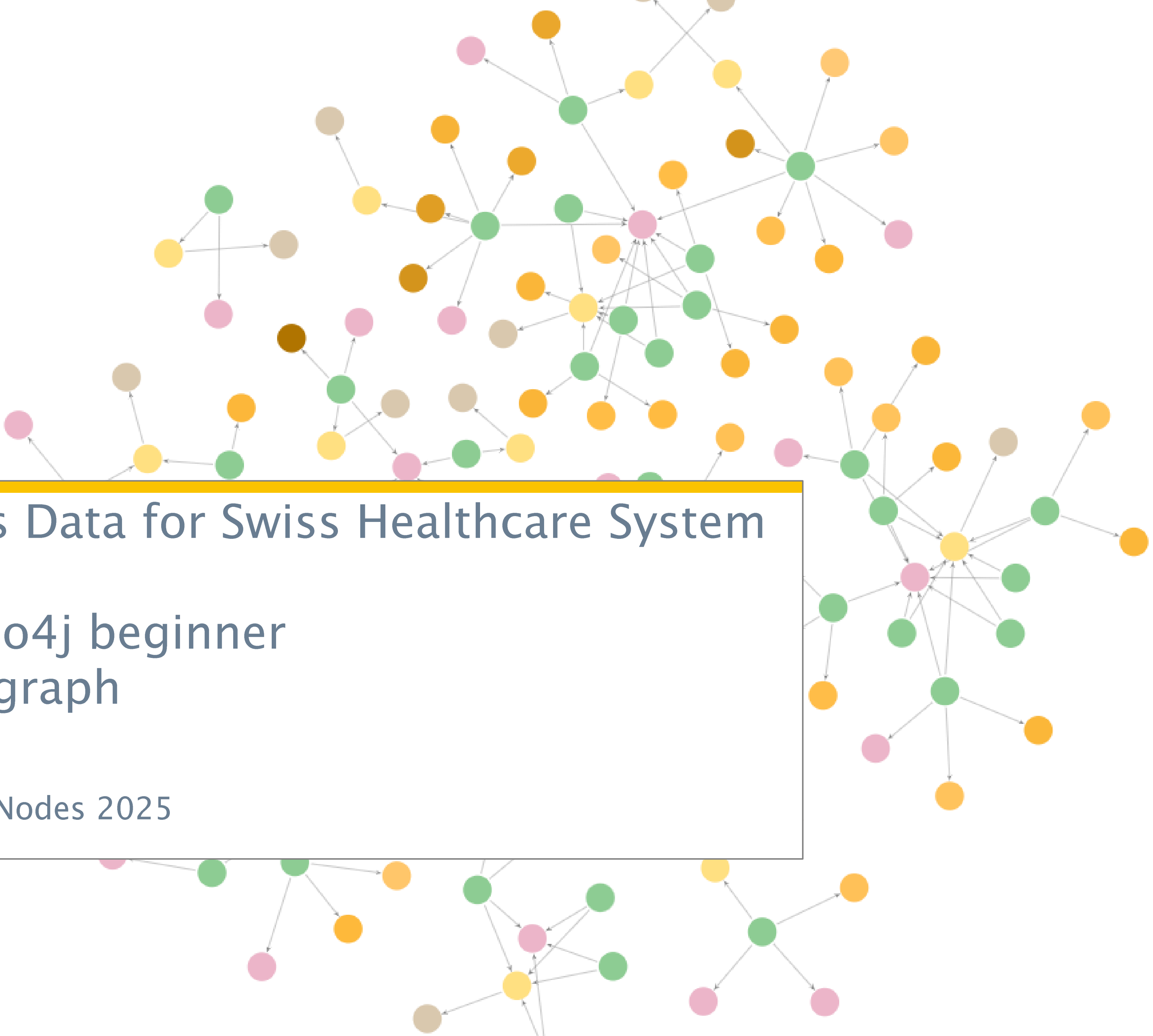
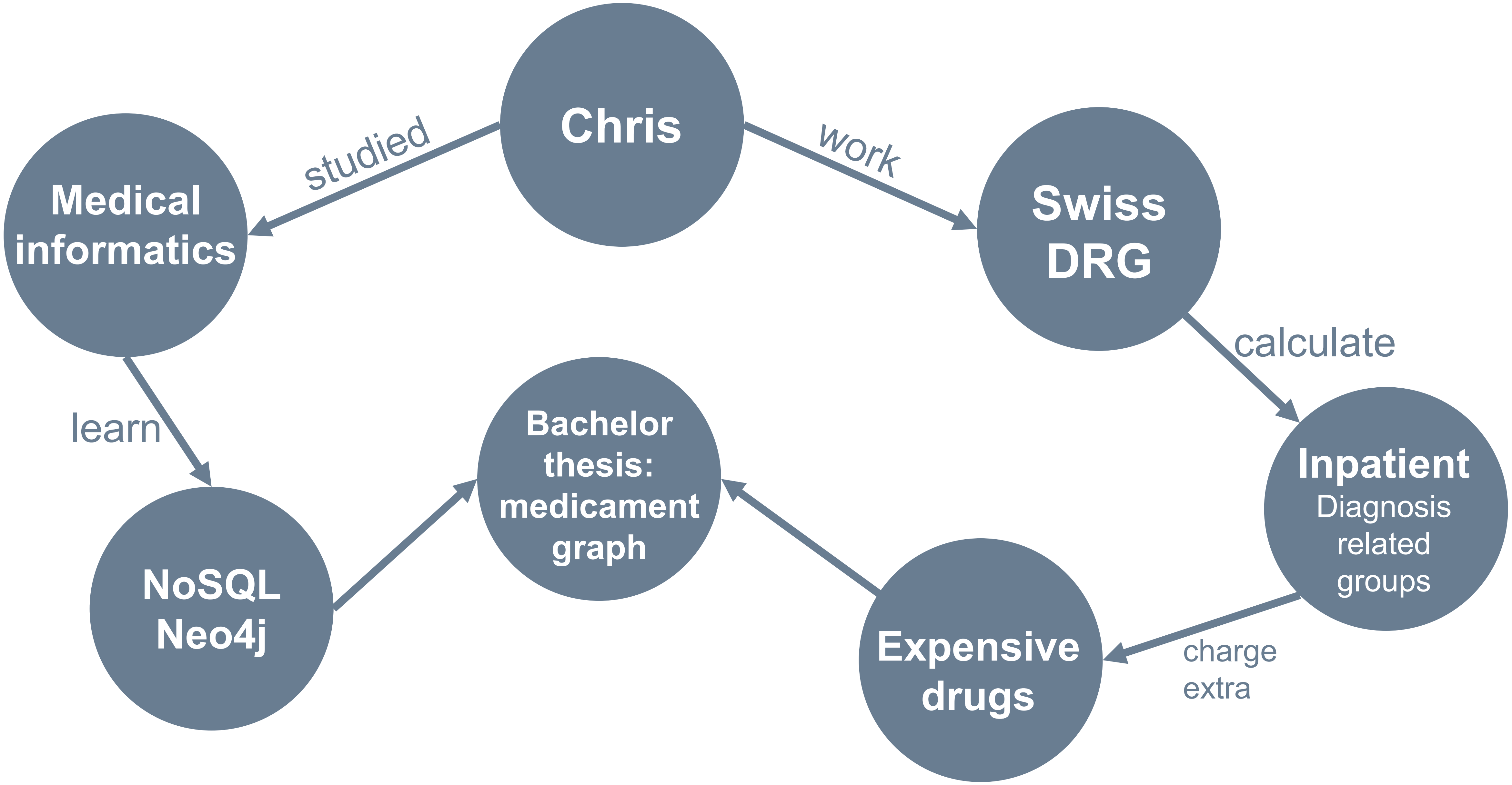
Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

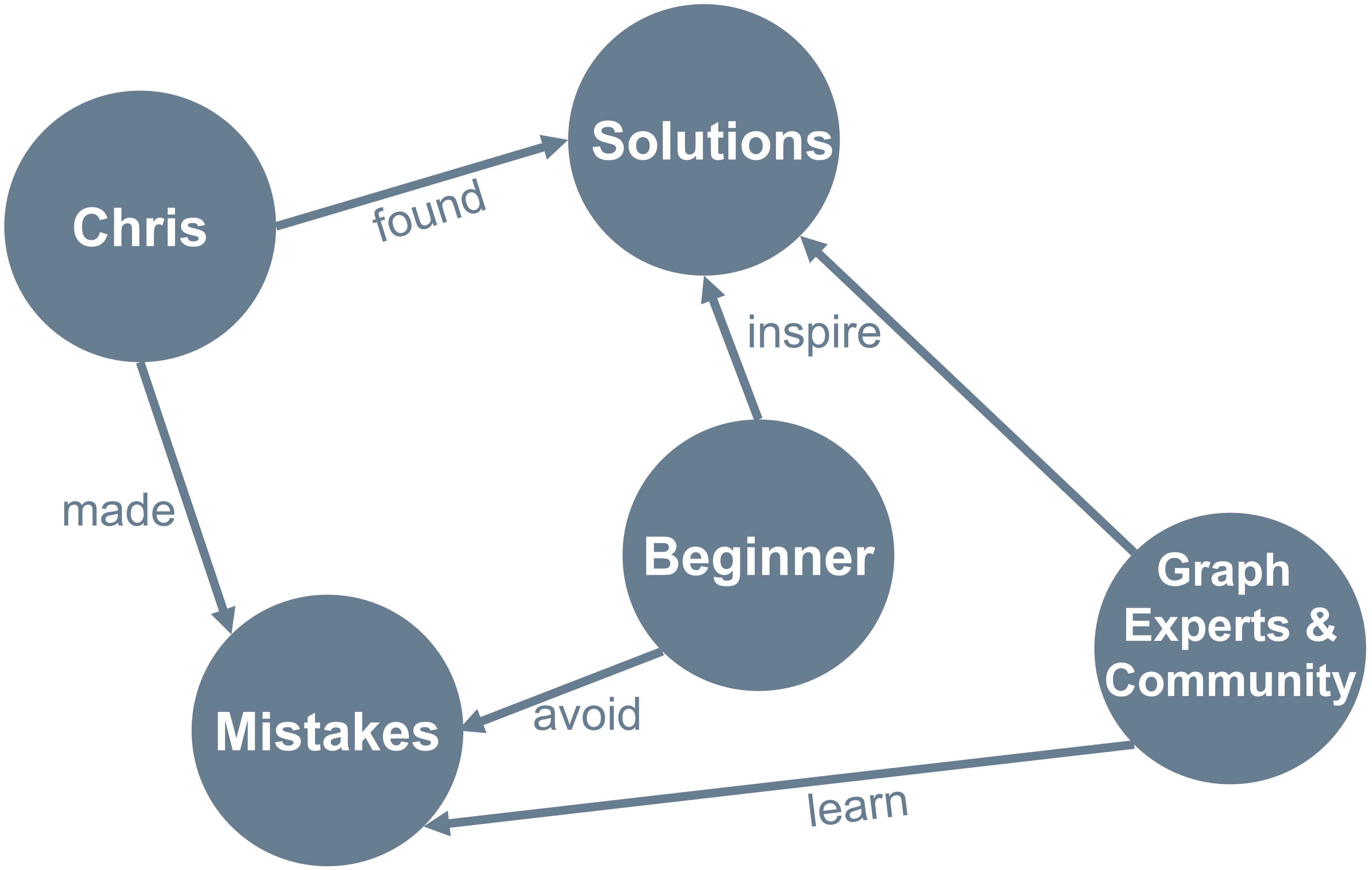


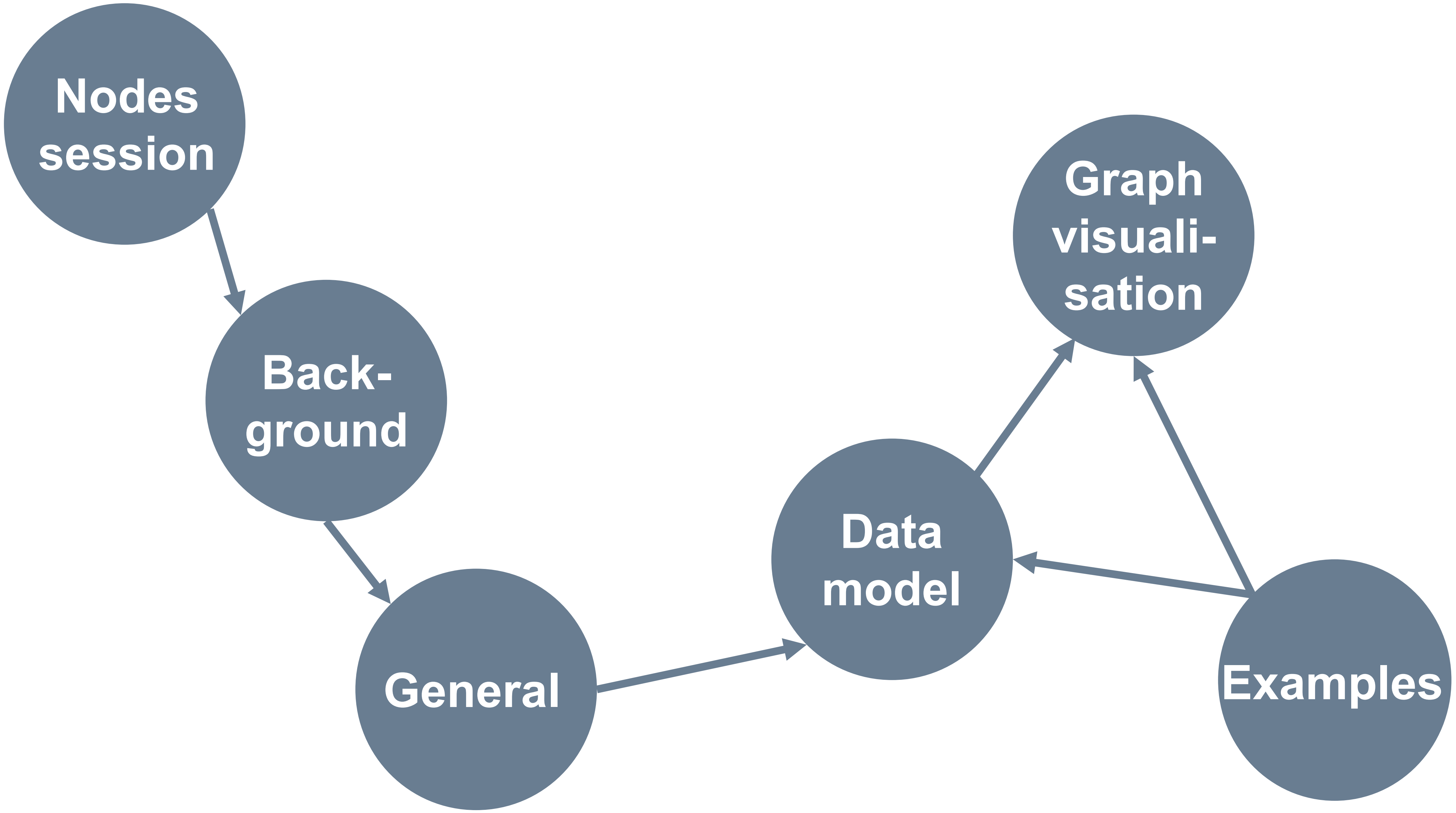
Knowledge Graph of Drugs Data for Swiss Healthcare System

Tipps and tricks from a Neo4j beginner for your (first knowledge) graph

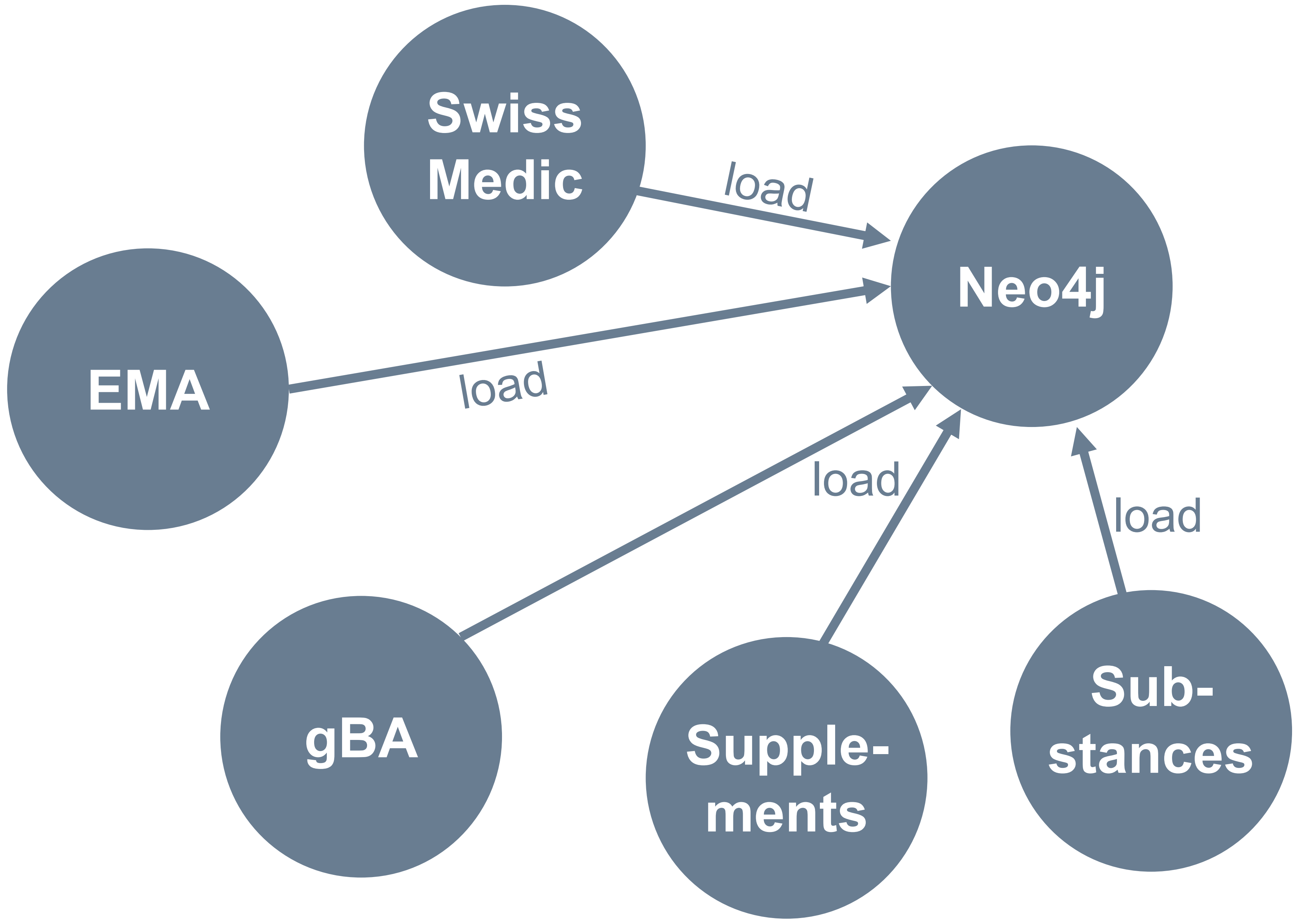
Christian Franke, 6th November 2025, Nodes 2025


Chris
studied
work
Medical informatics
Swiss DRG
calculate
learn
Inpatient
Diagnosis related groups
Bachelor thesis: medicament graph
NoSQL Neo4j
Expensive drugs
charge extra

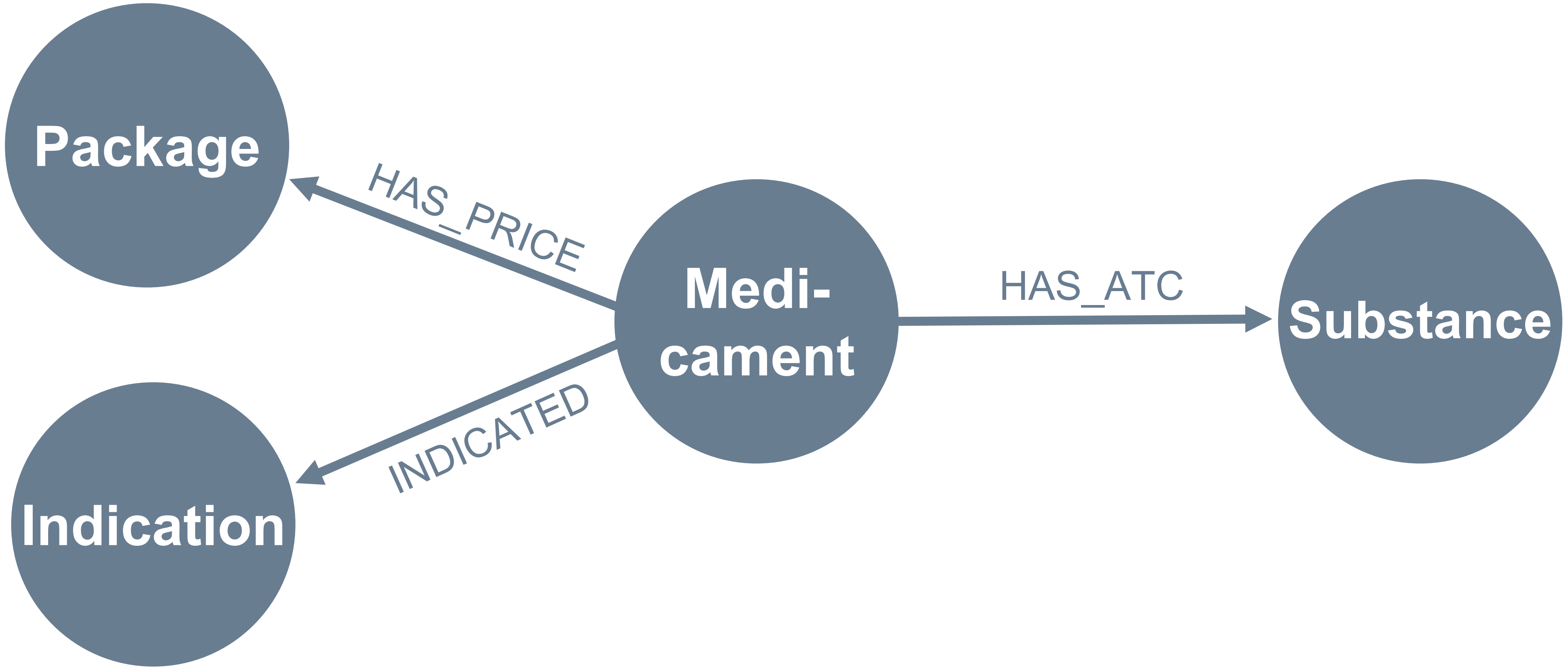

Solutions
Chris
found
inspire
made
Beginner
Graph Experts & Community
Mistakes
avoid
learn


Nodes session
Back-ground
General
Data model
Graph visuali-sation
Examples

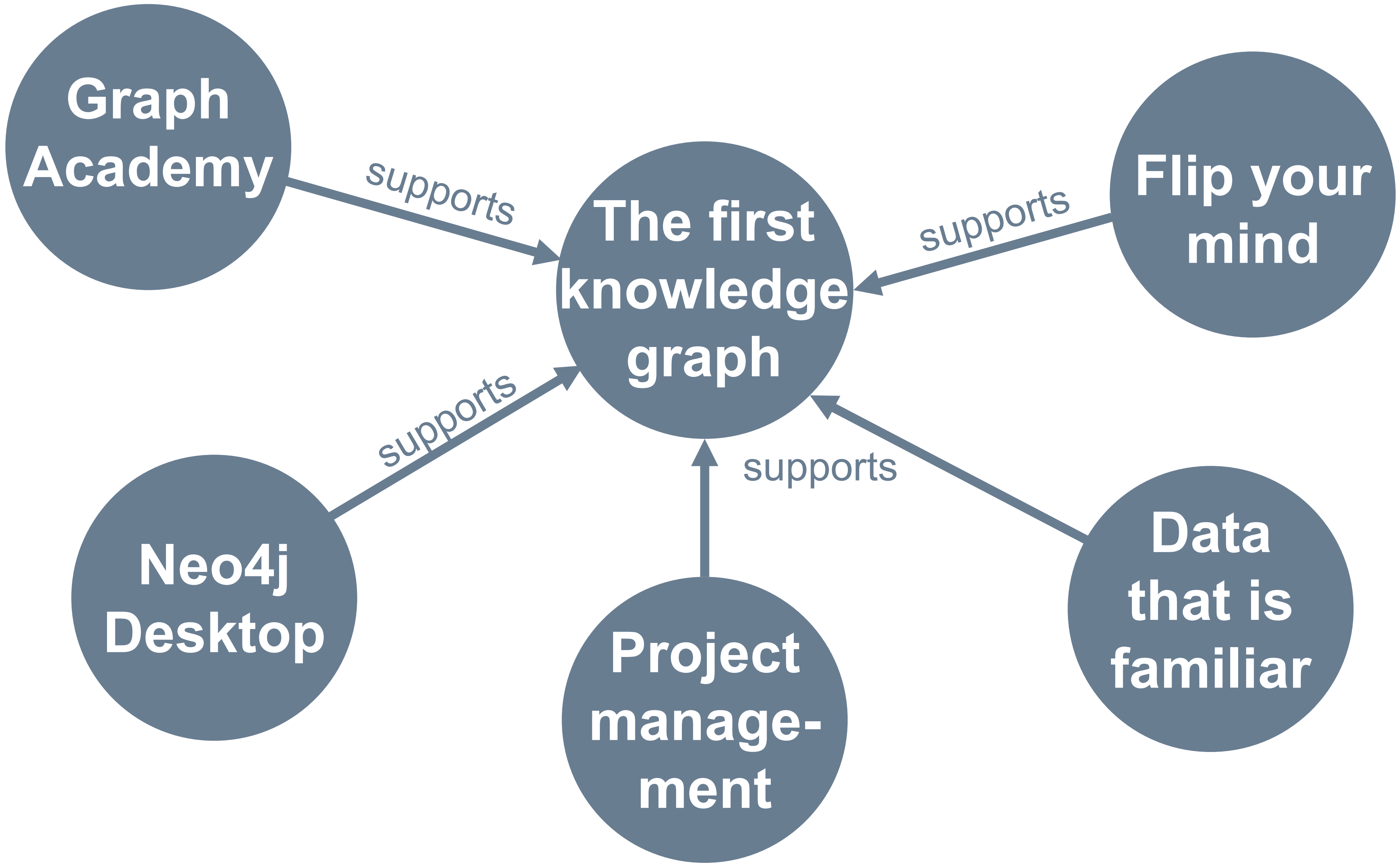
Back-
ground


Swiss Medic
load
Neo4j
EMA
load
load
load
gBA
Supple-ments
Sub-stances

Simplified data model

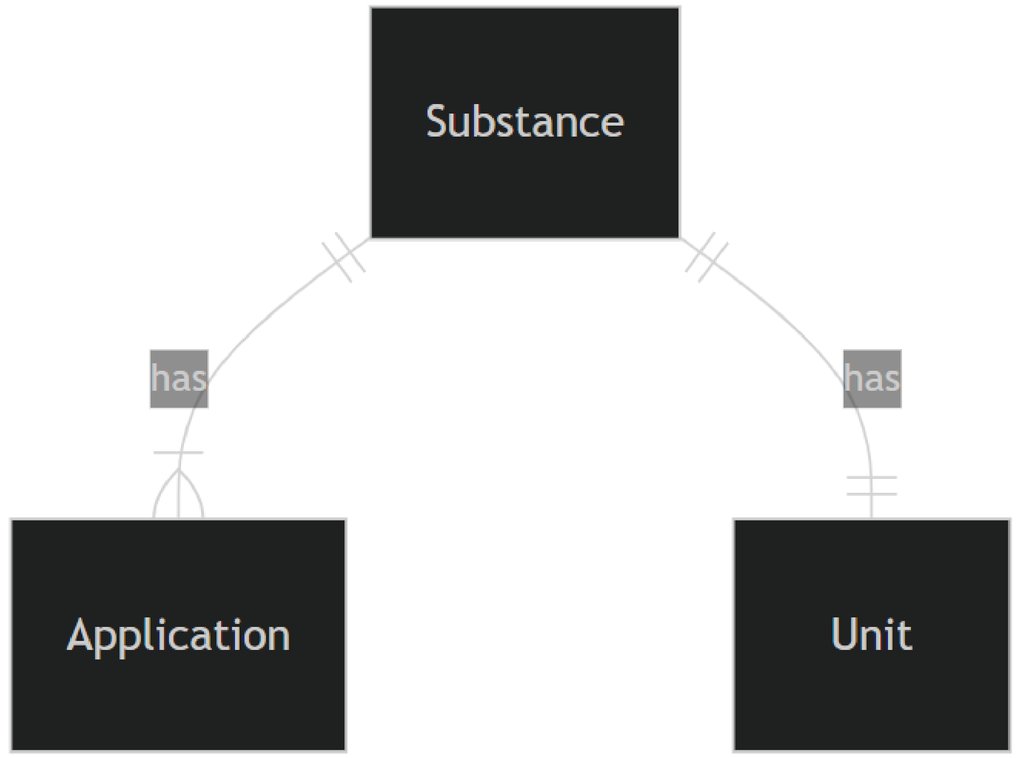


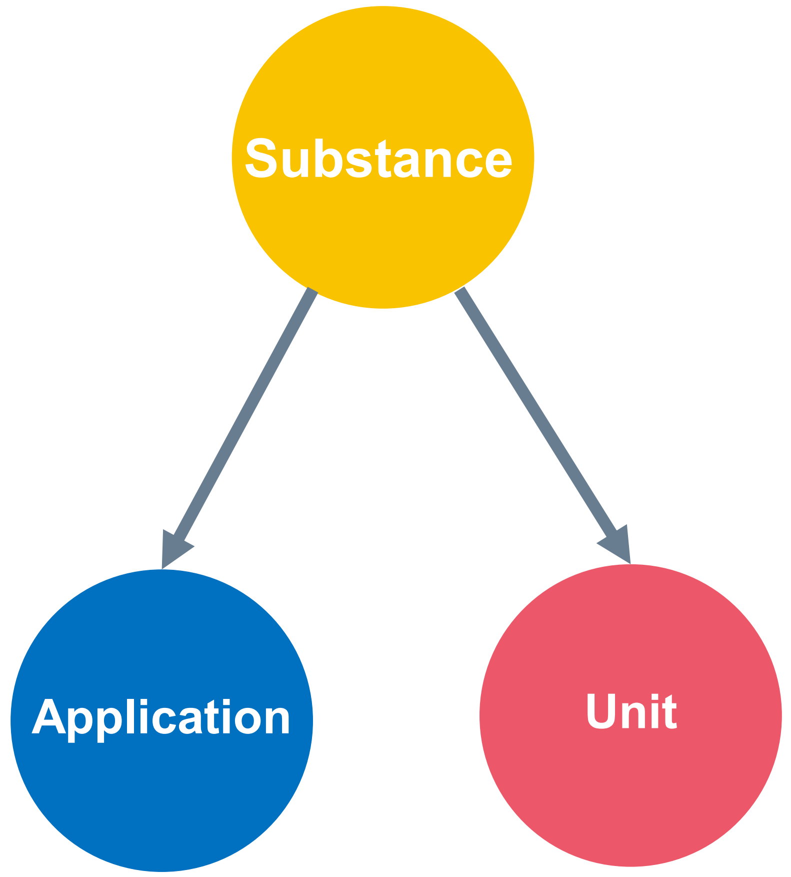
General


Graph Academy
supports
The first knowledge graph
supports
Flip your mind
supports
Neo4j Desktop
supports
Project manage-ment
Data that is familiar

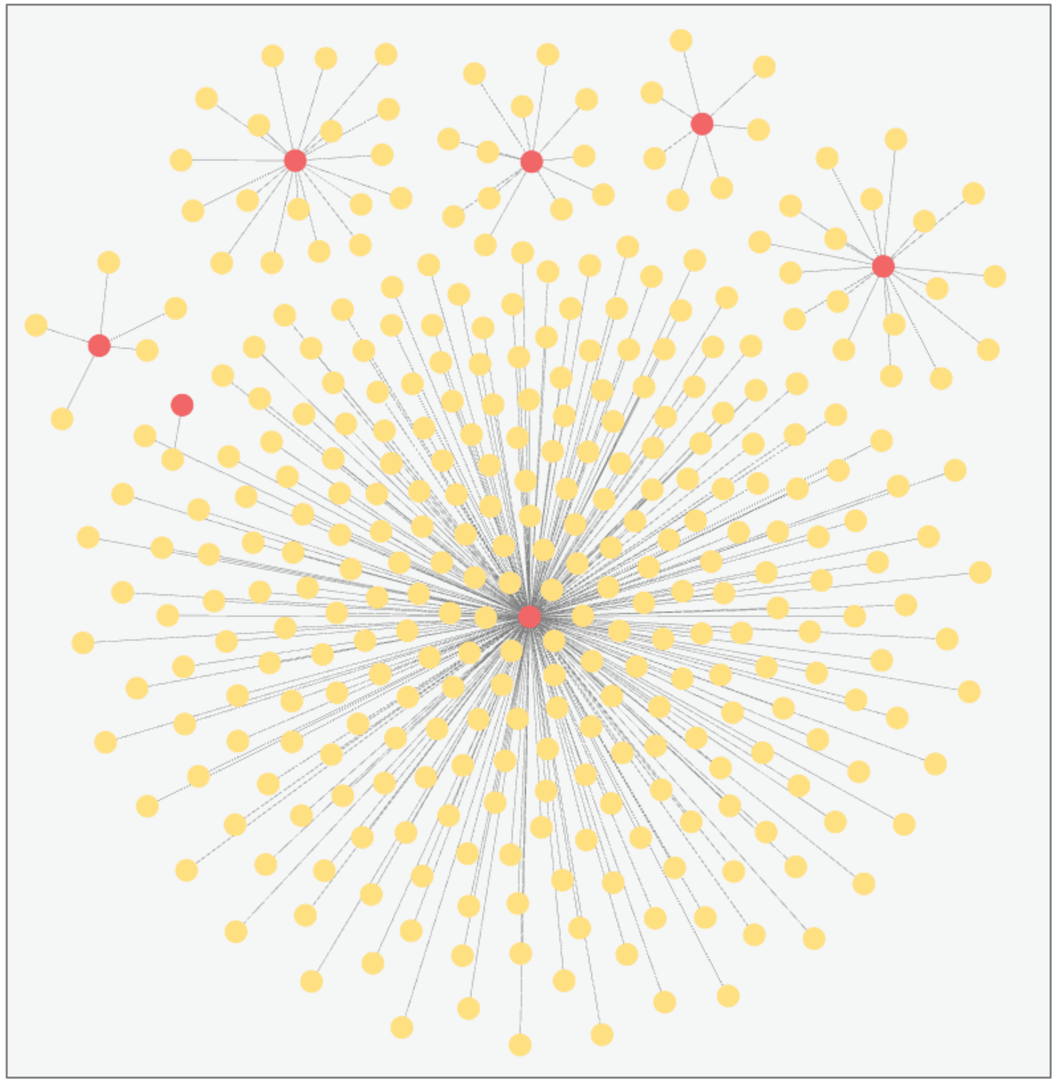
Example 1

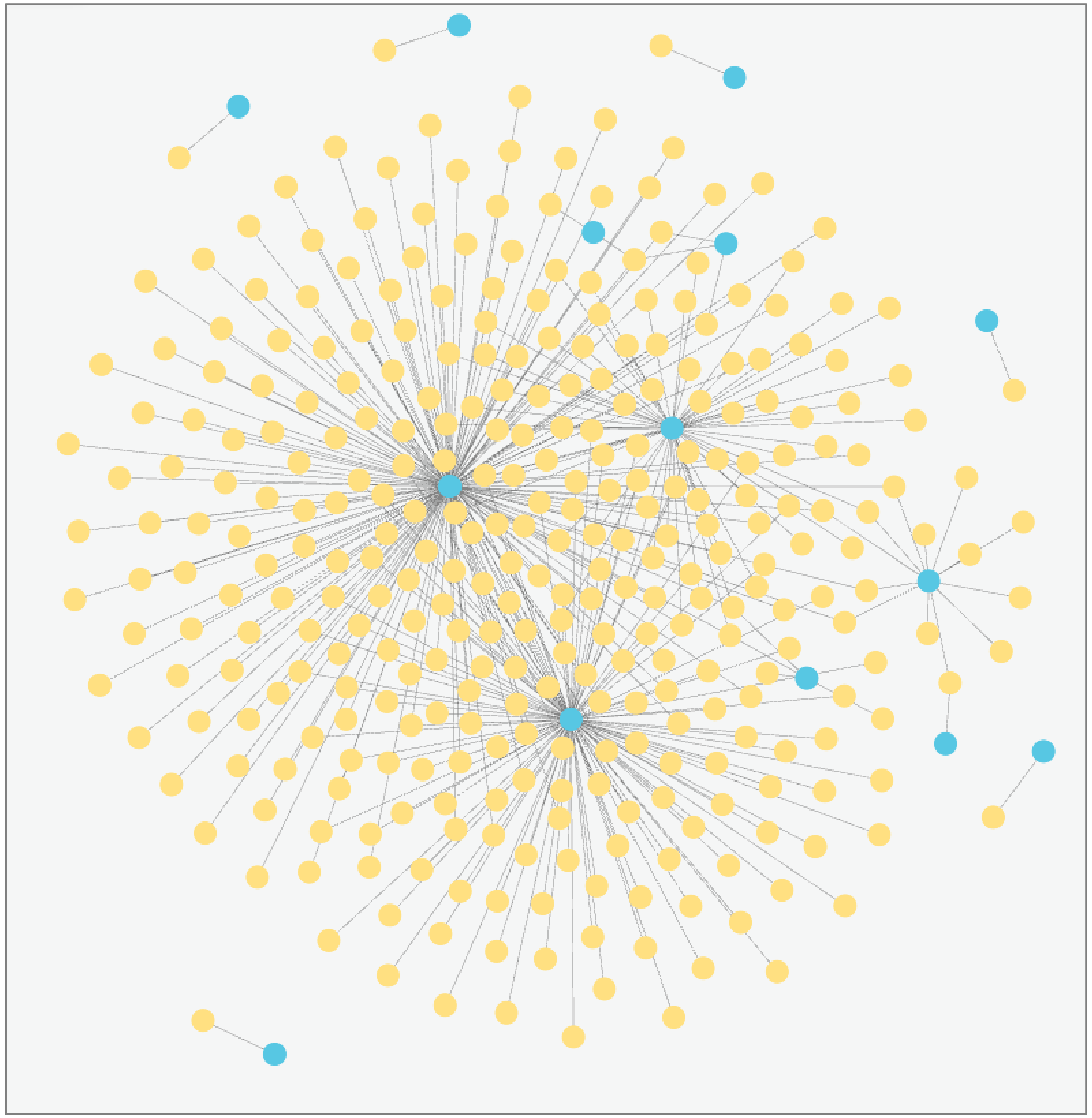
Creating labels and nodes can be easy.



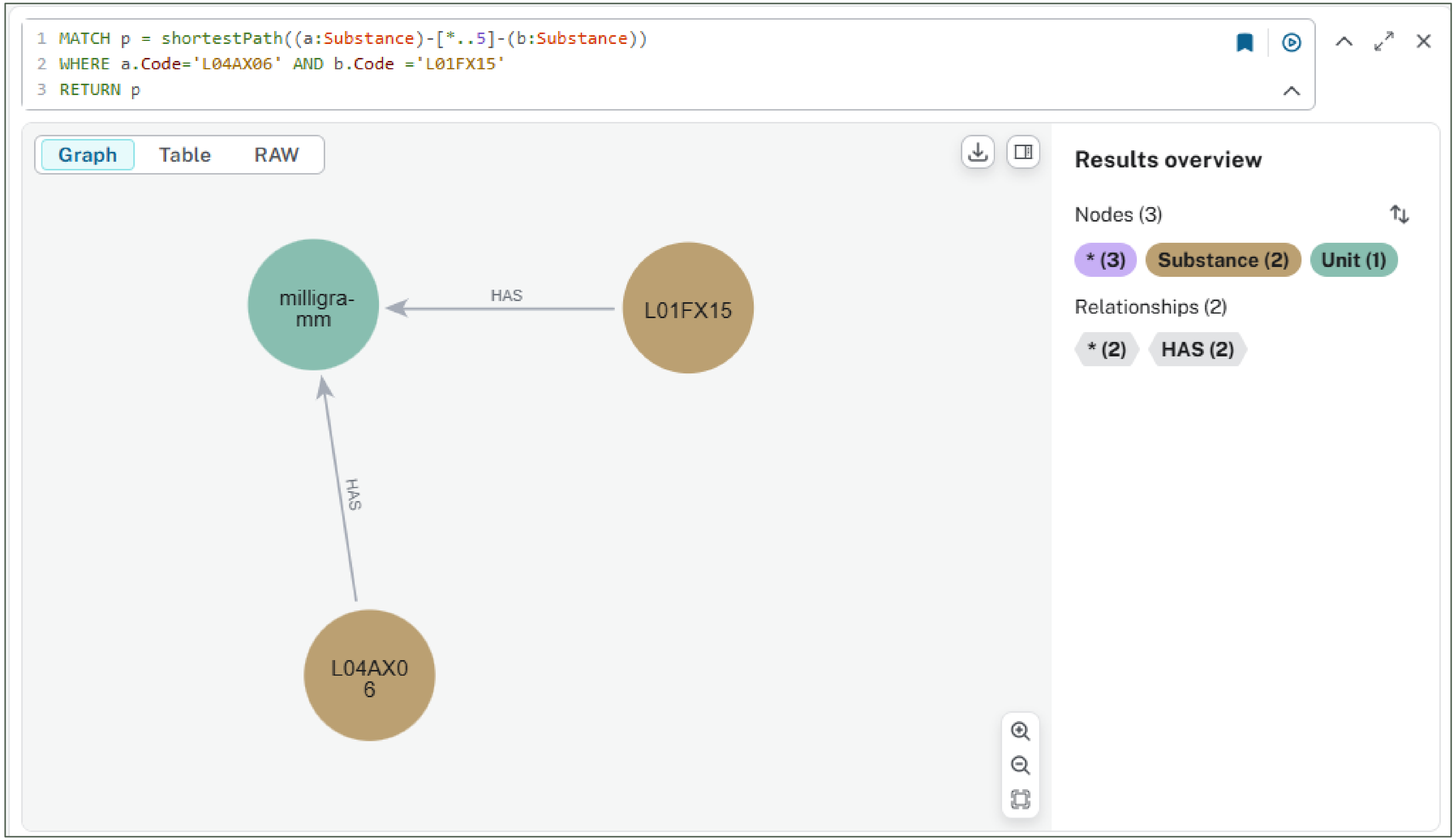


Just start with well know data.



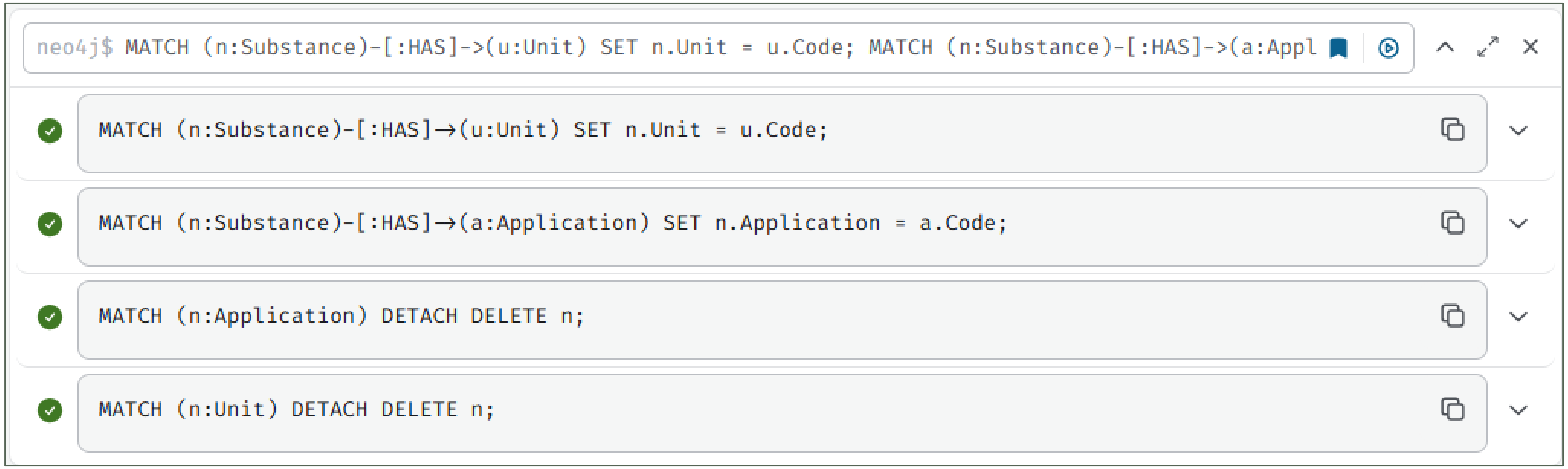


Graph algorithms are useless with "common attributes"



```
MATCH p = shortestPath((a:Substance)-[*..5]-(b:Substance))
WHERE a.Code='L04AX06' AND b.Code ='L01FX15'
RETURN p
```

After adding data as attributes to nodes and deleting the “common labels”...



```
neo4j$ MATCH (n:Substance)-[:HAS]->(u:Unit) SET n.Unit = u.Code; MATCH (n:Substance)-[:HAS]->(a:Appl
```

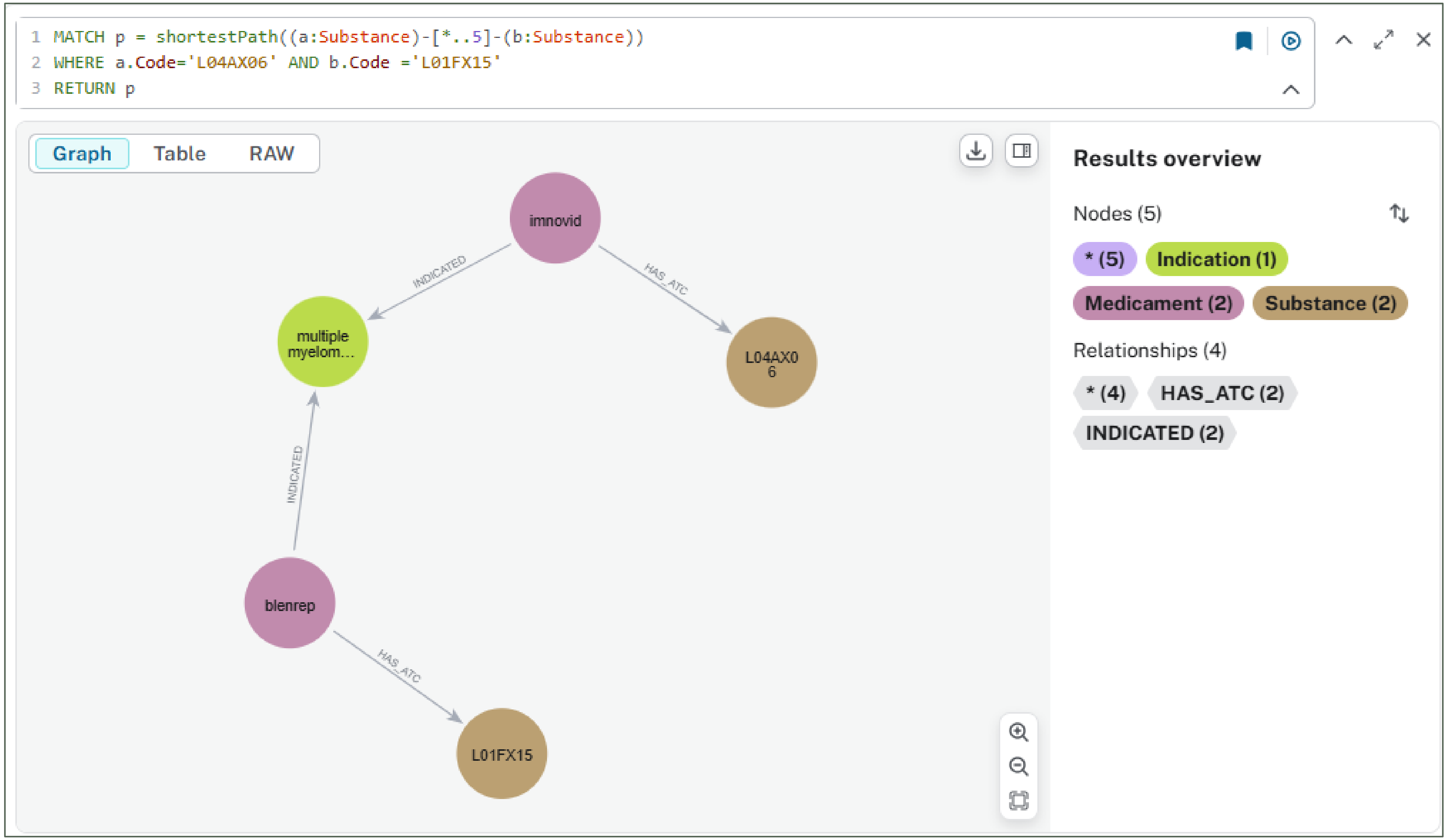
```
MATCH (n:Substance)-[:HAS]→(u:Unit) SET n.Unit = u.Code;
```

```
MATCH (n:Substance)-[:HAS]→(a:Application) SET n.Application = a.Code;
```

```
MATCH (n:Application) DETACH DELETE n;
```

```
MATCH (n:Unit) DETACH DELETE n;
```

… shortest path can be useful!



```
MATCH p = shortestPath((a:Substance)-[*..5]-(b:Substance))
WHERE a.Code='L04AX06' AND b.Code ='L01FX15'
RETURN p
```

Results overview

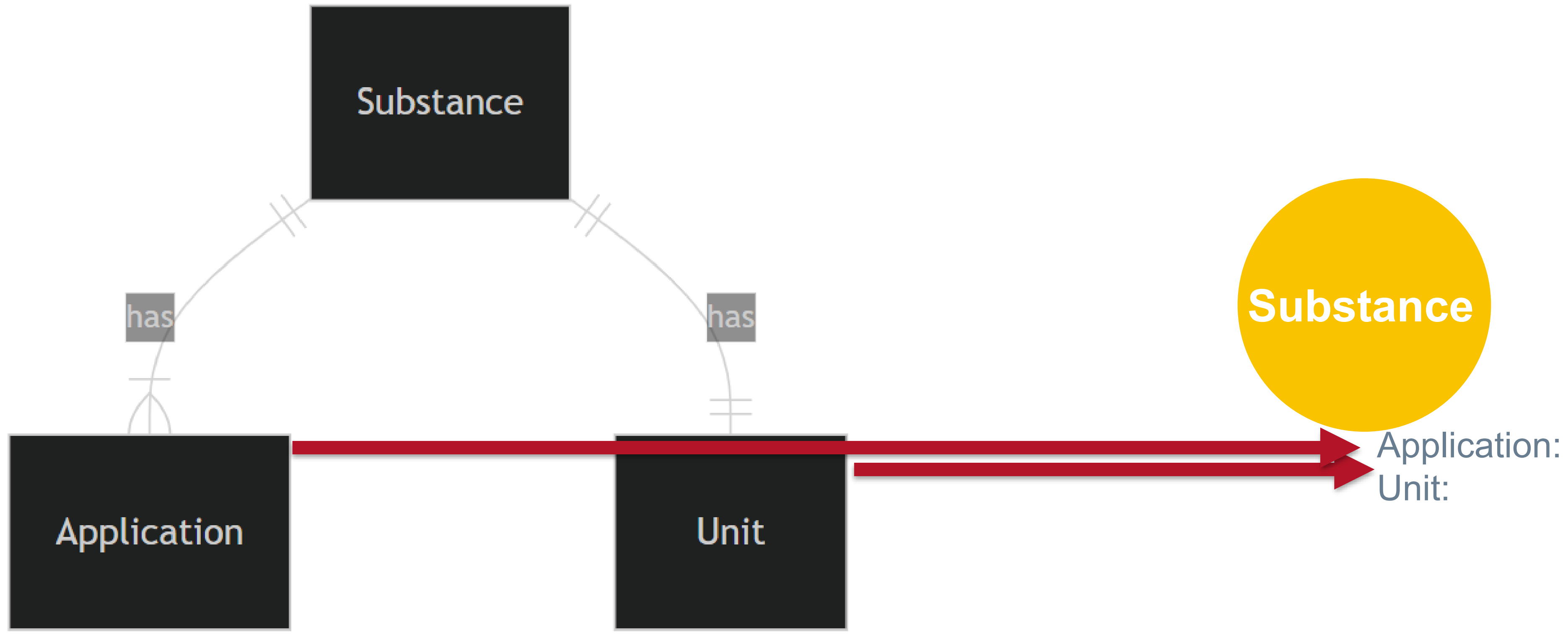
Nodes (5)

* (5) Indication (1) Medicament (2) Substance (2)

Relationships (4)

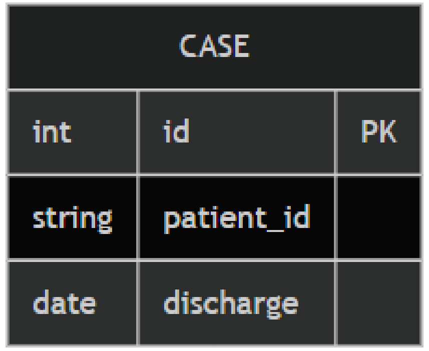
* (4) HAS_ATC (2) INDICATED (2)

Dimension tables (with only few rows) could be modelled as attributes and not as own labels.

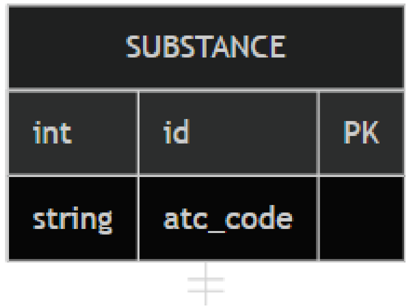


Example 2

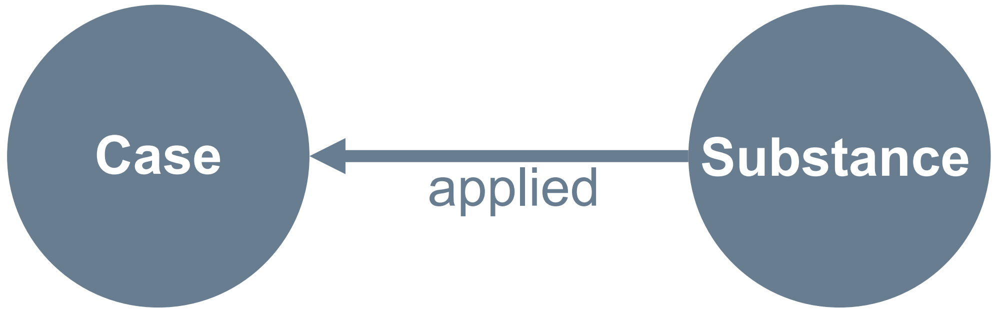
n:m table = graph relation



CASE		
int	id	PK
string	patient_id	
date	discharge	



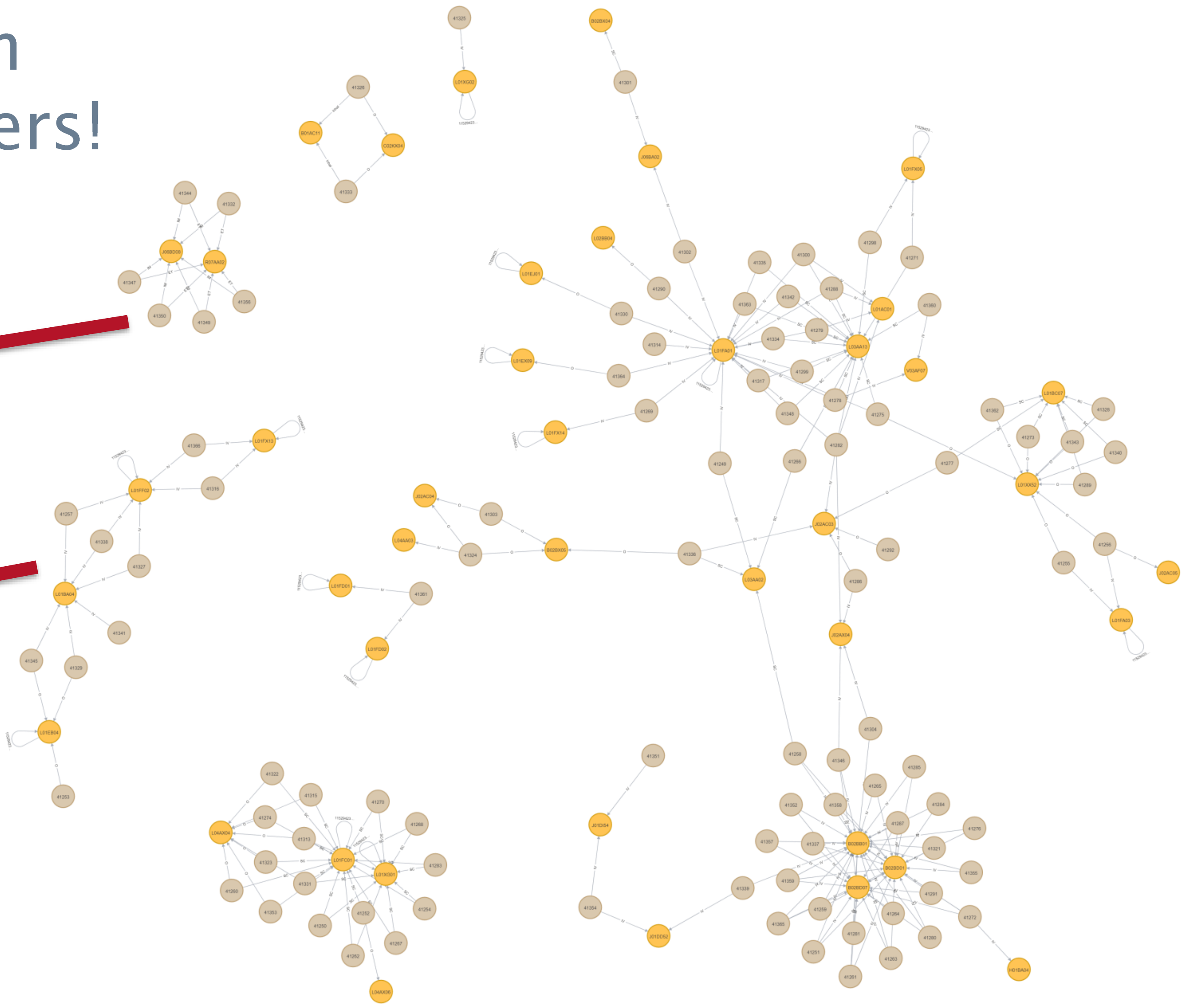
SUBSTANCE		
int	id	PK
string	atc_code	



```
LOAD CSV WITH HEADERS FROM 'file:///case_substance.csv' AS row
FIELDTERMINATOR ';'
MATCH (c:Case {id: toInteger(row.case_id)})
MATCH (s:Substance {id: toInteger(row.substance_id)})
MERGE (s)-[:APPLIED {dose: toInteger(row.dose)}]->(c)
```

n:m table = graph relation
Not correlations but clusters!

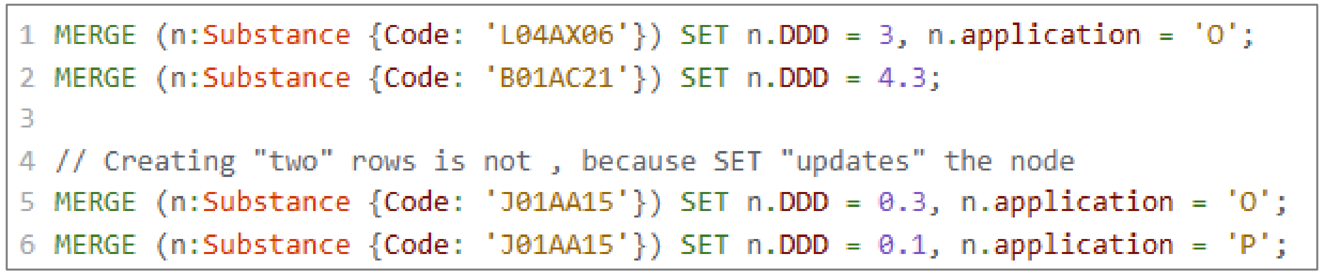
	A	B	C	D	E	F	…	N
A	6	6						
B		6						
C			3	2				
D				6	3			
E					5	2		
F						2		
…							…	
N								…



Example 3

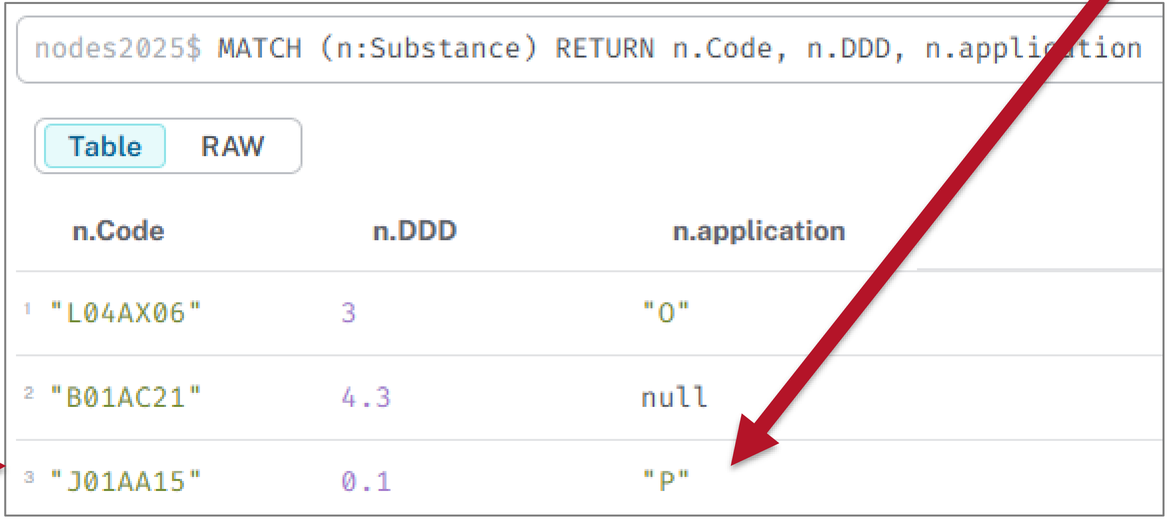
MERGE … SET is like UPDATE, if nodes exists
MERGE … SET is like INSERT, if nodes dose not exists

Code	DDD	Application
L04AX06	3	O
B01AC21	4.3	
J01AA15	0.3, 0.1	O, P



```
MERGE (n:Substance {Code: 'L04AX06'}) SET n.DDD = 3, n.application = 'O';
MERGE (n:Substance {Code: 'B01AC21'}) SET n.DDD = 4.3;

// Creating "two" rows is not , because SET "updates" the node
MERGE (n:Substance {Code: 'J01AA15'}) SET n.DDD = 0.3, n.application = 'O';
MERGE (n:Substance {Code: 'J01AA15'}) SET n.DDD = 0.1, n.application = 'P';
```

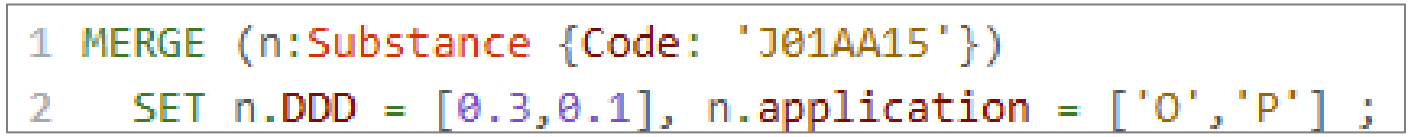
Code	DDD	Application
L04AX06	3	
B01AC21	4.3	
J01AA15	0.3	O
J01AA15	0.1	P



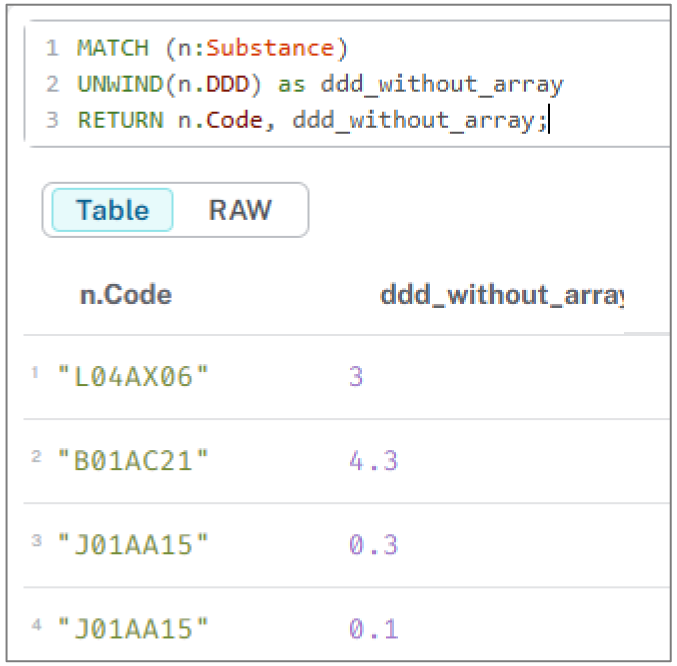
```
nodes2025$ MATCH (n:Substance) RETURN n.Code, n.DDD, n.application
```

n.Code	n.DDD	n.application
"L04AX06"	3	"O"
"B01AC21"	4.3	null
"J01AA15"	0.1	"P"

Attributes with arrays are normal and not the exception (unwind in neo4j is like unnest in PostgreSQL)

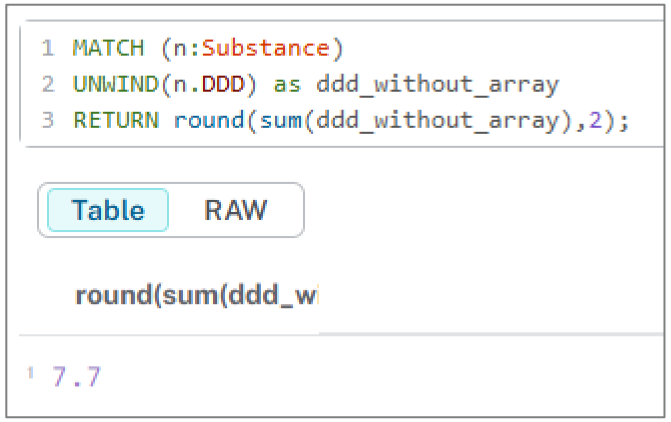


```
MERGE (n:Substance {Code: 'J01AA15'})
  SET n.DDD = [0.3,0.1], n.application = ['O','P'] ;
```



```
MATCH (n:Substance)
UNWIND(n.DDD) as ddd_without_array
RETURN n.Code, ddd_without_array;
```

```
n.Code        ddd_without_arra
"L04AX06"     3
"B01AC21"     4.3
"J01AA15"     0.3
"J01AA15"     0.1
```

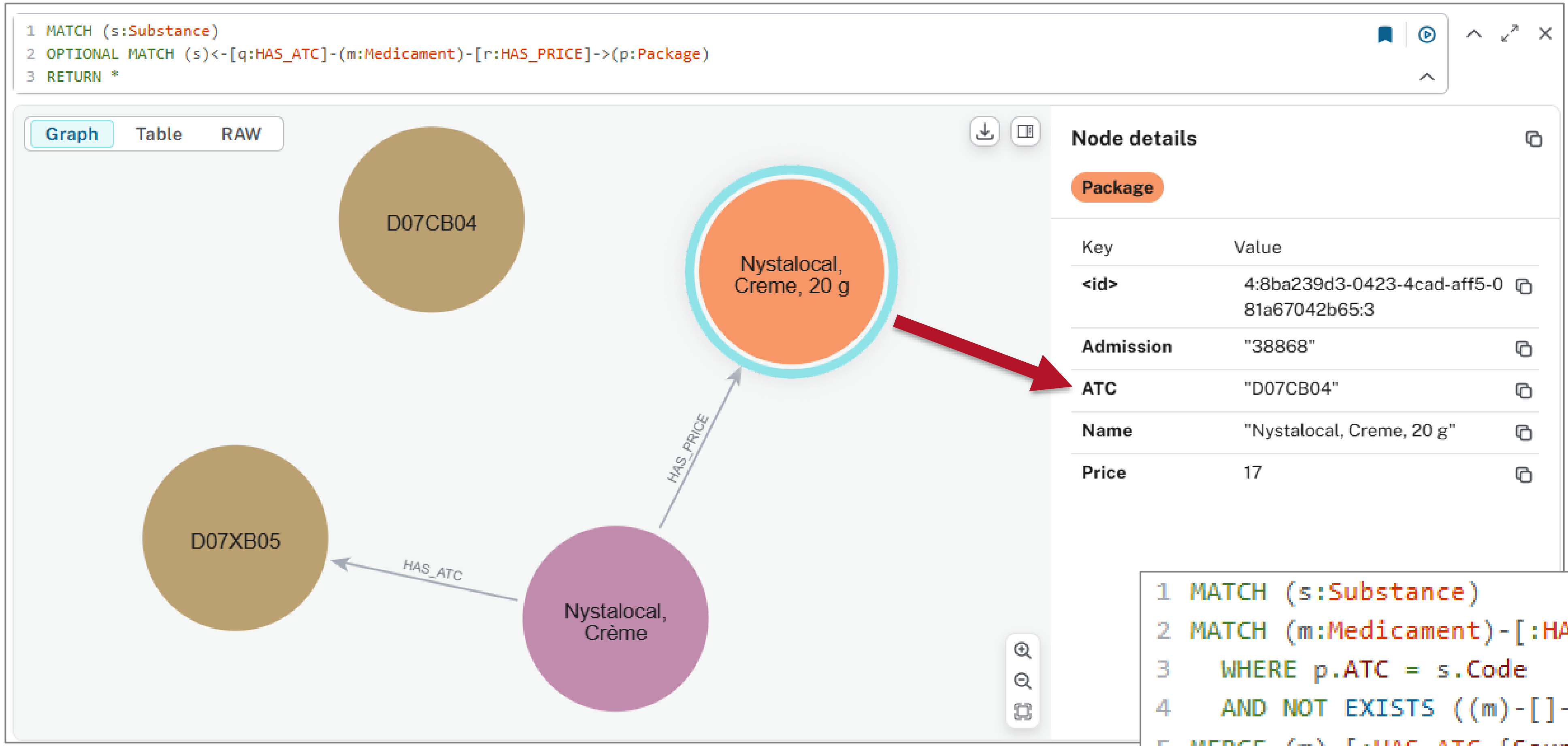


```
MATCH (n:Substance)
UNWIND(n.DDD) as ddd_without_array
RETURN round(sum(ddd_without_array),2);
```

```
round(sum(ddd_w
7.7
```

Example 4

You can add “indirect” relations with information from other nodes in a further modelling phase, if you stored them.

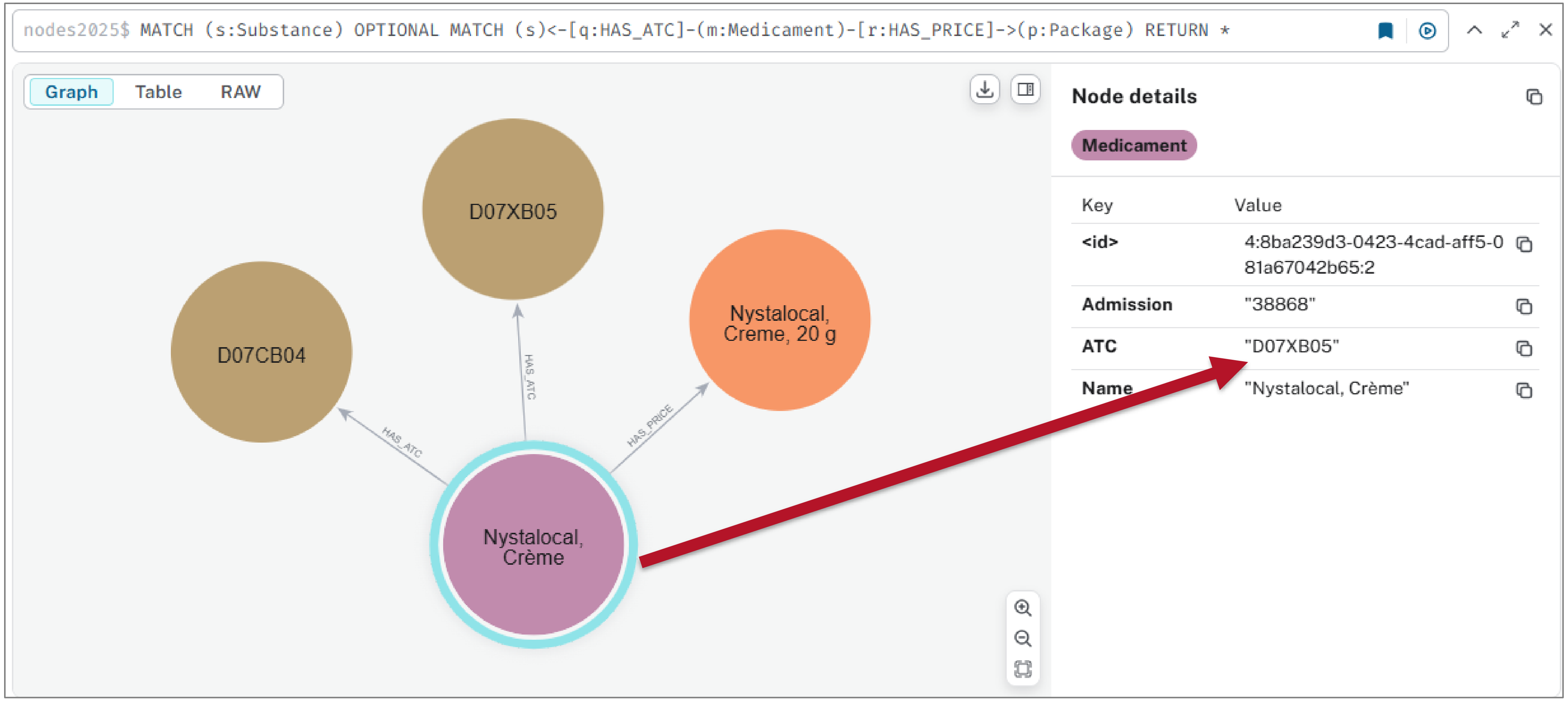


```
MATCH (s:Substance)
OPTIONAL MATCH (s)<-[q:HAS_ATC]-(m:Medicament)-[r:HAS_PRICE]->(p:Package)
RETURN *
```

Key	Value
<id>	4:8ba239d3-0423-4cad-aff5-081a67042b65:3
Admission	"38868"
ATC	"D07CB04"
Name	"Nystalocal, Creme, 20 g"
Price	17

```
MATCH (s:Substance)
MATCH (m:Medicament)-[:HAS_PRICE]->(p:Package)
  WHERE p.ATC = s.Code
  AND NOT EXISTS ((m)-[]->(s))
MERGE (m)-[:HAS_ATC {Source: 'Indirect'}]->(s);
```

With “multiple” relations you enrich your graph and you need fewer complex data cleaning.



```
nodes2025$ MATCH (s:Substance) OPTIONAL MATCH (s)<-[q:HAS_ATC]-(m:Medicament)-[r:HAS_PRICE]->(p:Package) RETURN *
```

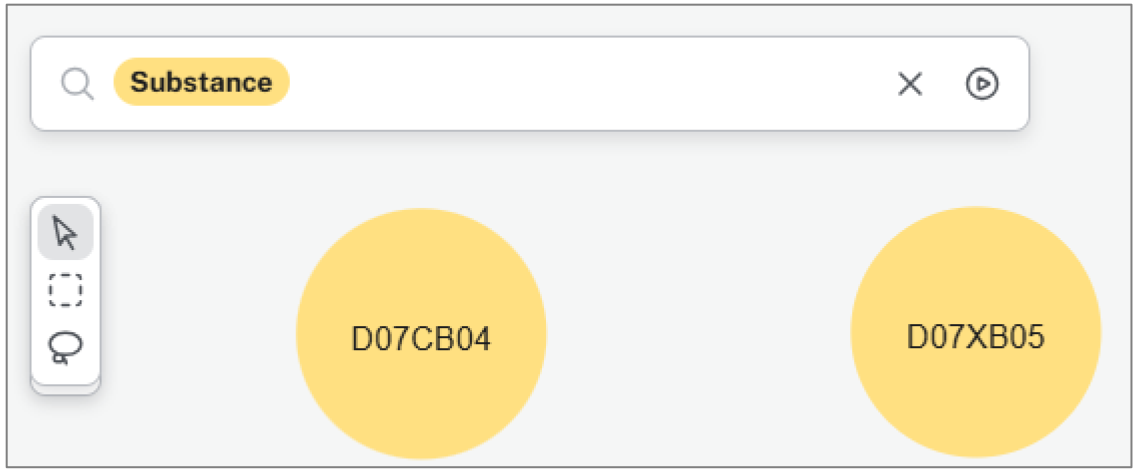
Node details

Medicament

Key	Value
<id>	4:8ba239d3-0423-4cad-aff5-081a67042b65:2
Admission	"38868"
ATC	"D07XB05"
Name	"Nystalocal, Crème"

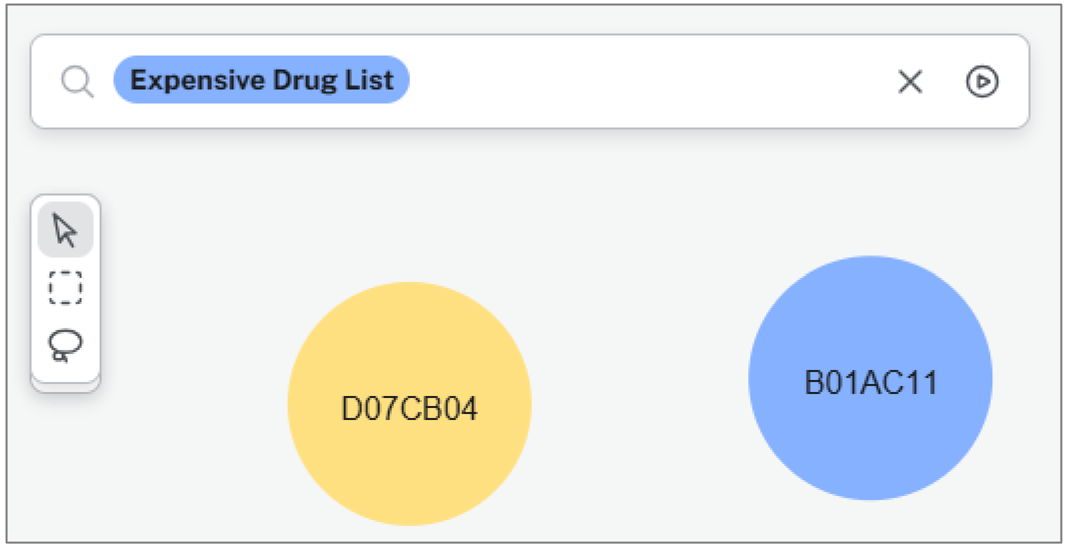
Example 5

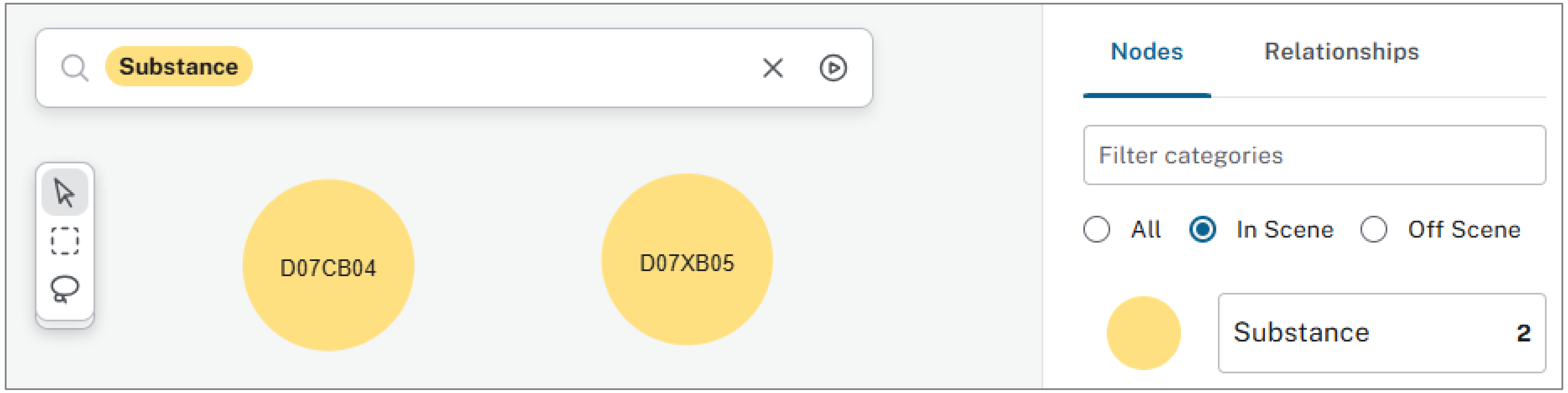
Now we use Explore/Bloom with same substance nodes like before.



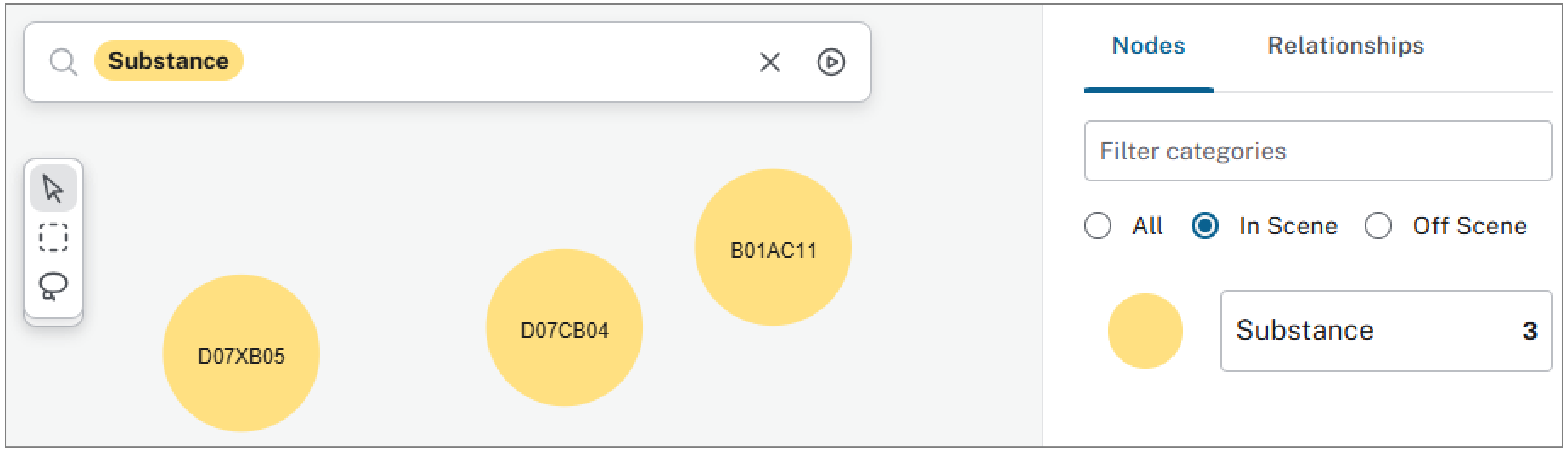
We introduce a new label.

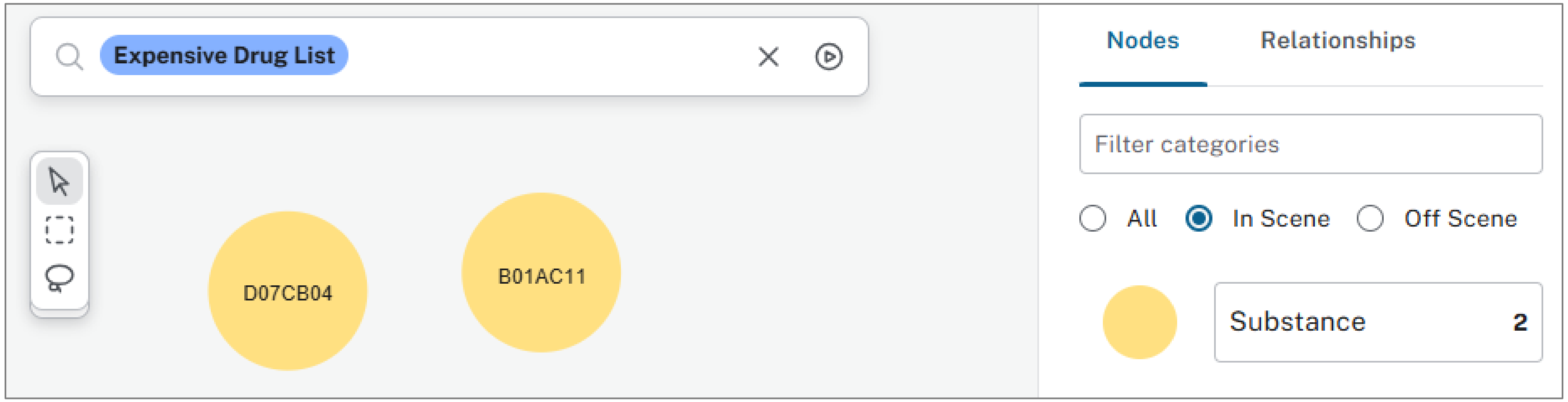
```
// Create new label
MERGE (n:Substance {Code: 'D07CB04'})
SET n:`Expensive Drug List`;

// Create new node with new label
MERGE (n:`Expensive Drug List` {Code: 'B01AC11'})
SET n.Description = 'Iloprost';
```





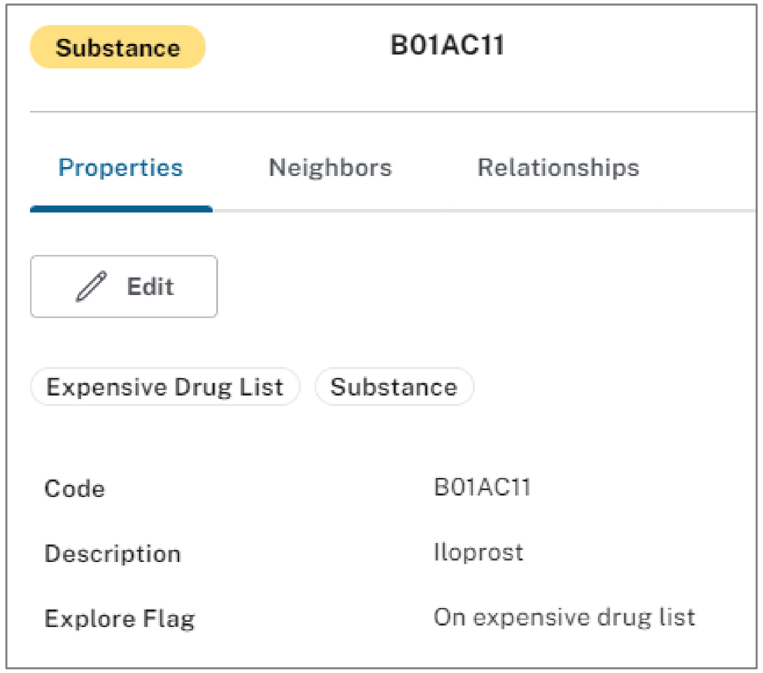
```
// Set new node with "old" label
MATCH (n:`Expensive Drug List`) WHERE n.Code = 'B01AC11'
SET n:Substance;
```





After setting an attribute, you can use rule-based colors.

```
MATCH (n:`Expensive Drug List`)
SET n.`Explore Flag` = 'On expensive drug list';
```



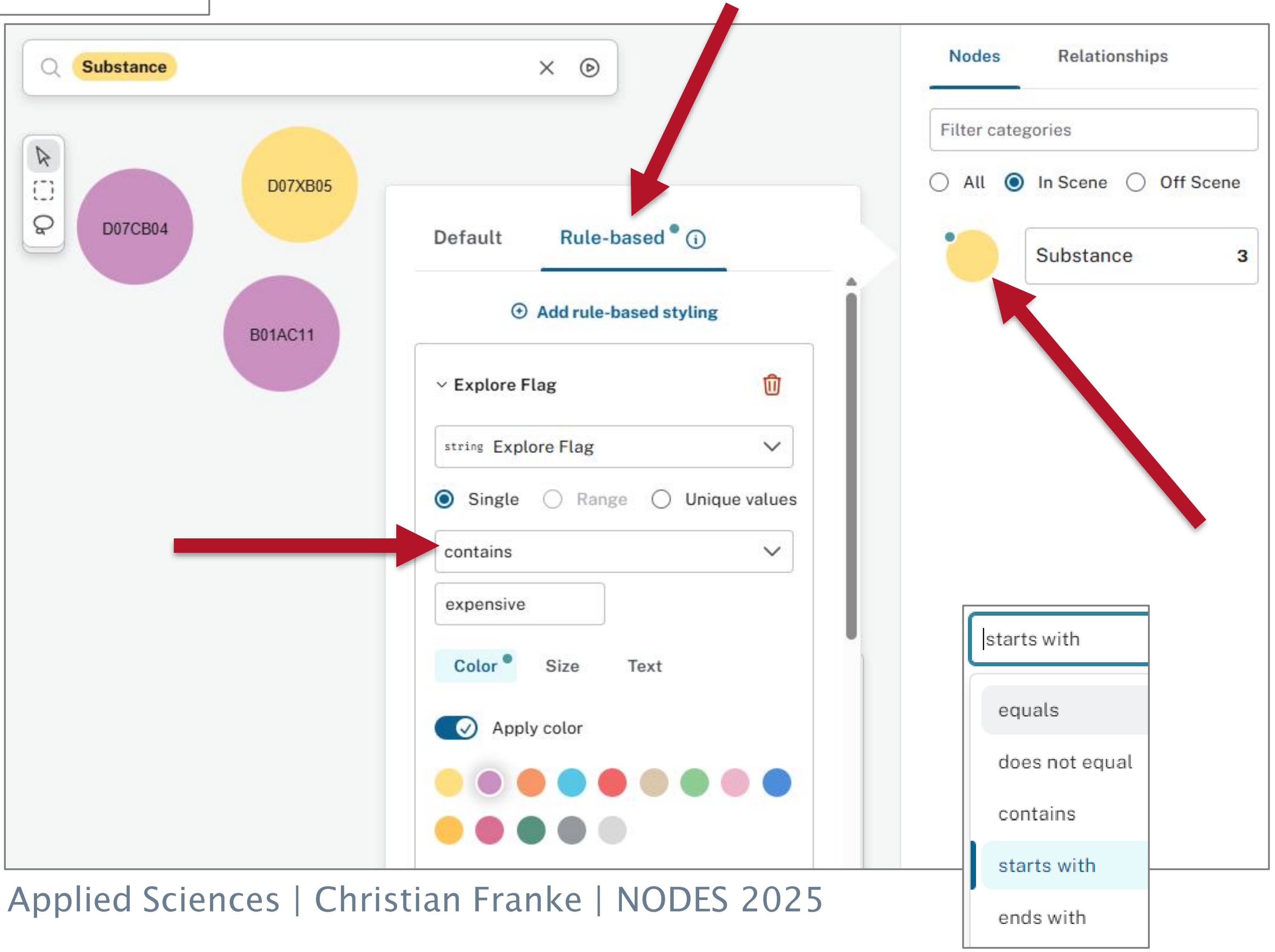
Substance B01AC11

Properties | Neighbors | Relationships

Edit

Expensive Drug List | Substance

Code	B01AC11
Description	Iloprost
Explore Flag	On expensive drug list



Default | Rule-based

Add rule-based styling

Explore Flag

string Explore Flag

[✓ Single] Range Unique values

contains

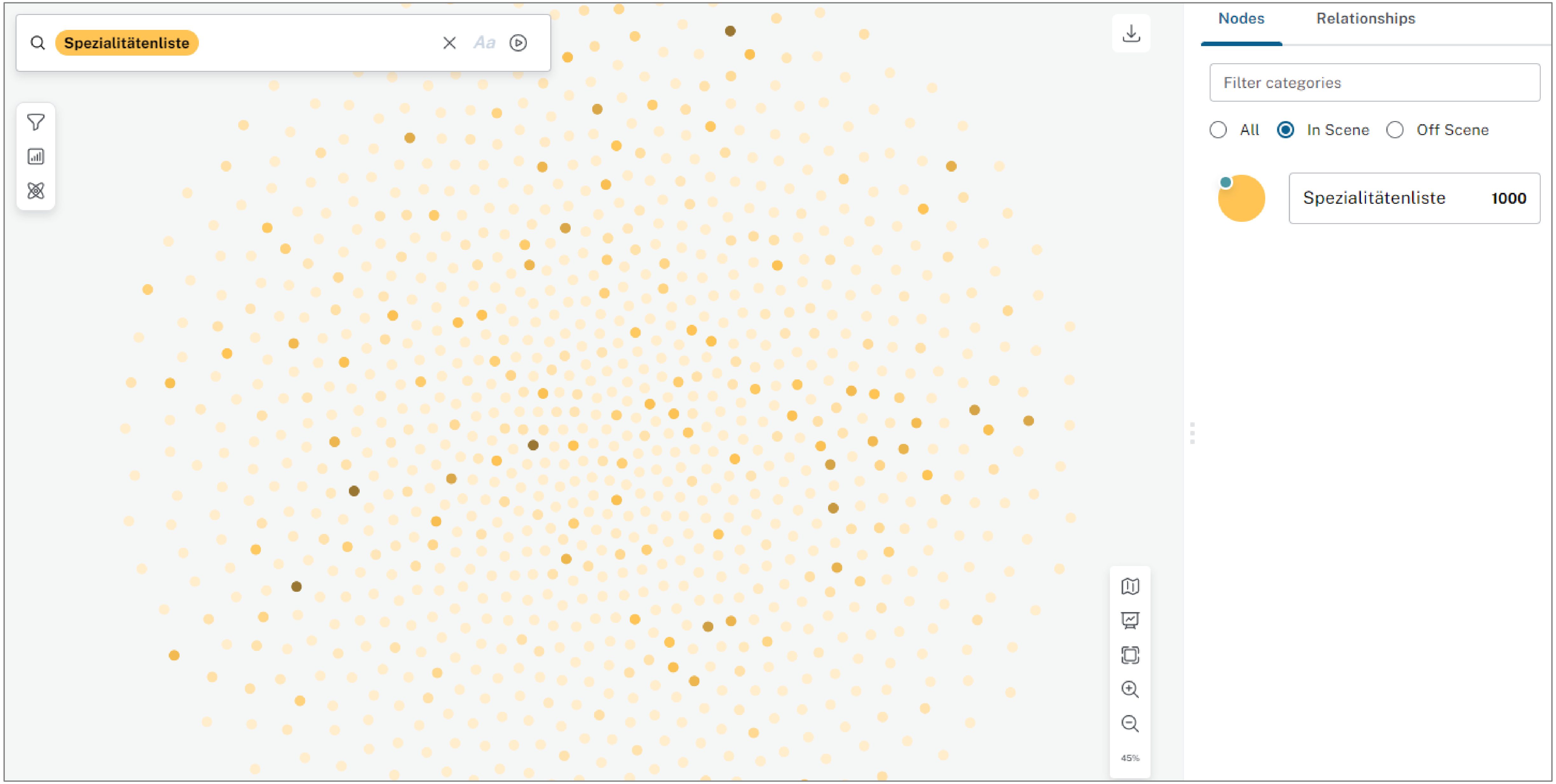
expensive

Color | Size | Text

Apply color ✓

starts with

equals
does not equal
contains
starts with
ends with



Spezialitätenliste

Nodes

Relationships

Filter categories

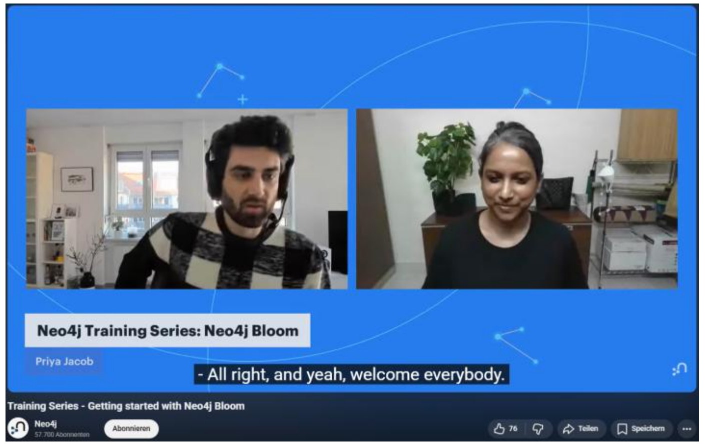
All [✓ In Scene] Off Scene

Spezialitätenliste	1000

Example 6

If perspectives have version numbers, you probably must change deeplinks in other applications.

```
https://console-preview.neo4j.io/tools/explore?
search=Substance%20informations%20for%20L01FY01&
perspective=Medi-Graph%20V1.0&
run=true
```


Neo4j Training Series: Neo4j Bloom
Priya Jacob
Training Series - Getting started with Neo4j Bloom
Neo4j
Abonnieren
Teilen
Speichern

- All right, and yeah, welcome everybody.

Training Series – Getting started with Neo4j Bloom
https://www.youtube.com/live/7yS2e4p0_H4

Summary

Summary

- Be careful with transferring “hierarchy” or “dimension tables” from RDBMS to graphs: probably better use attributes and not lables
- In graphs you can create edges instead of n:m tables
- Arrays are normal in graphs and not the exception (“forget” 3NF or BCNF)
- MERGE … SET can be UPDATE or INSERT in SQL
- Flexible relationships in graphs are useful, especially if there is “tidy” data in your source
- Store important codes as attribute in the nodes, so that you can “recycle” them
- Bloom/Explore offers a lot of individual solutions with saved cyphers, colors, symbols, sizes etc.
- Don’t use version numbers for important perspectives in Bloom/Explore

Contact and further information

linkedin.com/in/christian-franke-b7b7a238b/

github.com/teletrabbie/nodes2025

christianfranke.quarto.pub