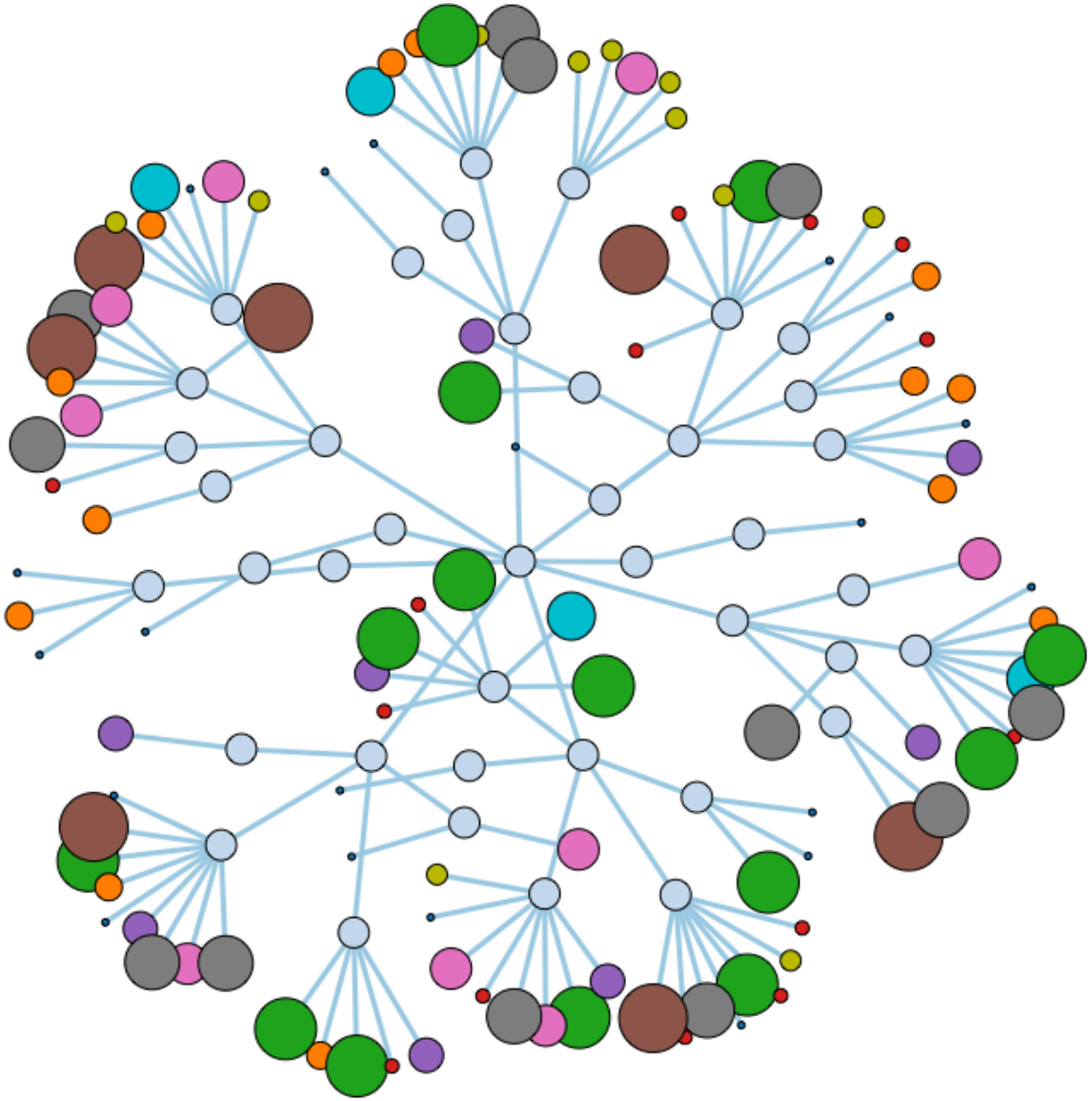
Thinking in Graphs

@iansrobinson
ian.robinson@neotechnology.com

#neo4j

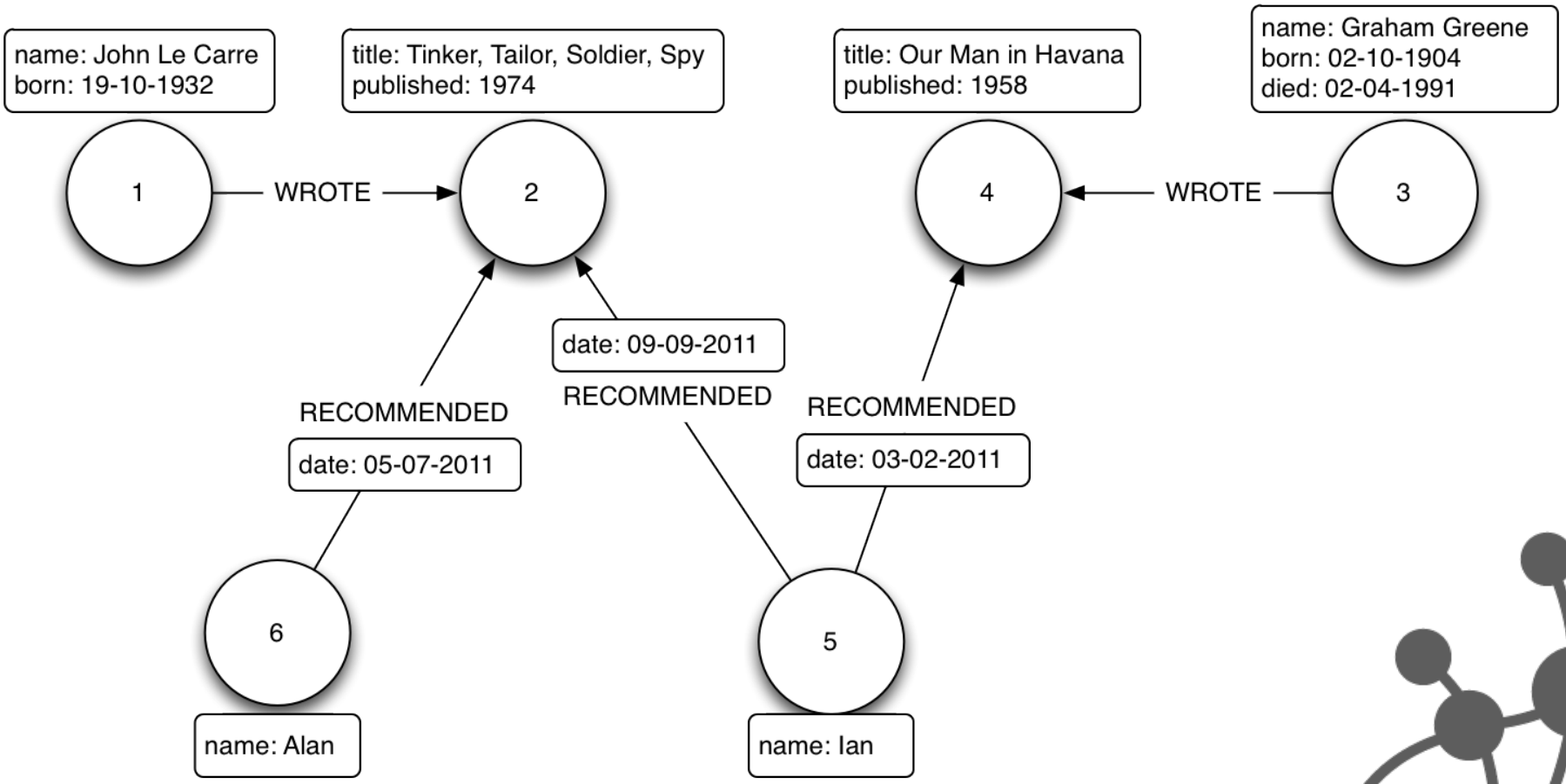
Data Complexity

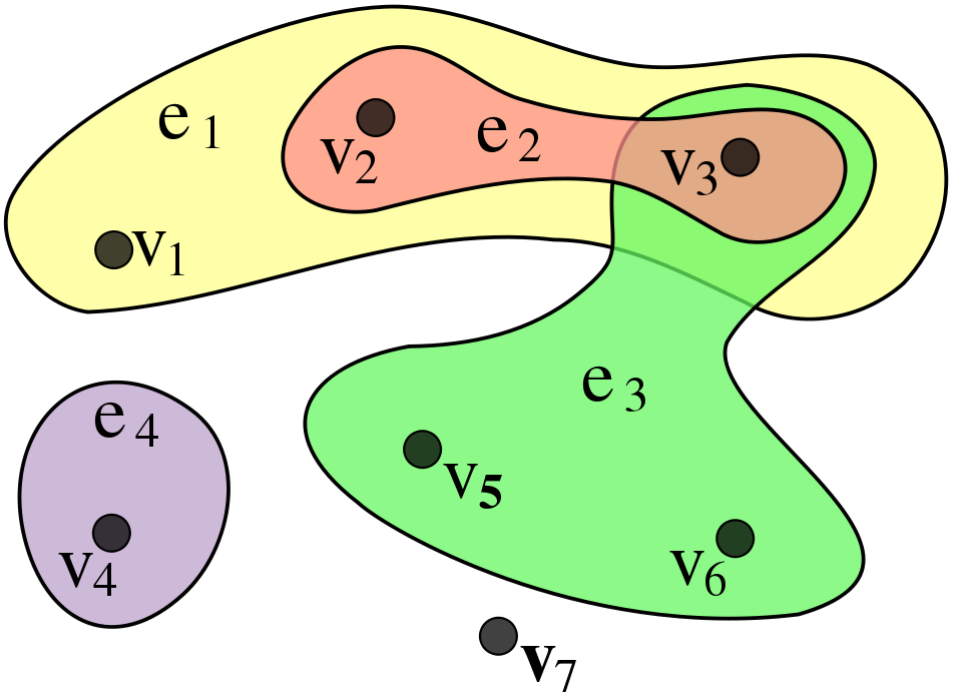
complexity = f(size, connectedness, semi-structure)

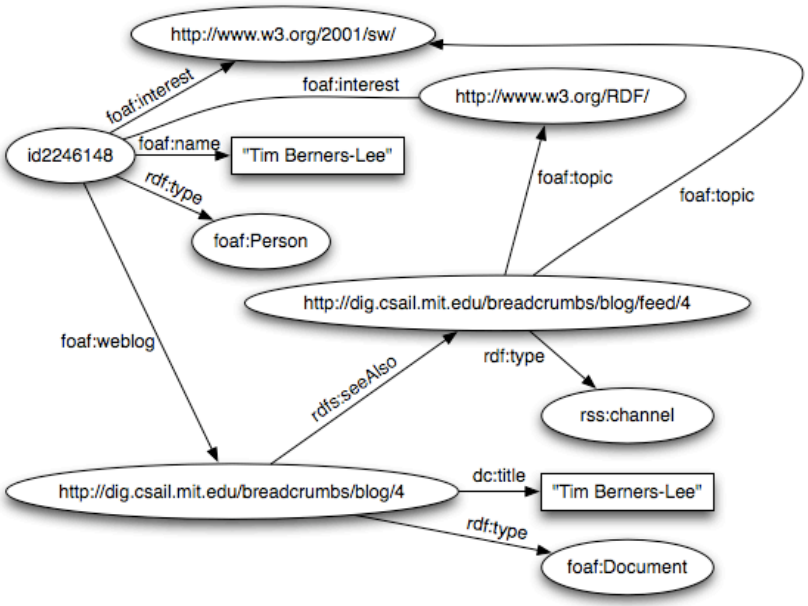


#neo4j

Property Graph Data Model

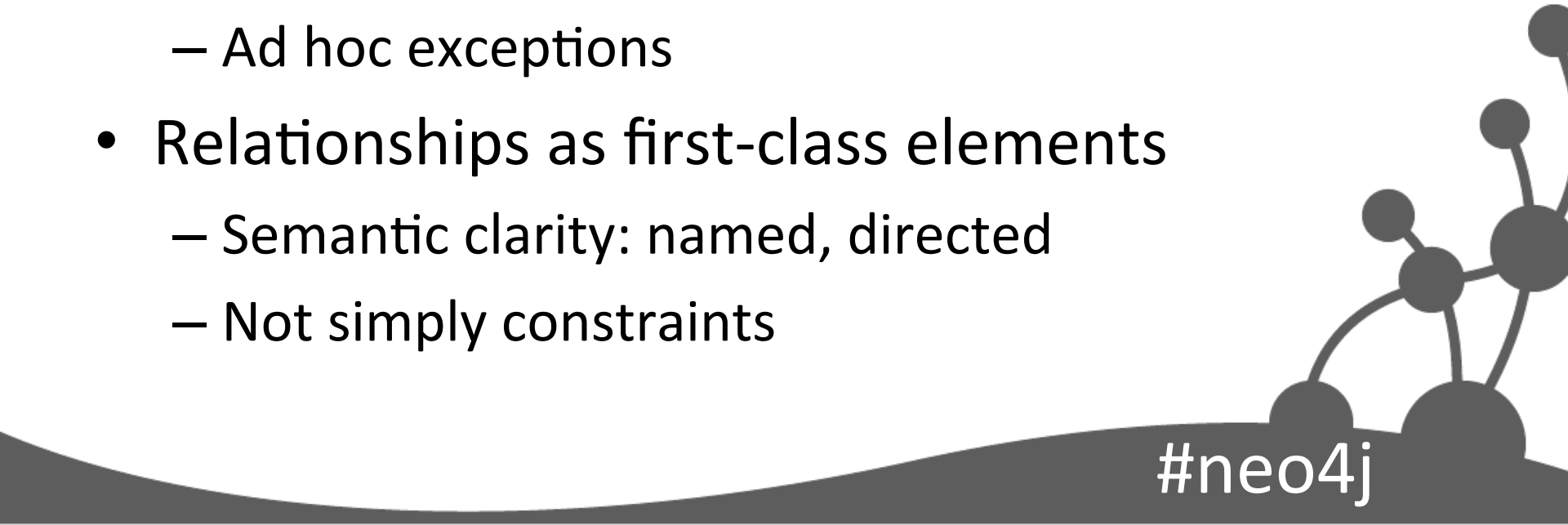



e 1
v2
e 2
v3
v1
e 3
e 4
v5
v4
v6
v7

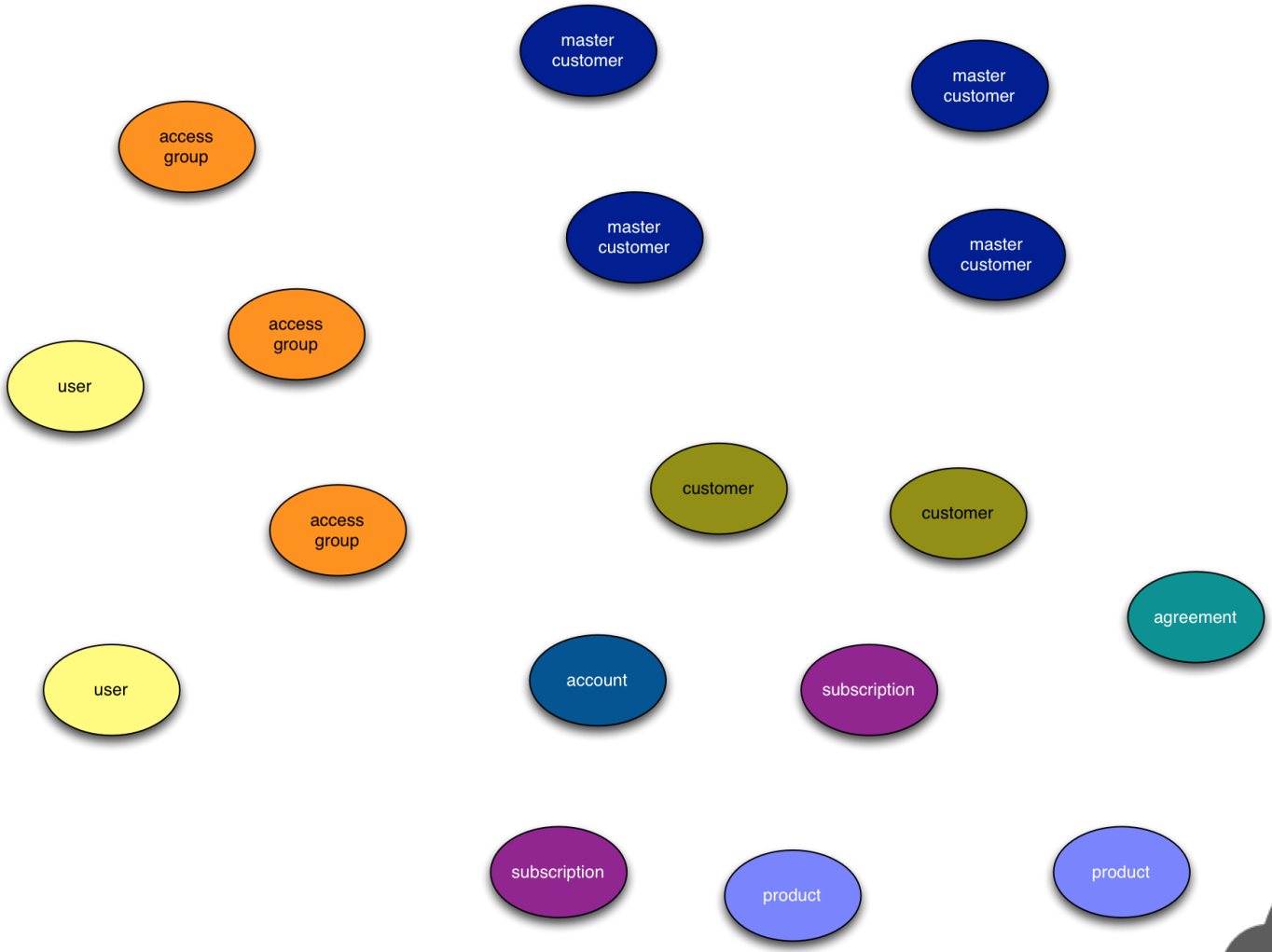

http://www.w3.org/2001/sw/
foaf:interest
foaf:interest
http://www.w3.org/RDF/
id2246148
foaf:name
"Tim Berners-Lee"
rdf:type
foaf:topic
foaf:topic
foaf:Person
http://dig.csail.mit.edu/breadcrumbs/blog/feed/4
foaf:weblog
rdf:type
rdfs:seeAlso
rss:channel
dc:title
http://dig.csail.mit.edu/breadcrumbs/blog/4
"Tim Berners-Lee"
rdf:type
foaf:Document

Expressive Graph Data Model

- Complex, densely-connected domains
  - Lots of join tables? Relationships
  - Lots of sparse tables? Semi-structure
- Messy data
  - Ad hoc exceptions
- Relationships as first-class elements
  - Semantic clarity: named, directed
  - Not simply constraints

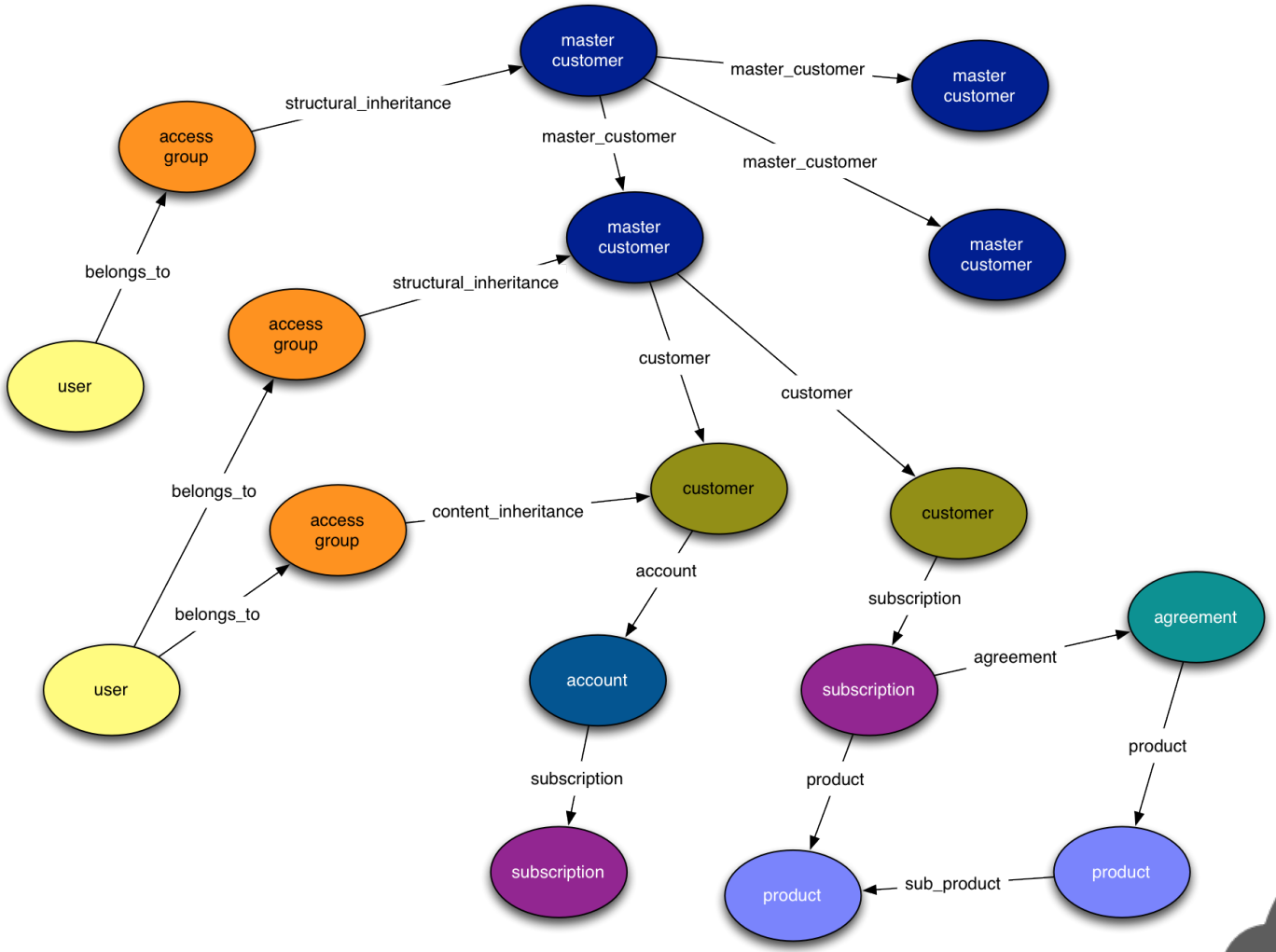


Schema-Free…



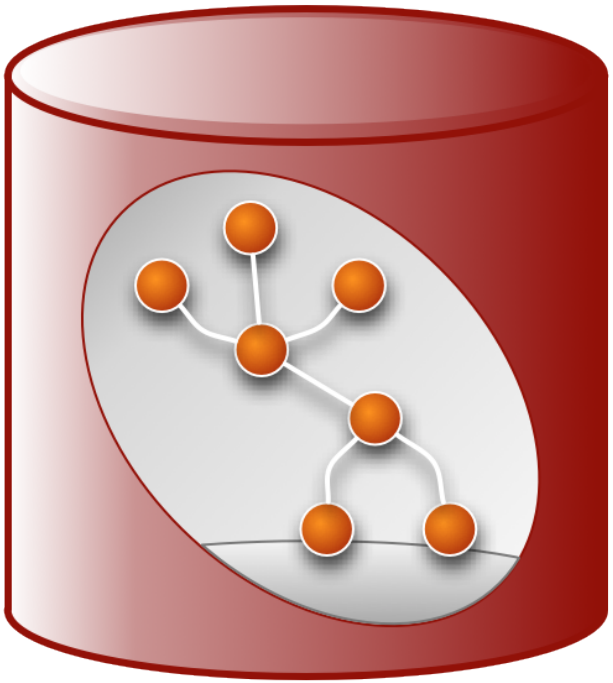
#neo4j

But Structured

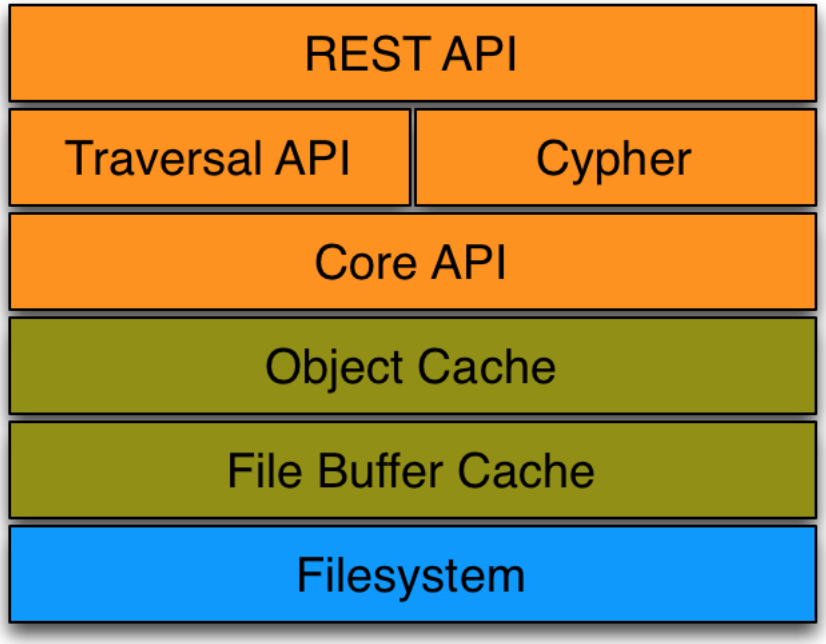


#neo4j

Neo4j

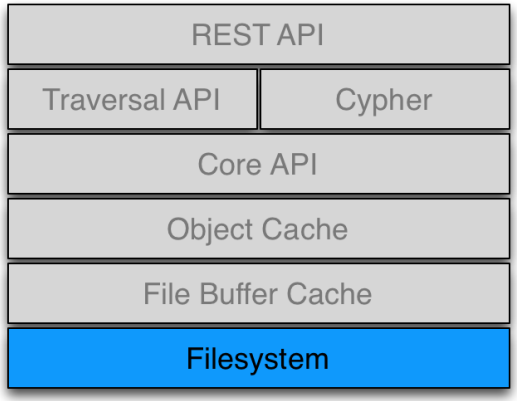


Neo4j Architecture



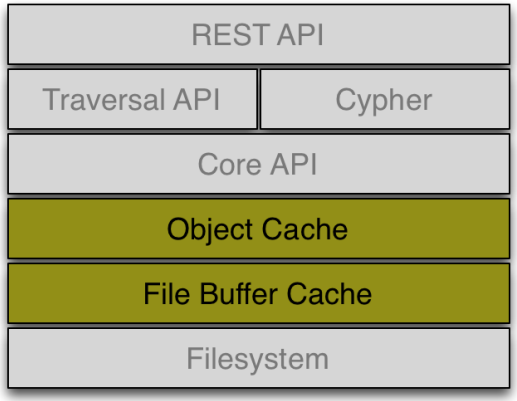
Filesystem

- Neo4j uses several stores:
  - Node
  - Relationship
  - Property
    - including short strings and small arrays
  - String
    - for long strings
  - Array
    - for large arrays



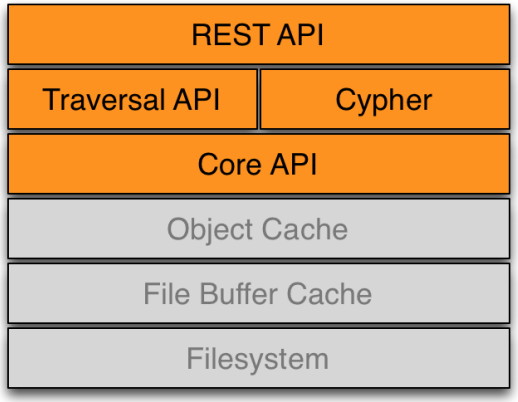
Caches

- File Buffer Cache:
  - Memory-mapped NIO
- Object Cache:
  - Nodes
  - Relationships

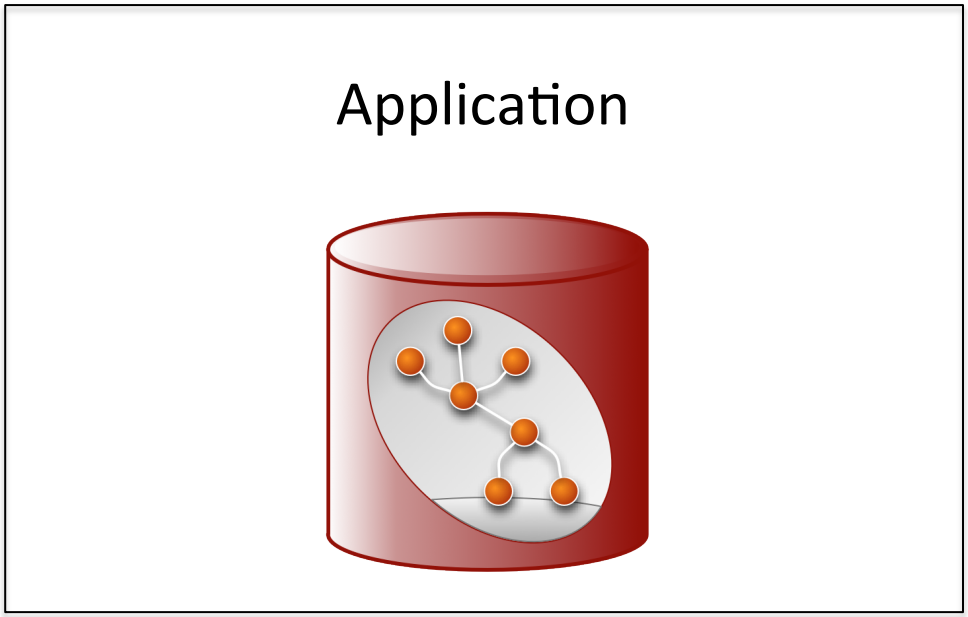


APIs

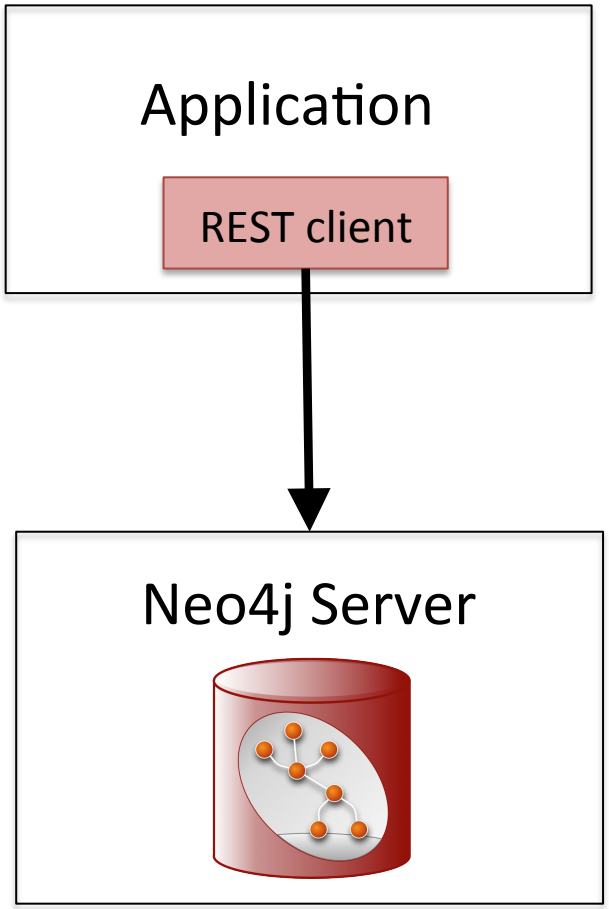
- Core API:
  - Nodes
  - Relationships
- Traversal API:
  - Lazily spider the graph
- Cypher
  - Graph pattern matching
- REST
  - JSON over HTTP



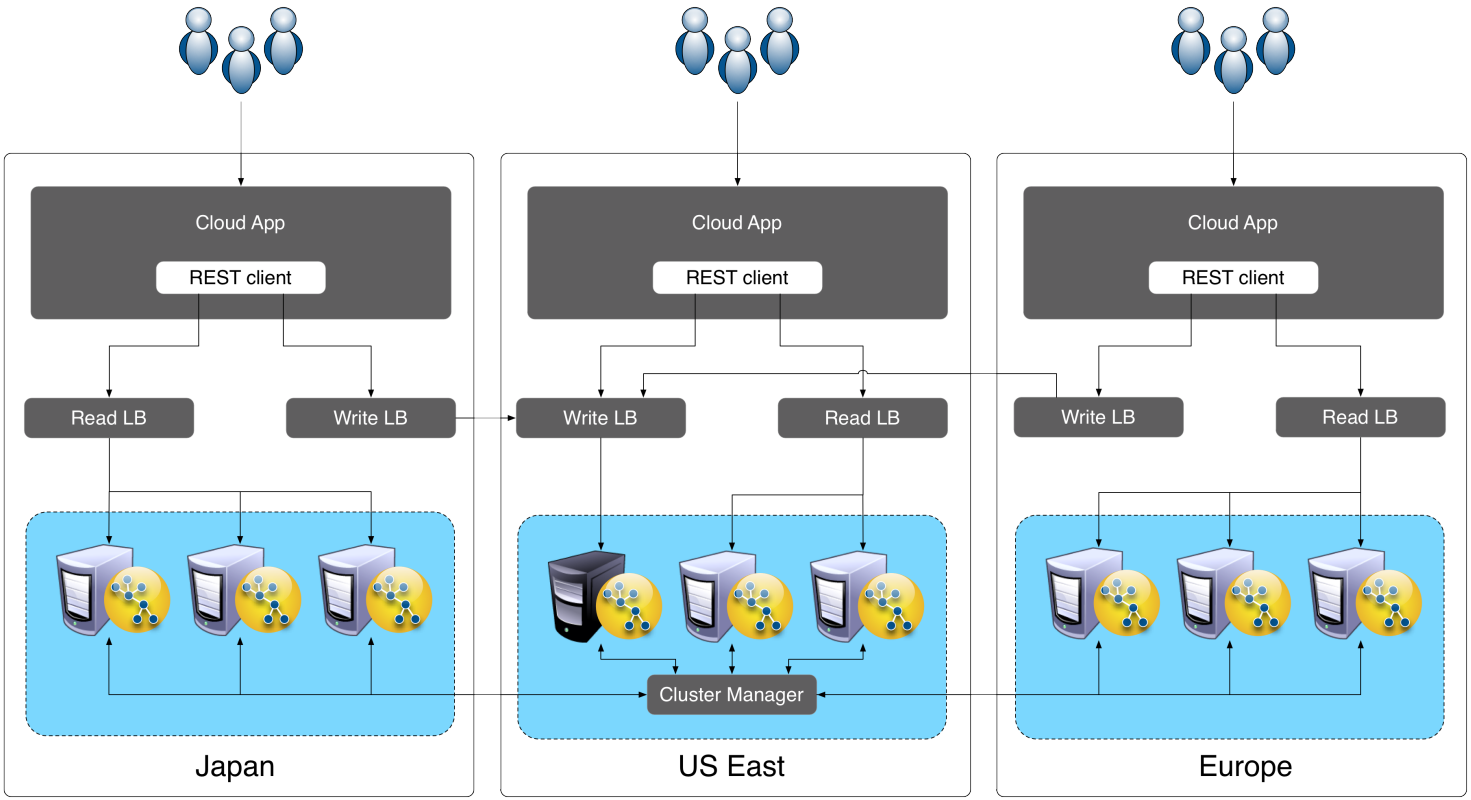
Embedded



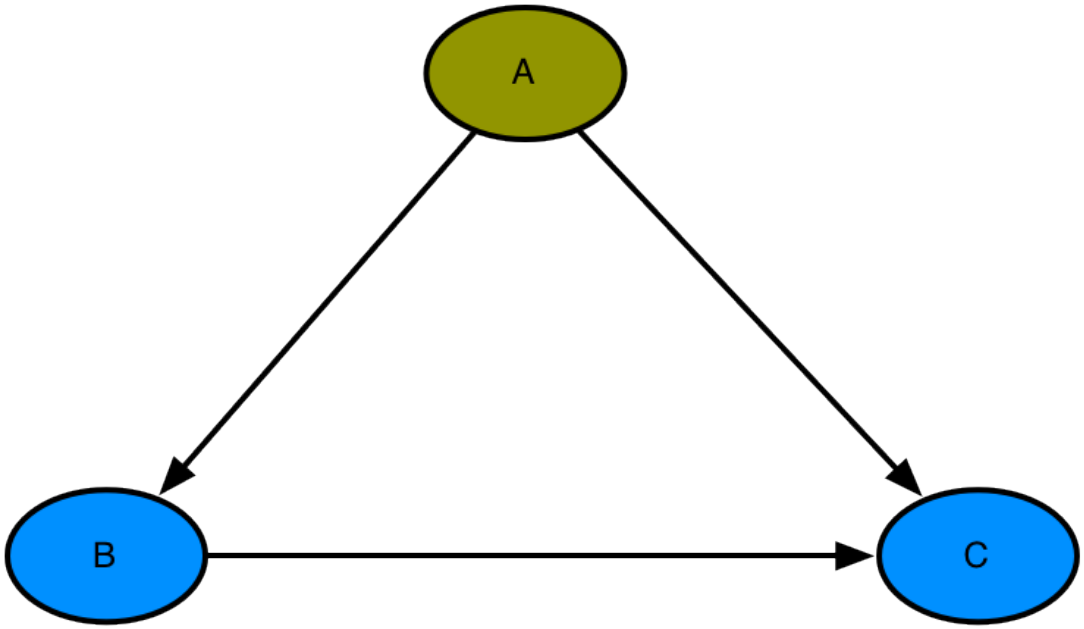
Server



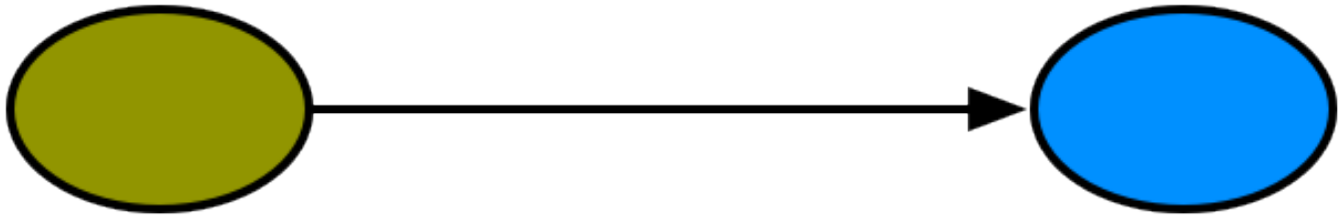
High Availability



Cypher – A Graph Language



Anonymous Nodes & Rels

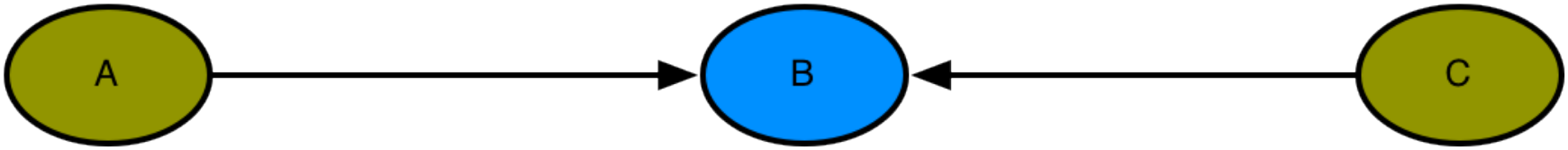


```
()-->()
```

ASCII Art Patterns

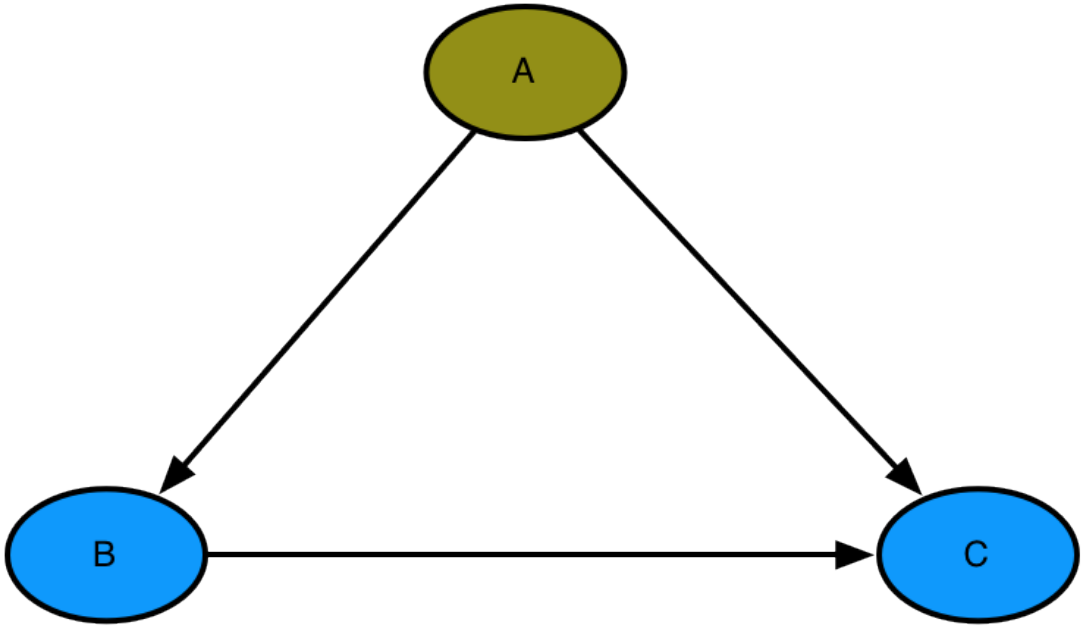

```
A-[:CONNECTED_TO]->B
```

ASCII Art Patterns



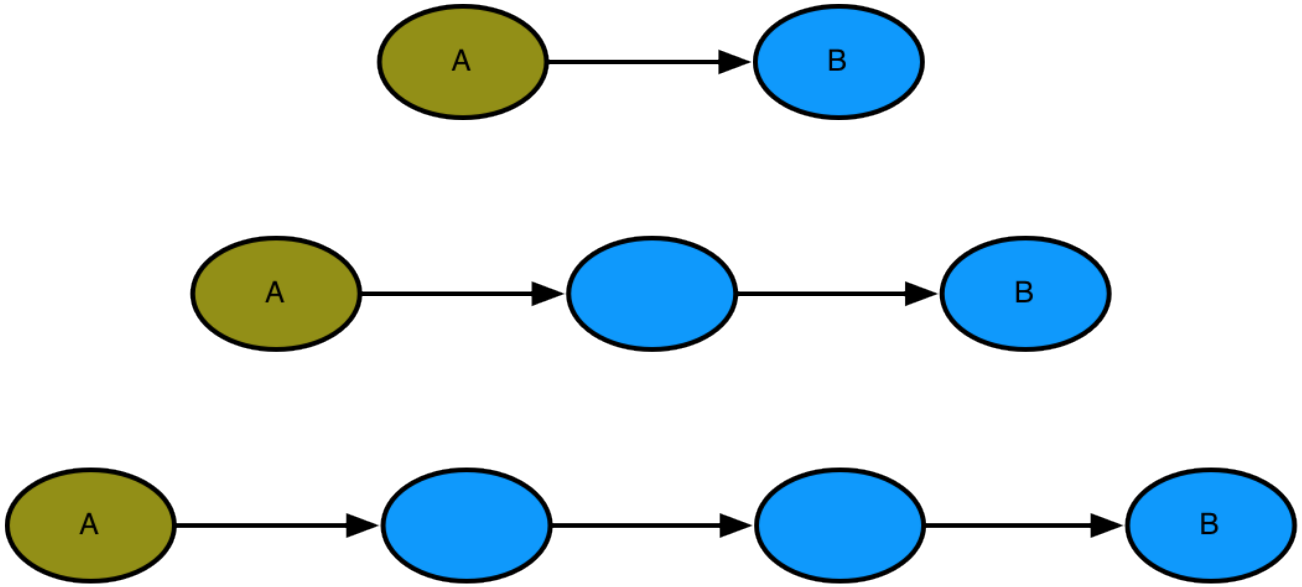
A--->B<---C

ASCII Art Patterns




```
A-->B-->C, A-->C
A-->B-->C<--A
```

Variable Length Paths



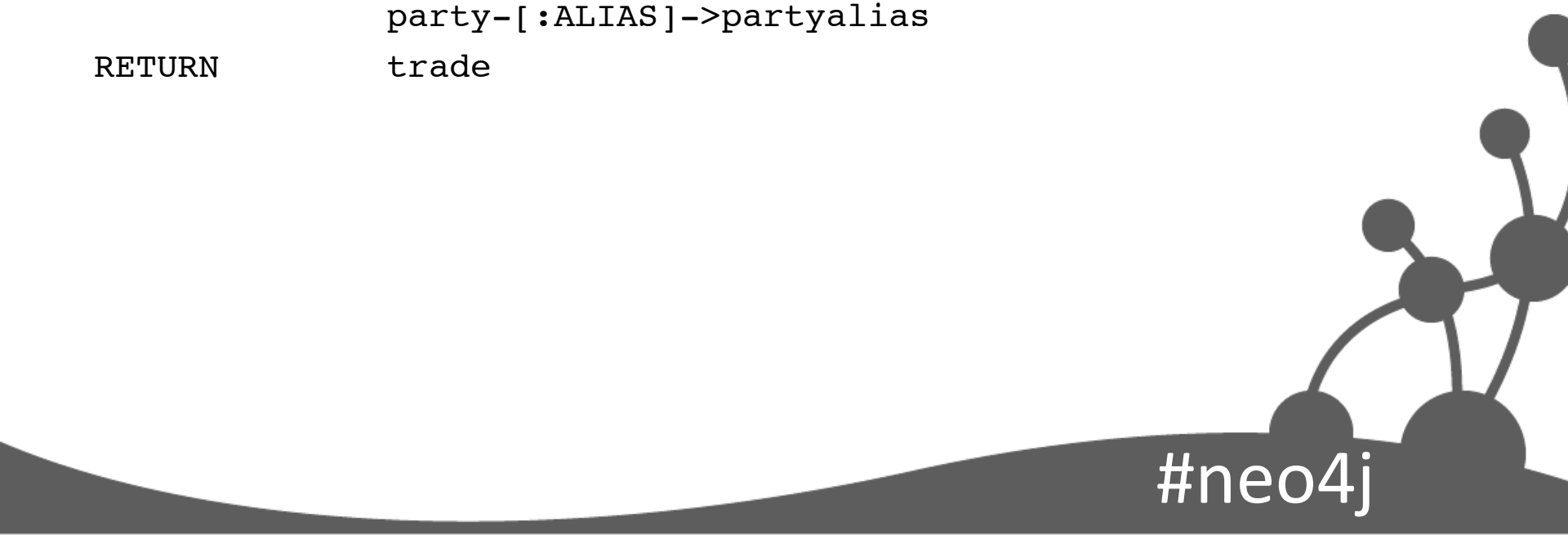
A-[*]->B

Create Some Data

```
CREATE
  (cc {name:'Cost Centre 1'}),
  (lb {name:'Ledger Book 1'}),
  cc-[:HAS_LEDGER_BOOK]->lb
```

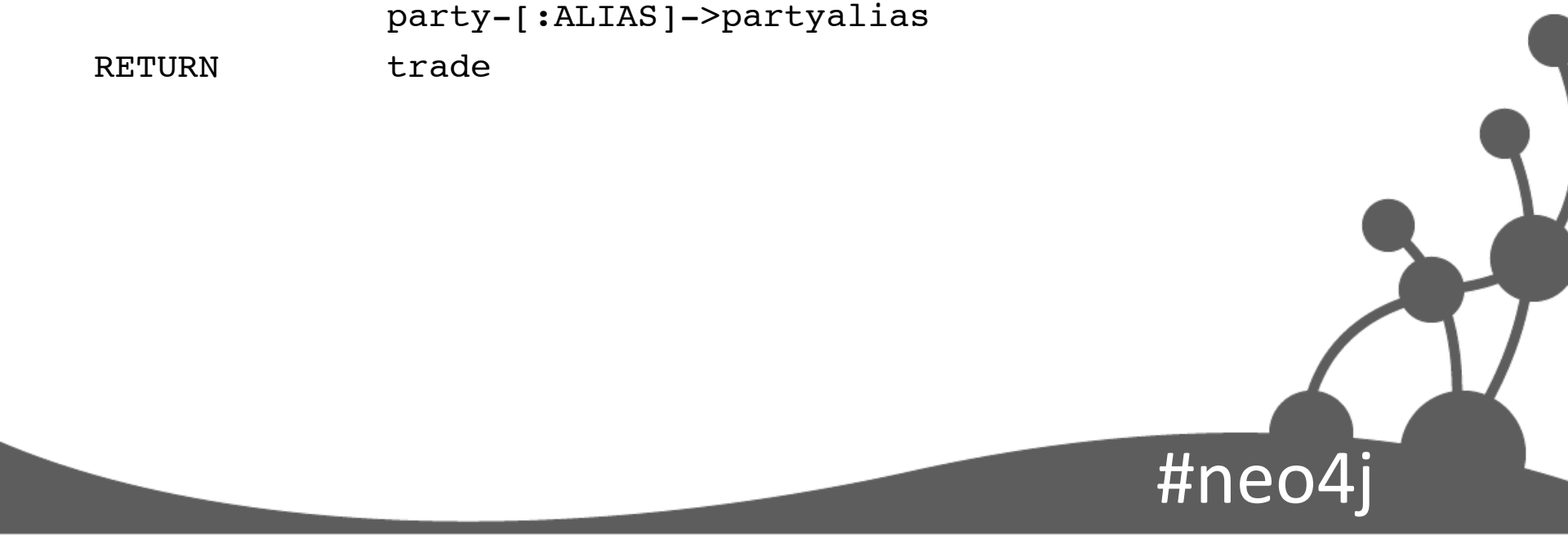
Add Data

```
START          party=node:party(name='Party 1'),
               sourcebook=node:source_book(name='Source Book 1')
CREATE UNIQUE  sourcebook-[:HAS_TRADE]->(trade {id:'0001'}),
               trade-[:PARTY_ALIAS]->
                   (partyalias {name:'Party Alias 1'}),
               trade-[:CURRENCY]->
                   (currency {name:'GBP', _label:'currency'}),
               party-[:ALIAS]->partyalias
RETURN         trade
```



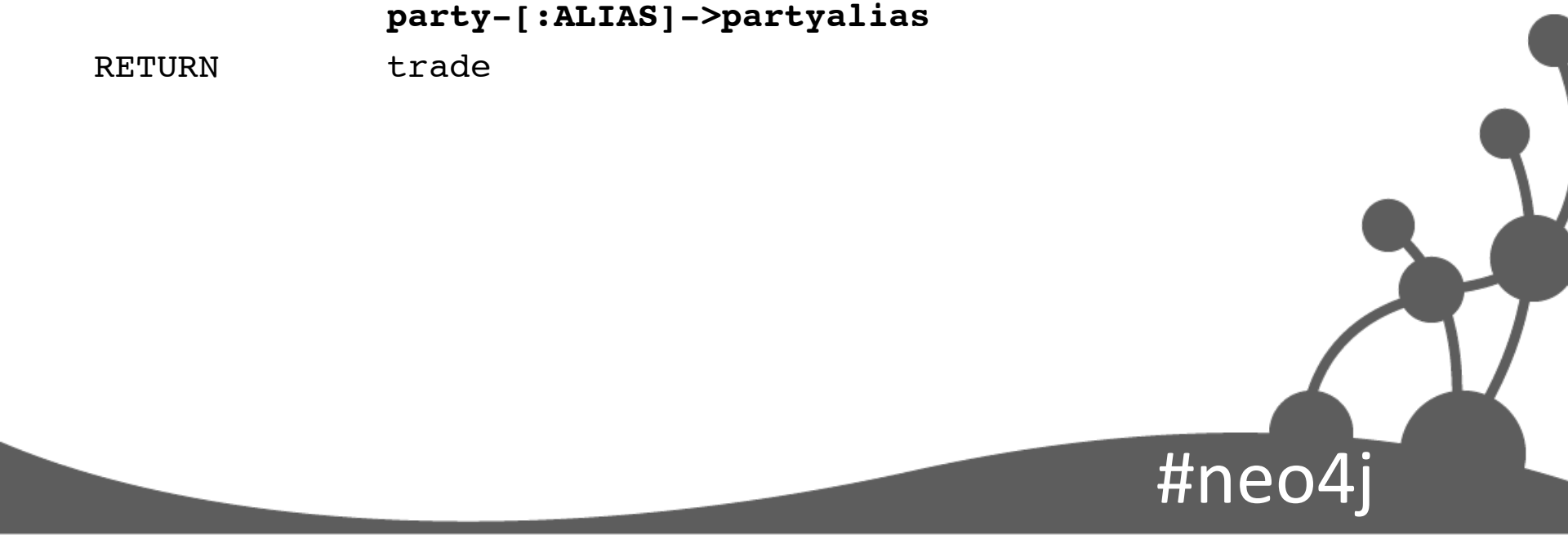
Find Start Nodes in Indexes

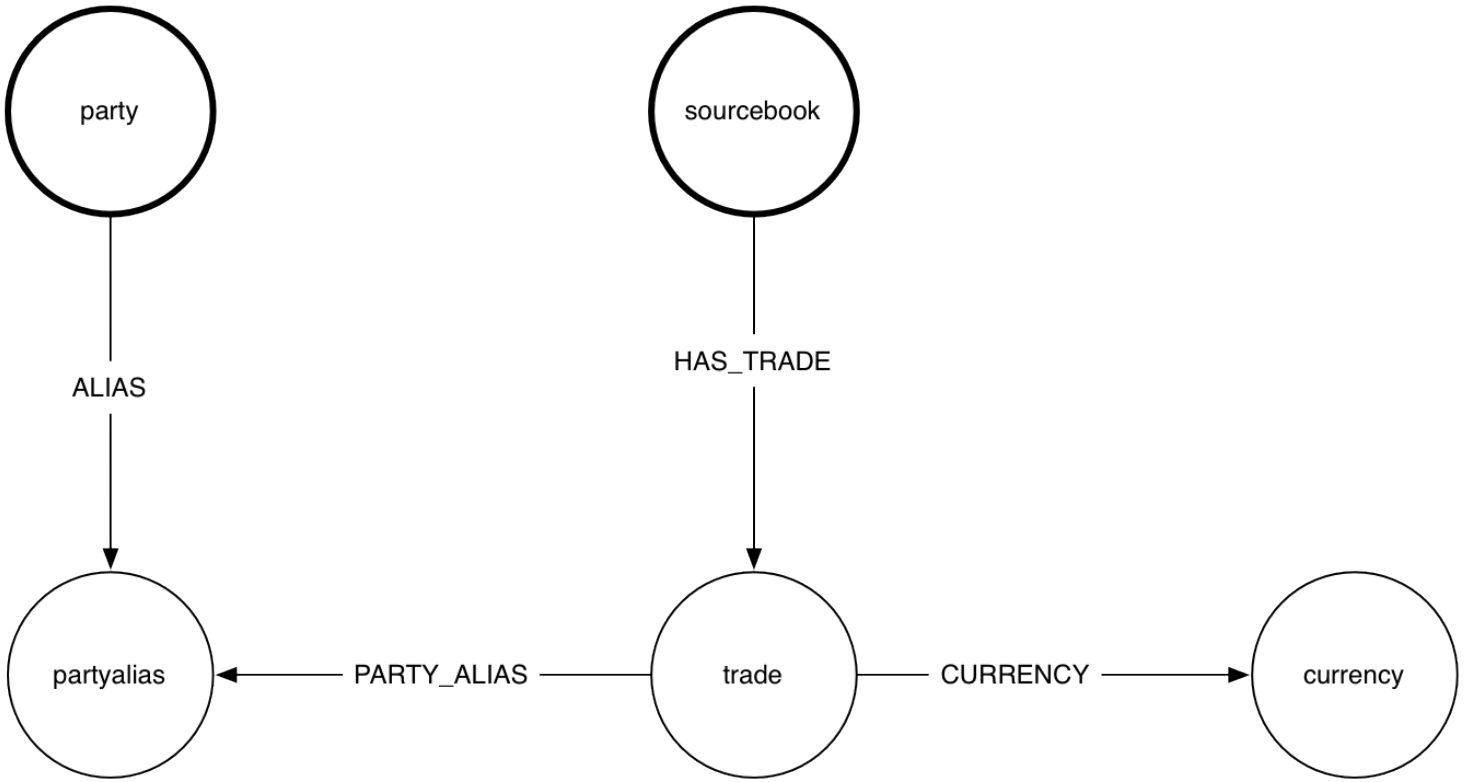
```
START          party=node:party(name='Party 1'),
               sourcebook=node:source_book(name='Source Book 1')
CREATE UNIQUE  sourcebook-[:HAS_TRADE]->(trade {id:'0001'}),
               trade-[:PARTY_ALIAS]->
                   (partyalias {name:'Party Alias 1'}),
               trade-[:CURRENCY]->
                   (currency {name:'GBP', _label:'currency'}),
               party-[:ALIAS]->partyalias
RETURN         trade
```



Describe New Data

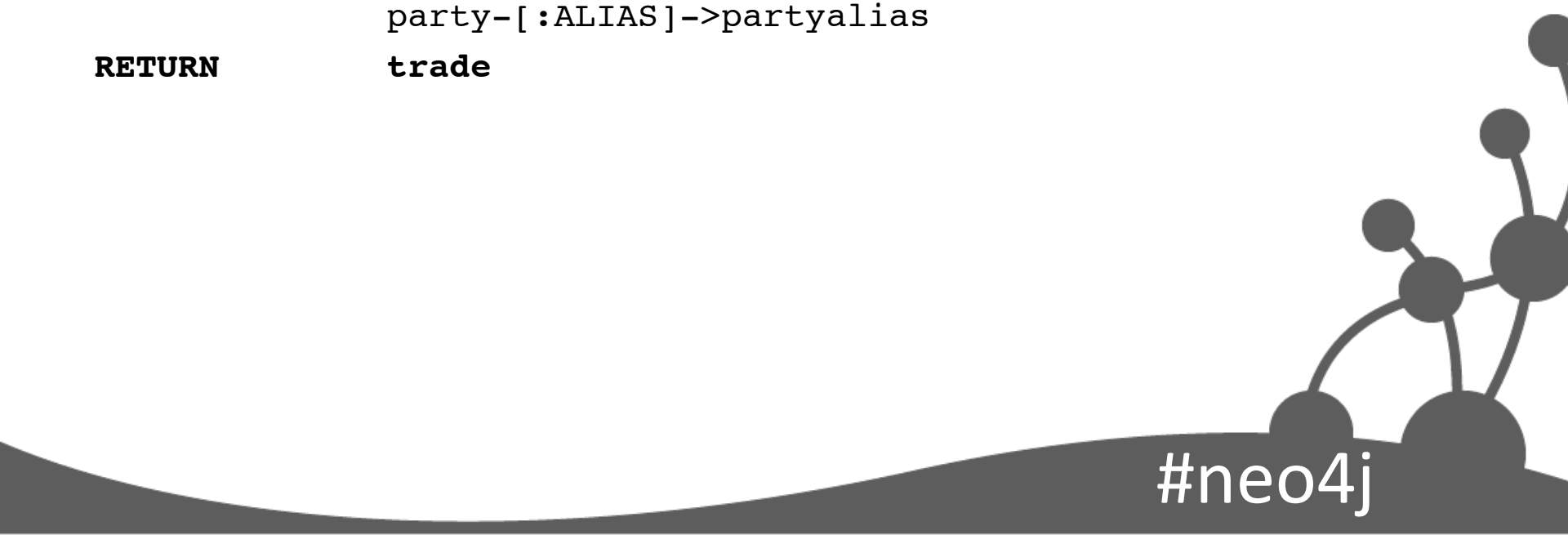
```
START          party=node:party(name='Party 1'),
               sourcebook=node:source_book(name='Source Book 1')
CREATE UNIQUE  sourcebook-[:HAS_TRADE]->(trade {id:'0001'}),
               trade-[:PARTY_ALIAS]->
                   (partyalias {name:'Party Alias 1'}),
               trade-[:CURRENCY]->
                   (currency {name:'GBP', _label:'currency'}),
               party-[:ALIAS]->partyalias
RETURN         trade
```




party
sourcebook
ALIAS
HAS_TRADE
partyalias
PARTY_ALIAS
trade
CURRENCY
currency

Return New Node

```
START          party=node:party(name='Party 1'),
               sourcebook=node:source_book(name='Source Book 1')
CREATE UNIQUE  sourcebook-[:HAS_TRADE]->(trade {id:'0001'}),
               trade-[:PARTY_ALIAS]->
                   (partyalias {name:'Party Alias 1'}),
               trade-[:CURRENCY]->
                   (currency {name:'GBP', _label:'currency'}),
               party-[:ALIAS]->partyalias
RETURN         trade
```

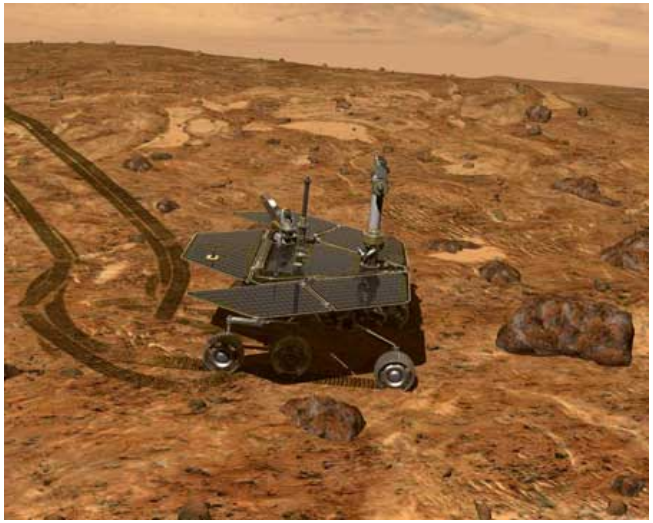


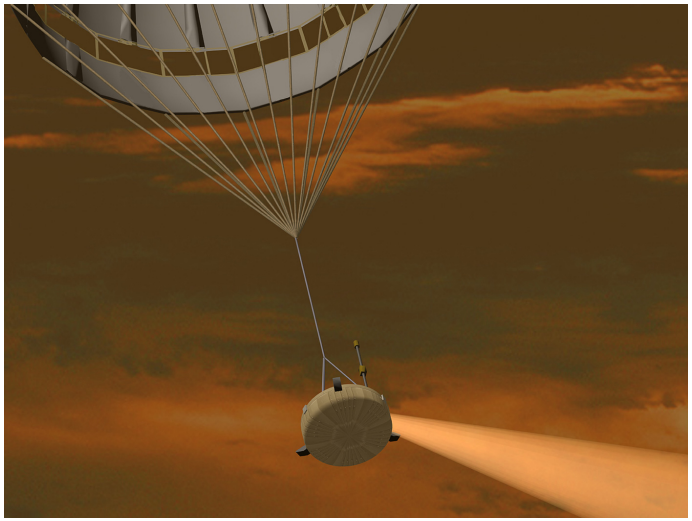
#neo4j

Querying the Graph

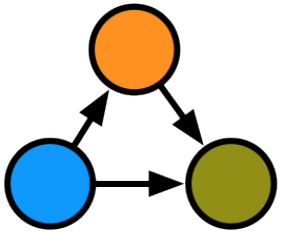
Graph local:

- One or more start nodes
- Explore surrounding graph
- Millions of joins per second

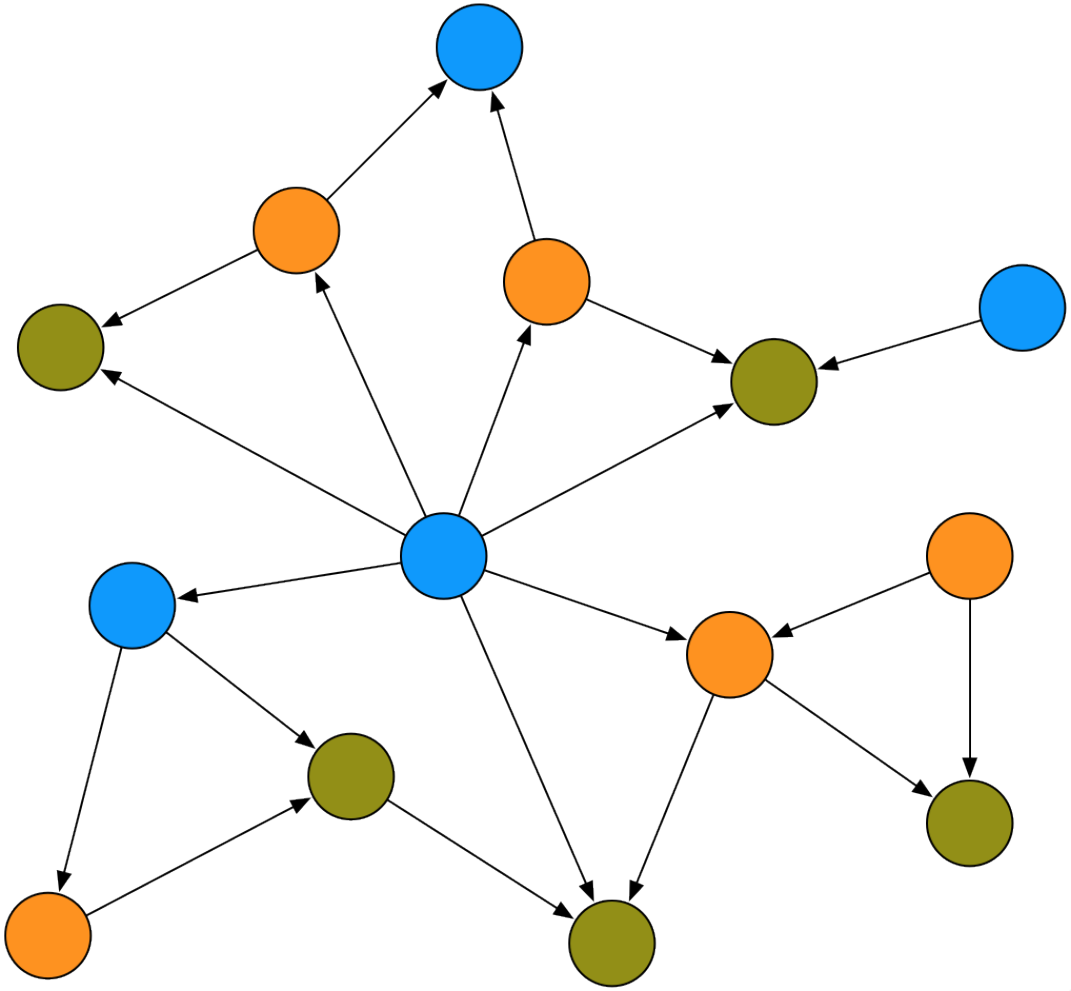




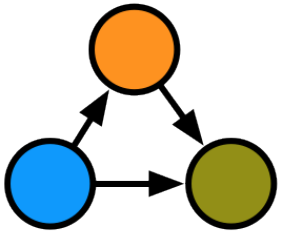
Pattern Matching



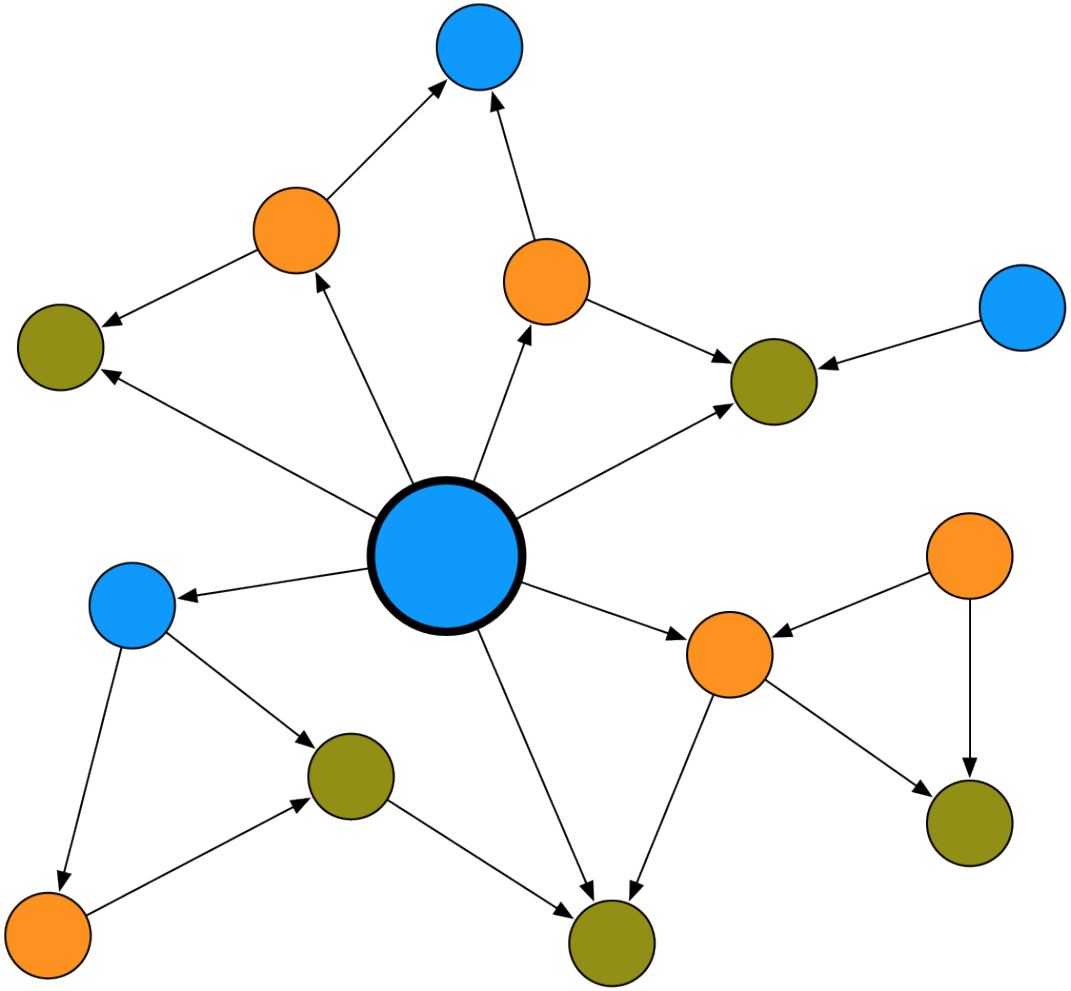
Pattern



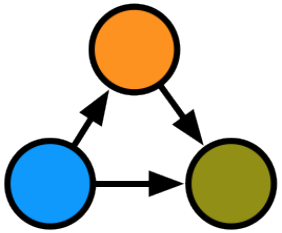
Start Node



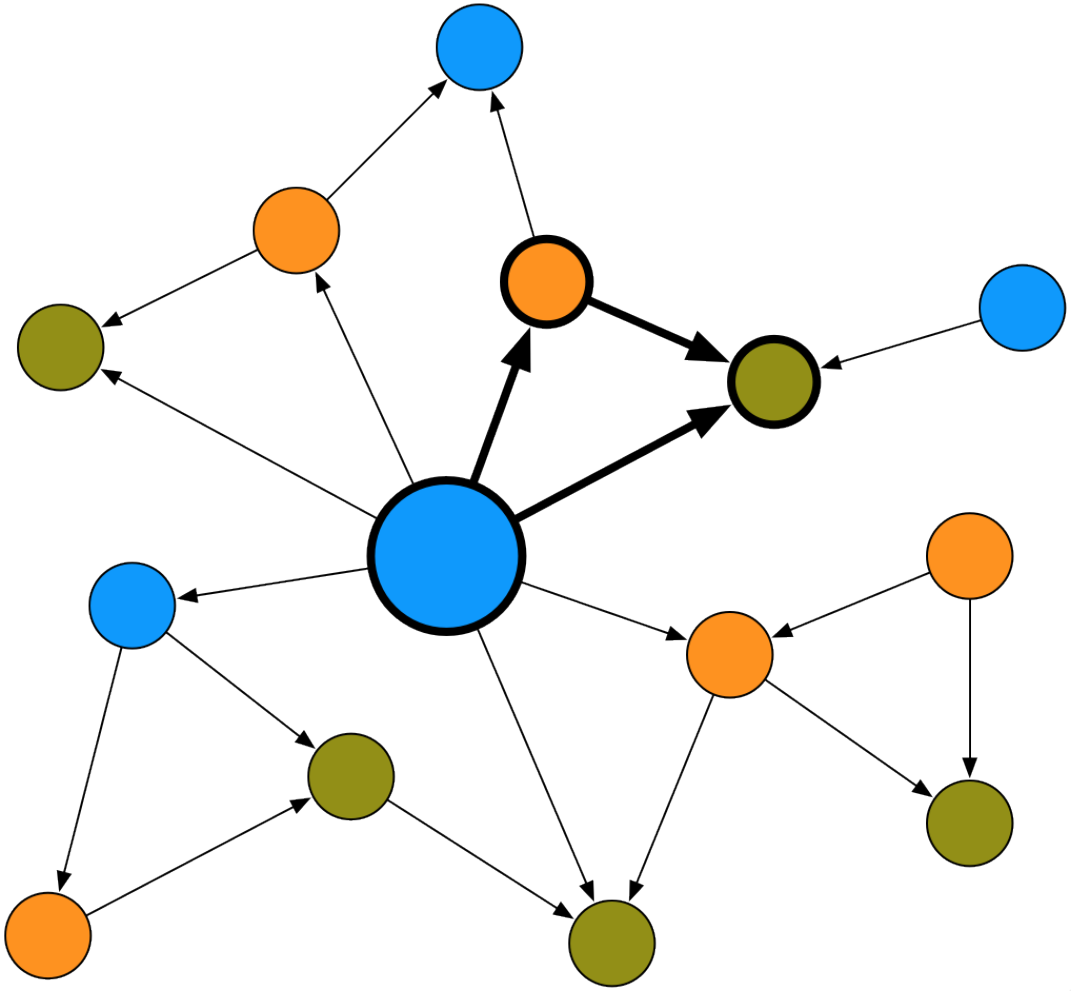
Pattern



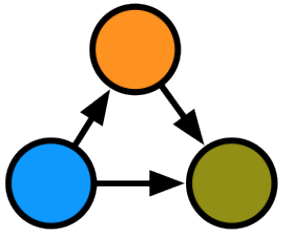
Match



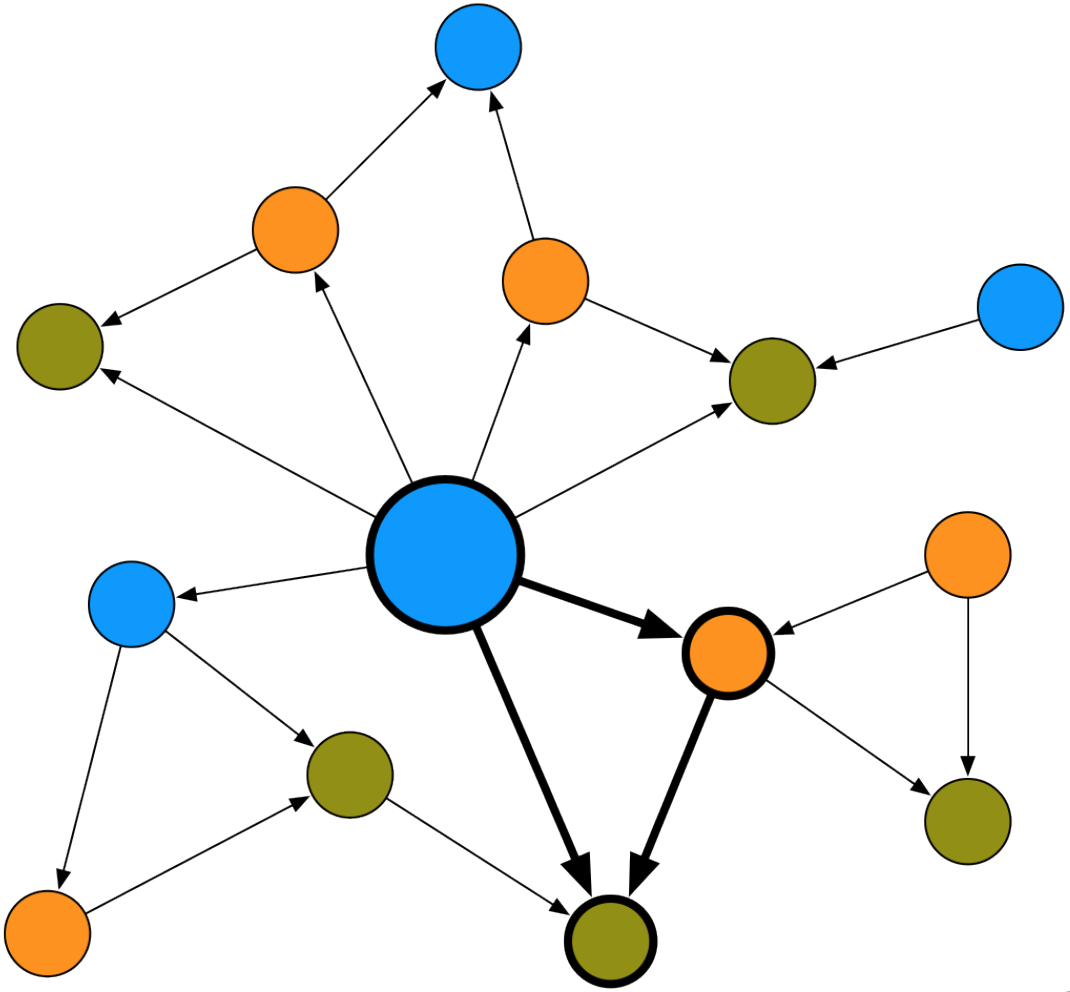
Pattern



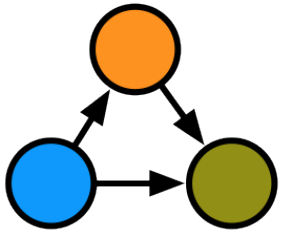
Match



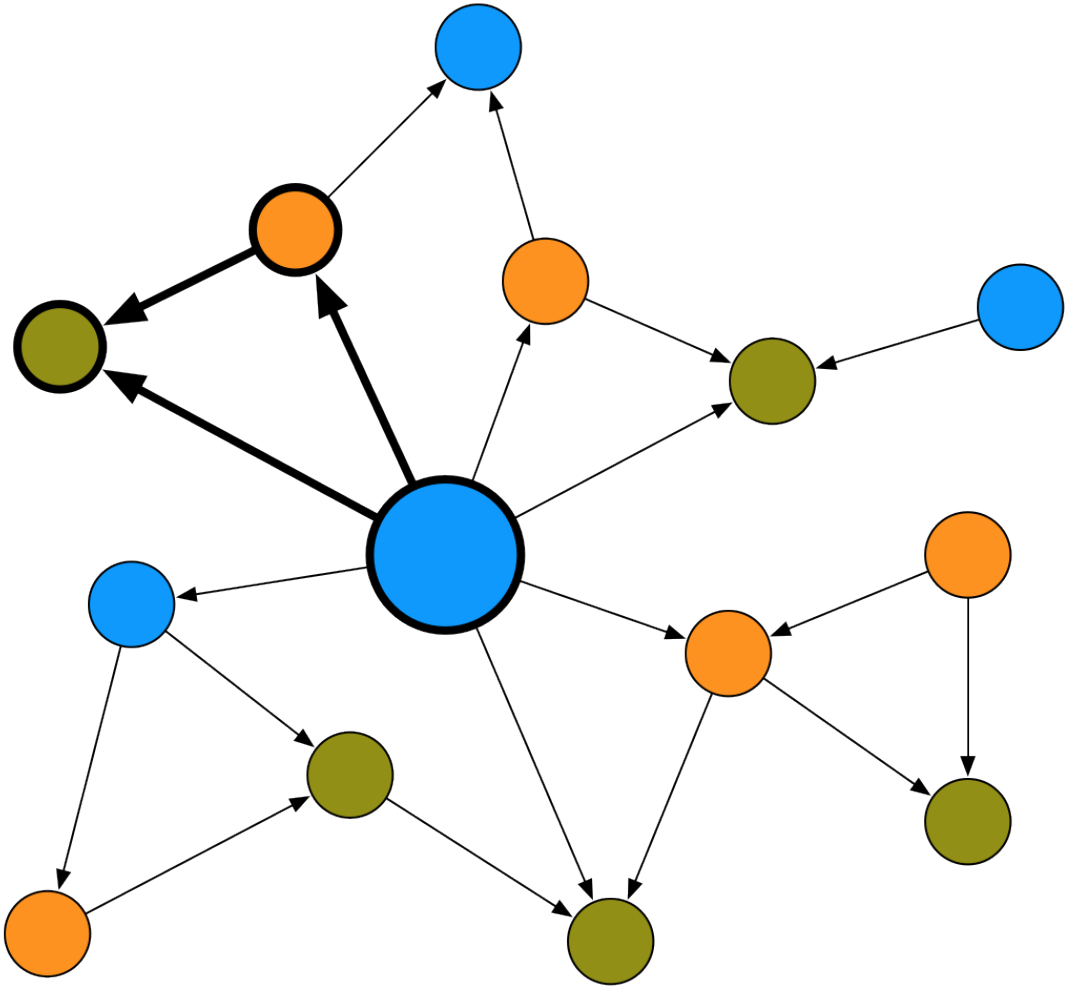
Pattern



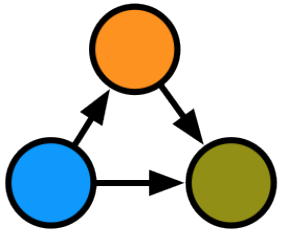
Match



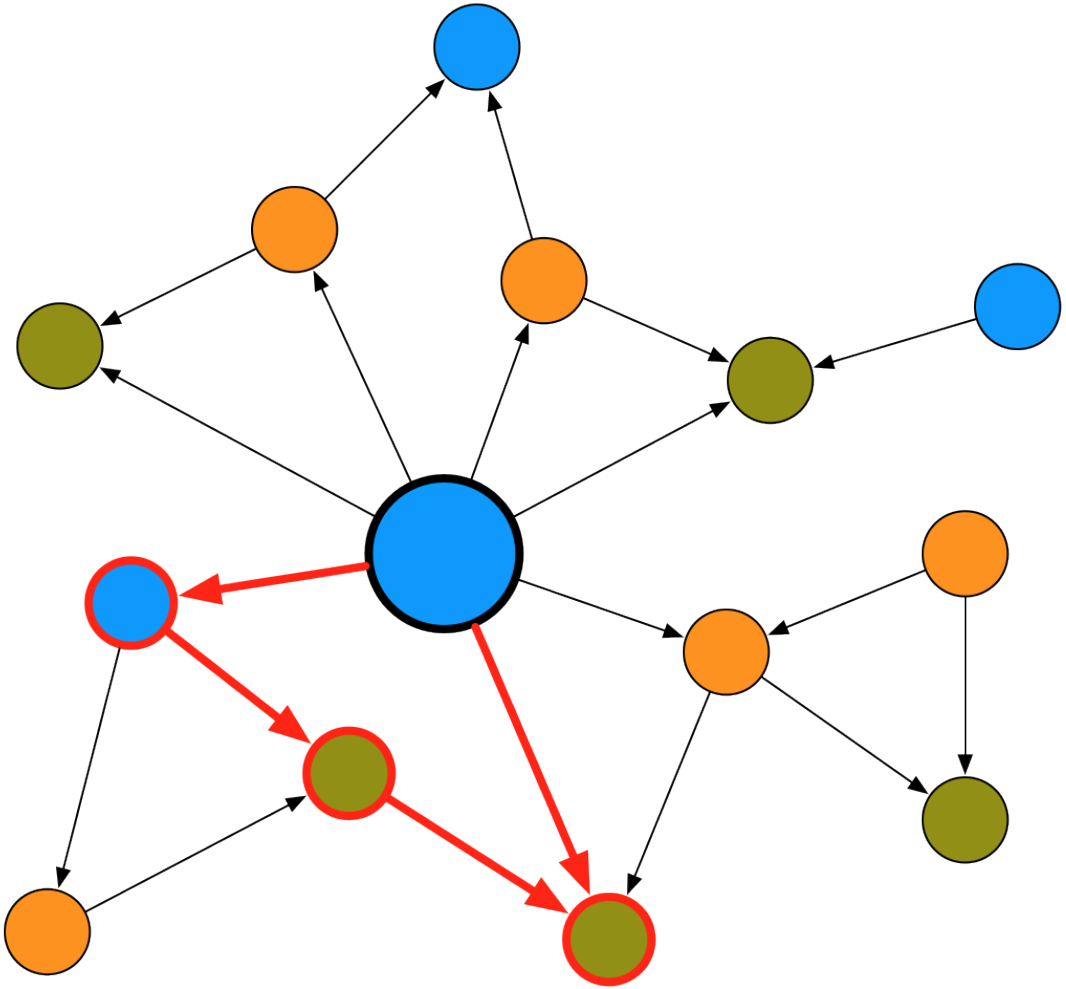
Pattern



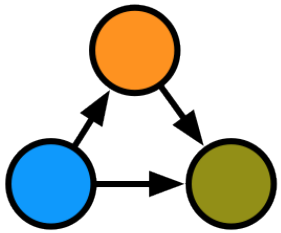
Non-Match



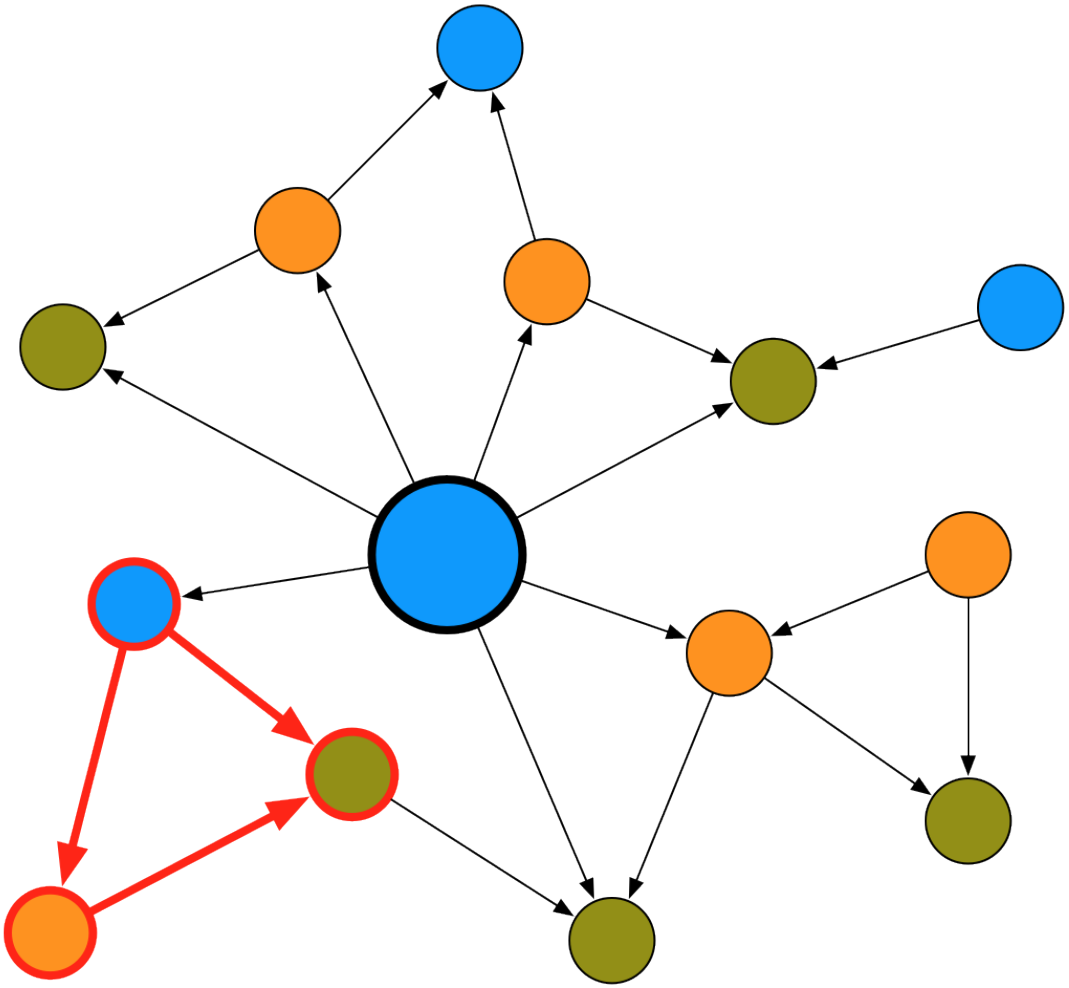
Pattern



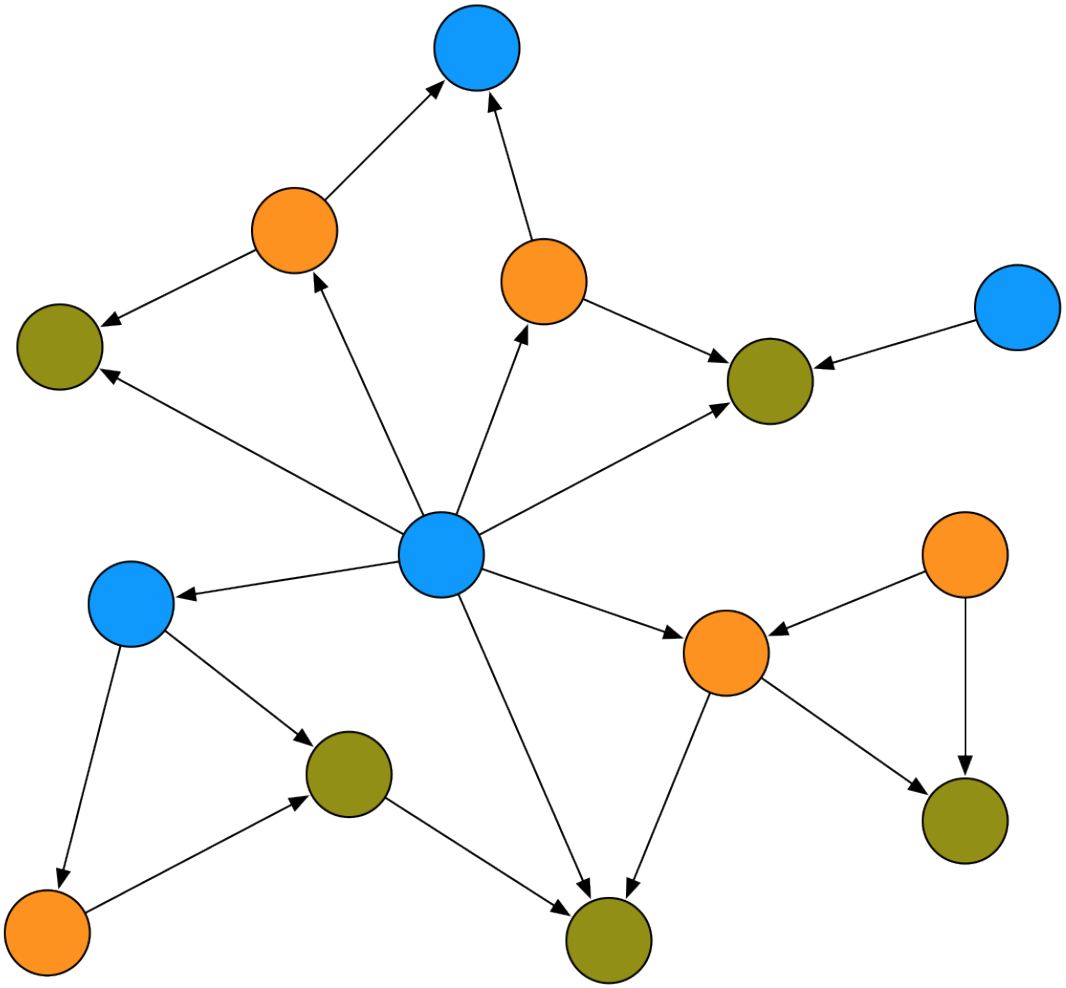
Non-Match



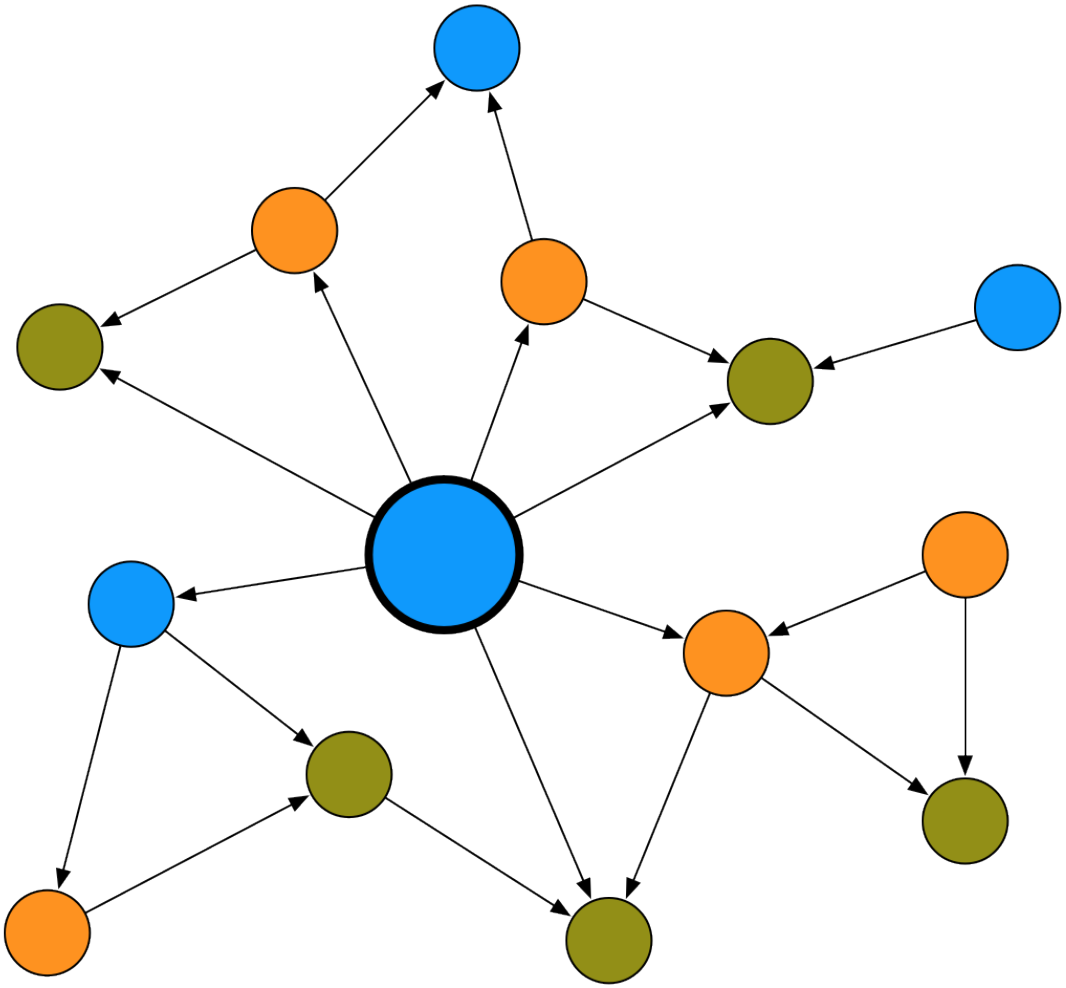
Pattern



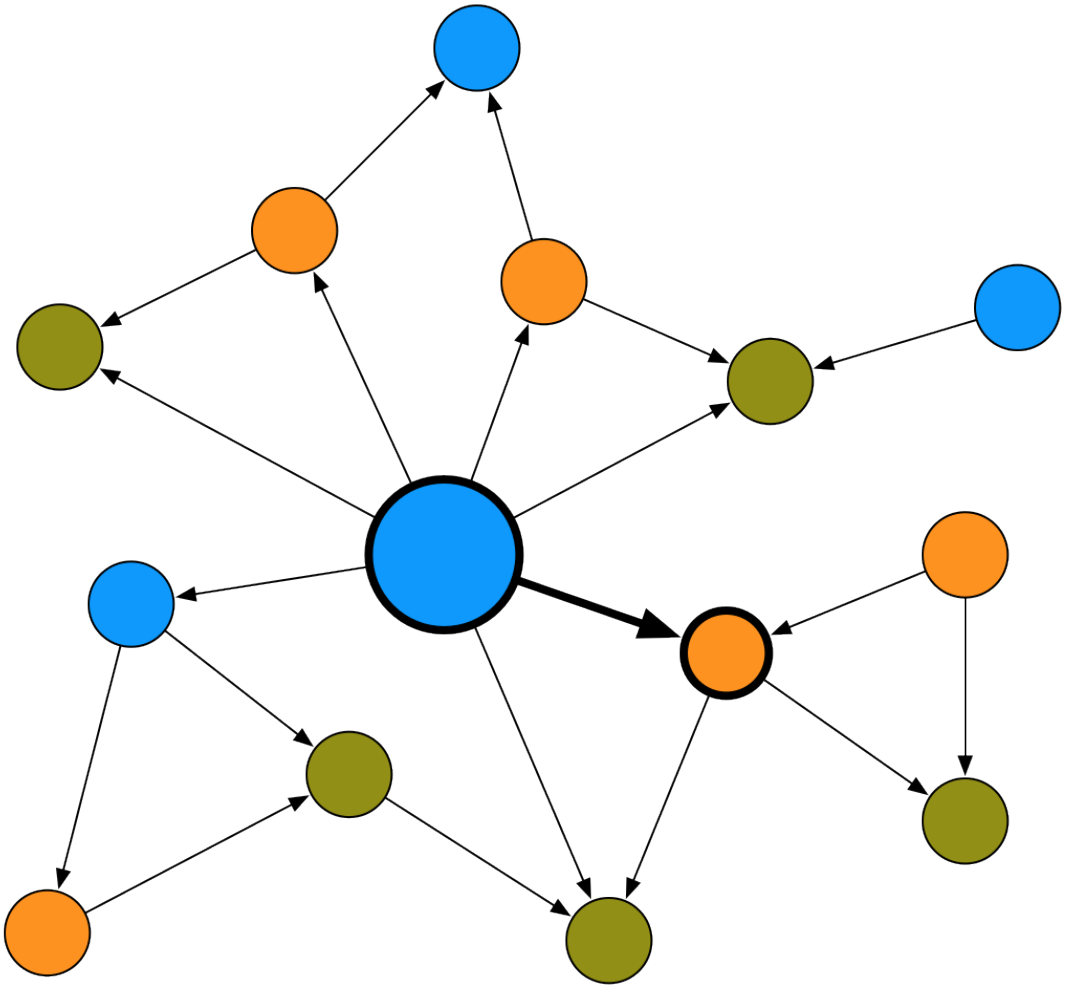
Traversal



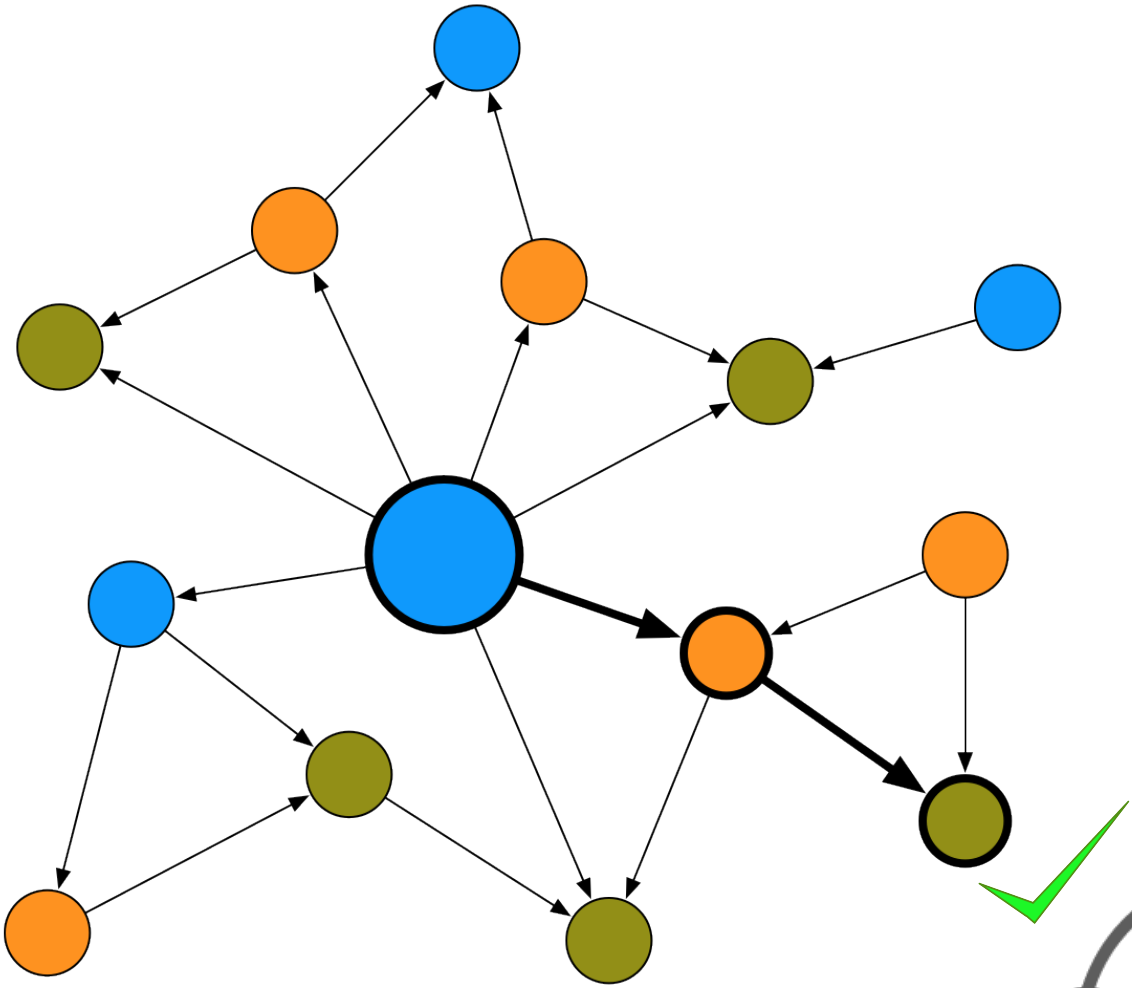
Start Node



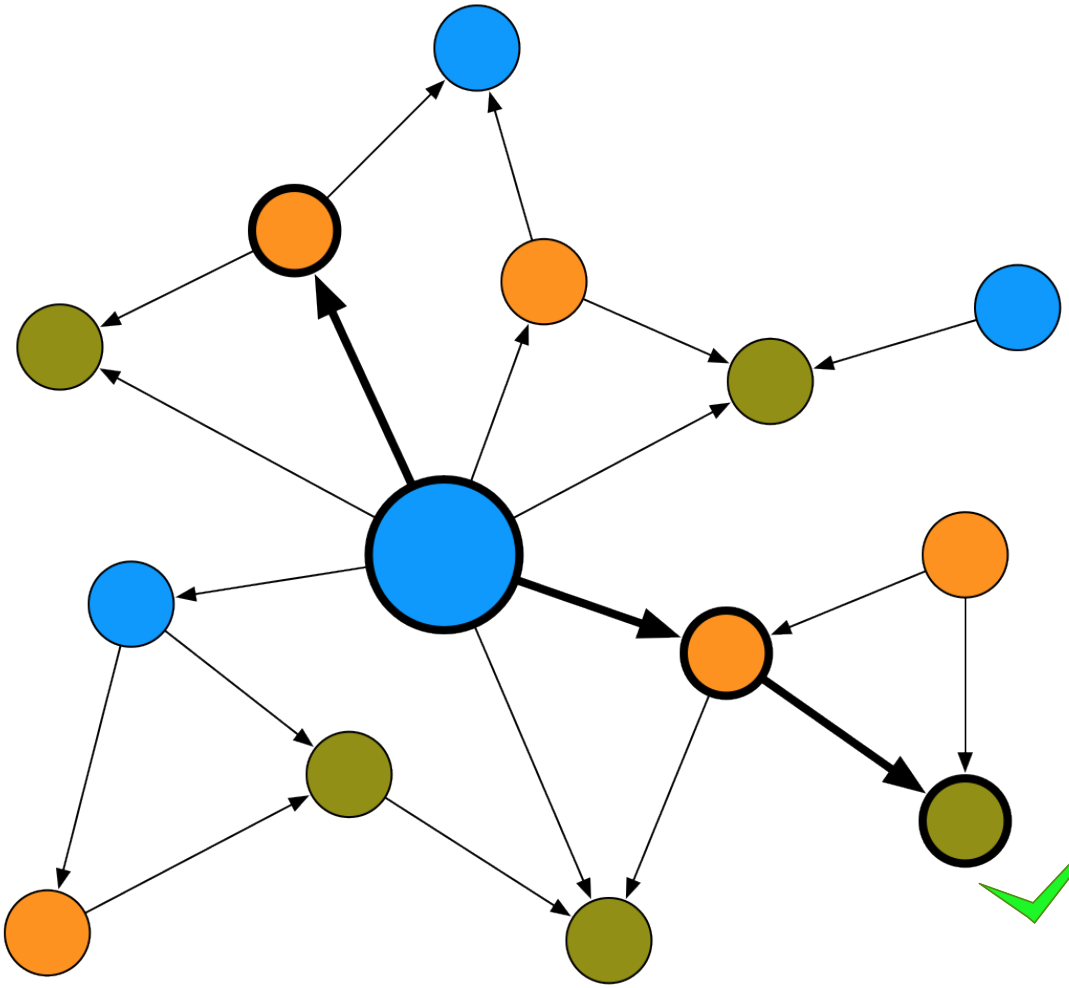
Follow Relationships



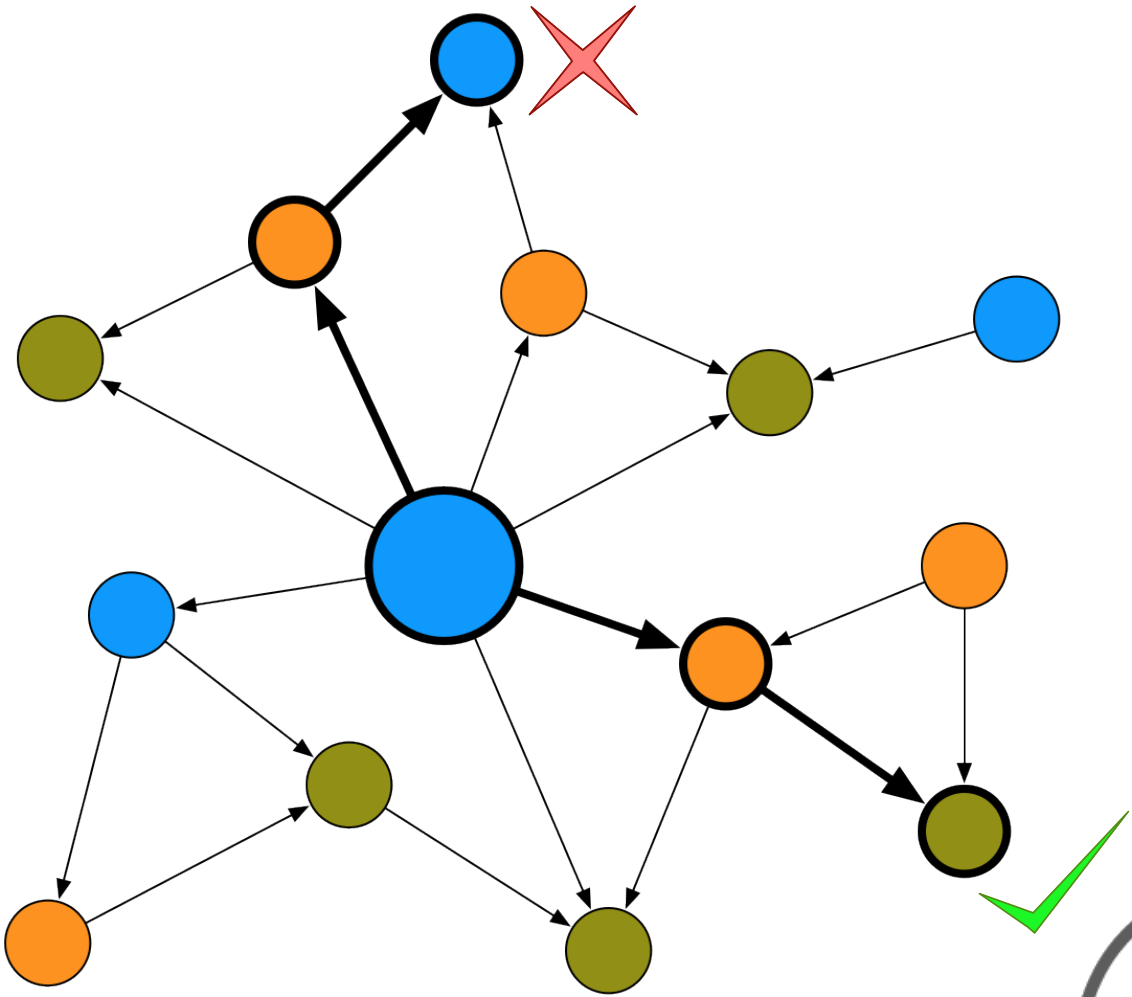
Evaluate Node



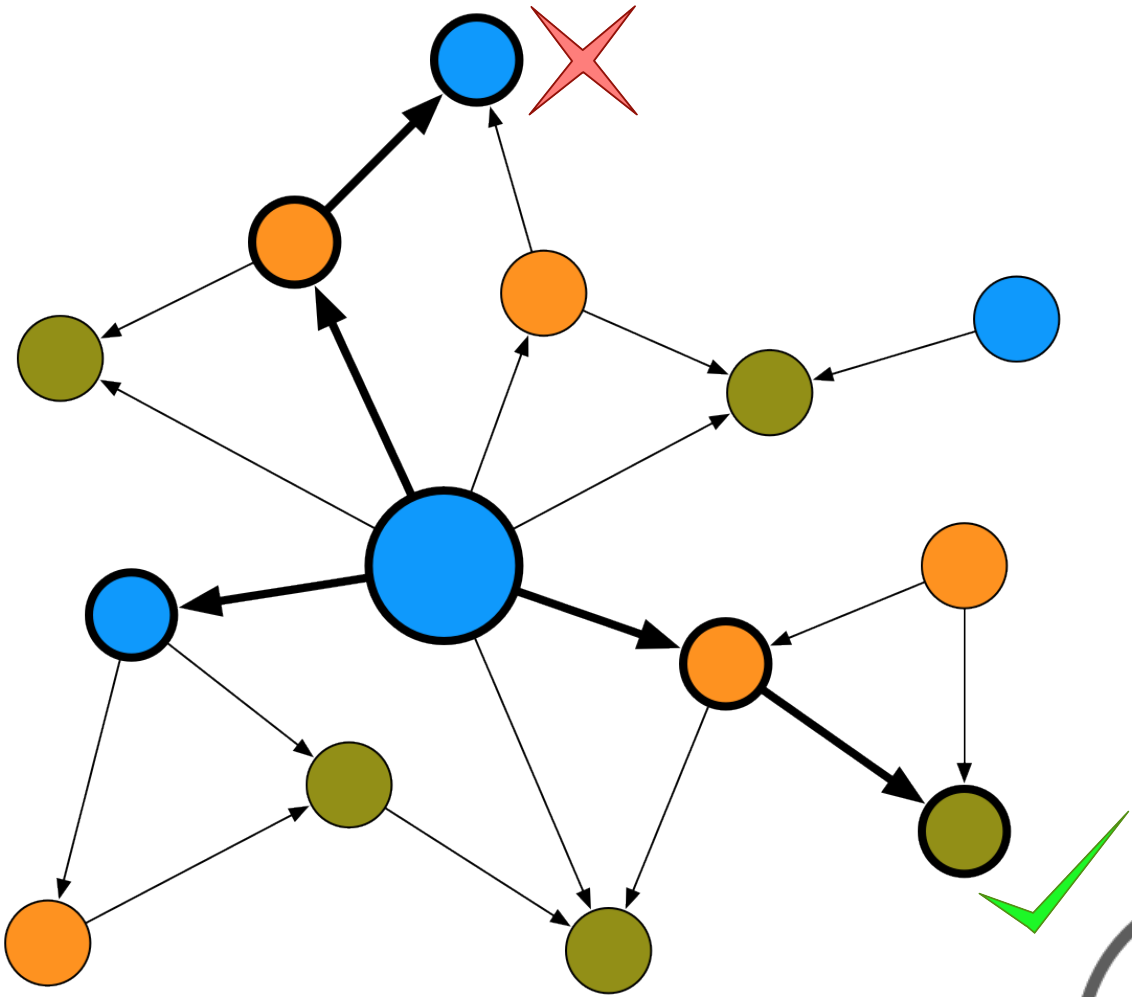
Continue Traversing



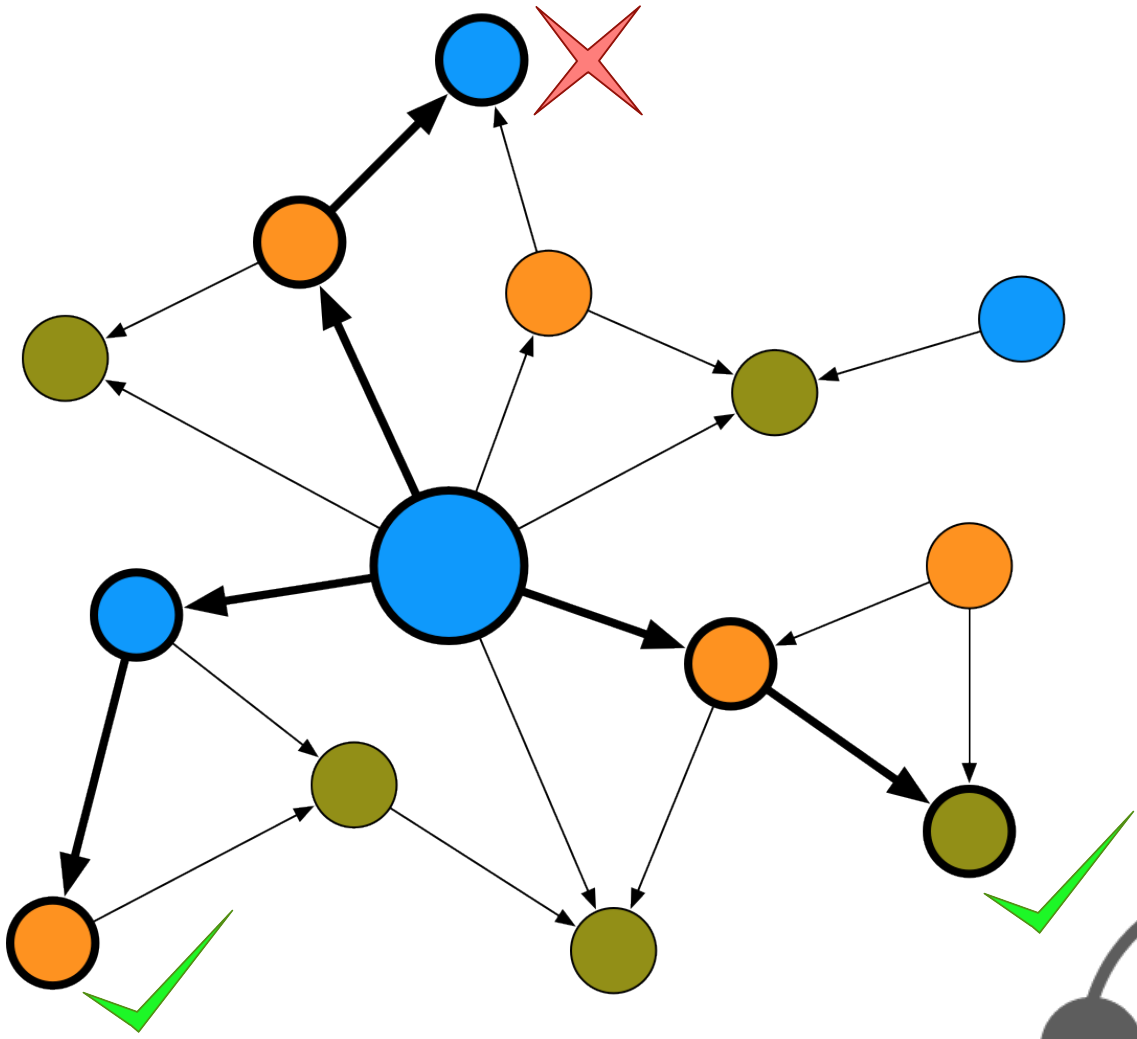
Continue Traversing



Continue Traversing



Continue Traversing



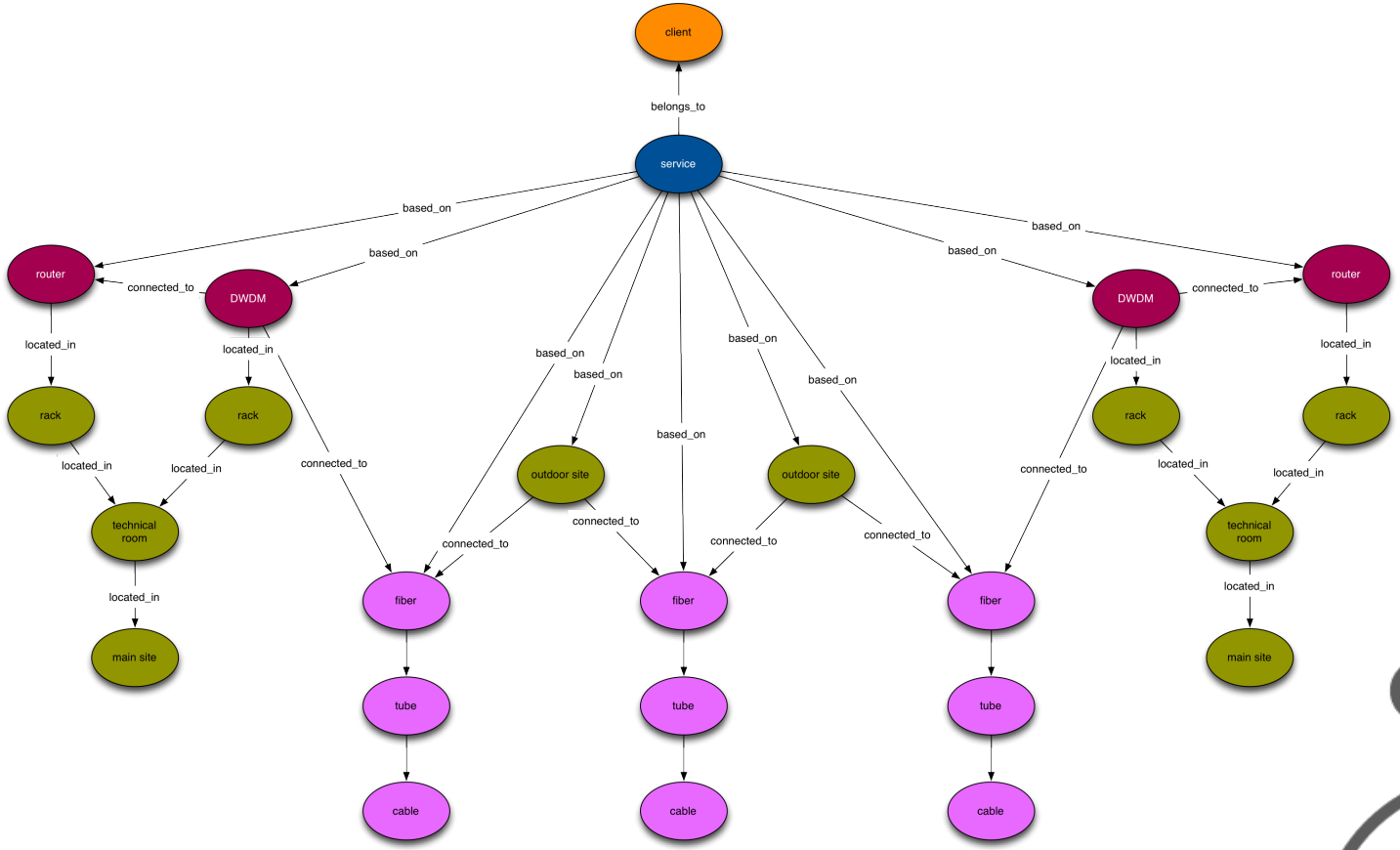
All Trades for Cost Centre

```
START  cc=node:cost_centre(name='Cost Centre 1')
MATCH  cc-[*3]->trade
RETURN trade;
```

Is Party Connected?

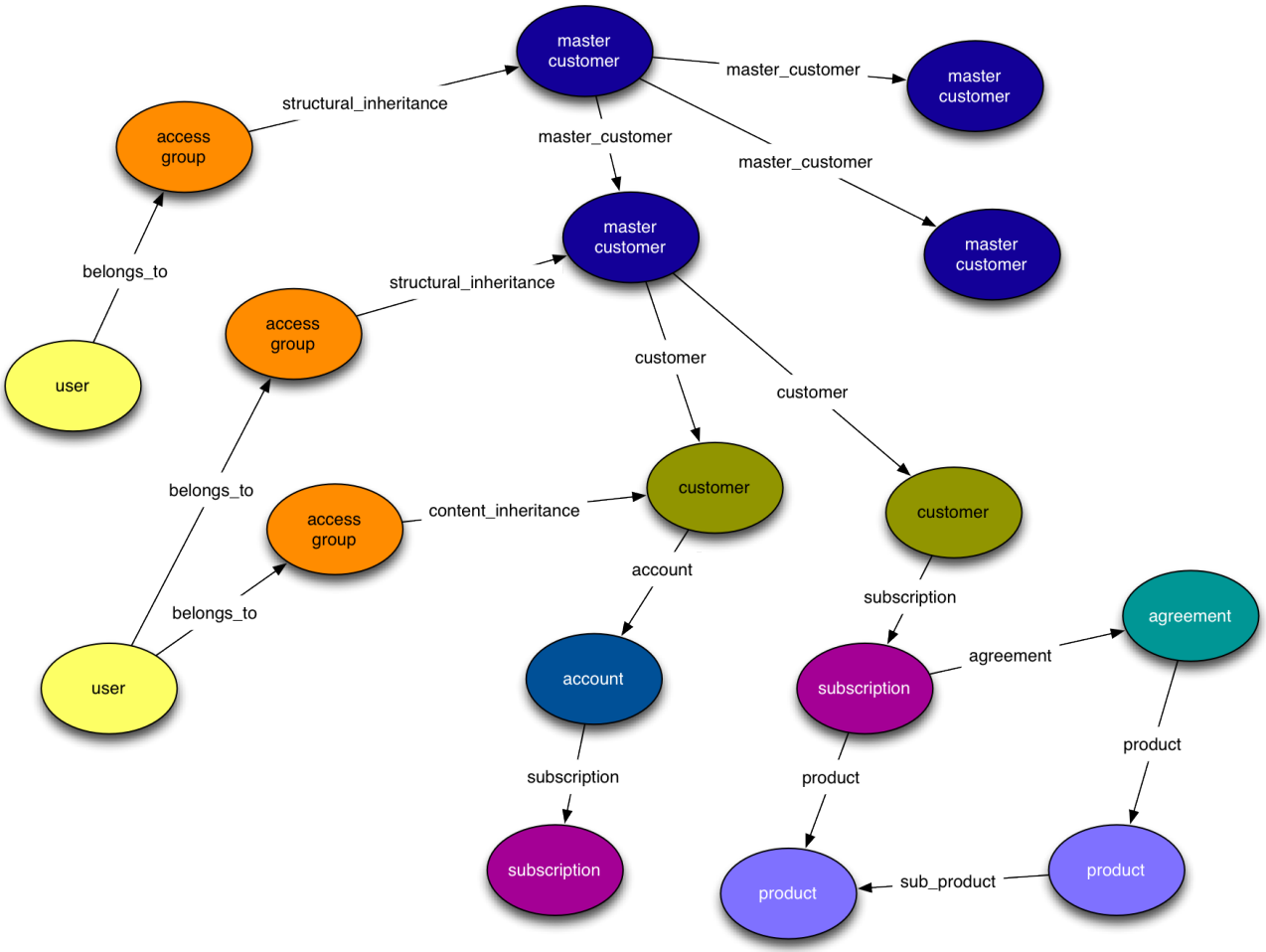
```
START  cc=node:cost_centre(name='Cost Centre 1'),
       party=node:party(name='Party 1')
MATCH  p=shortestPath(cc-[*..5]-party)
RETURN length(p);
```

Network Impact Analysis



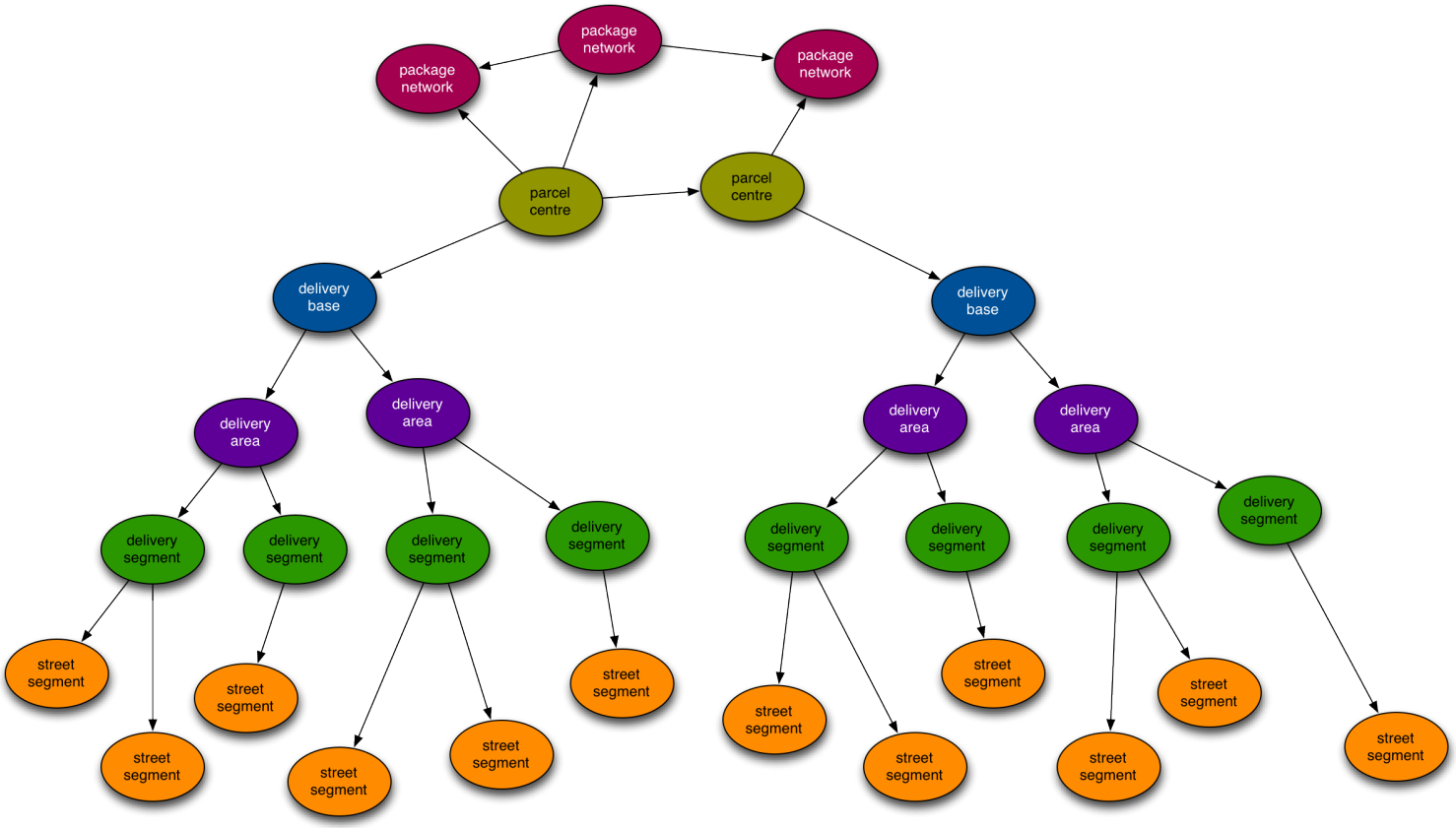
#neo4j

Asset Management & Access Control

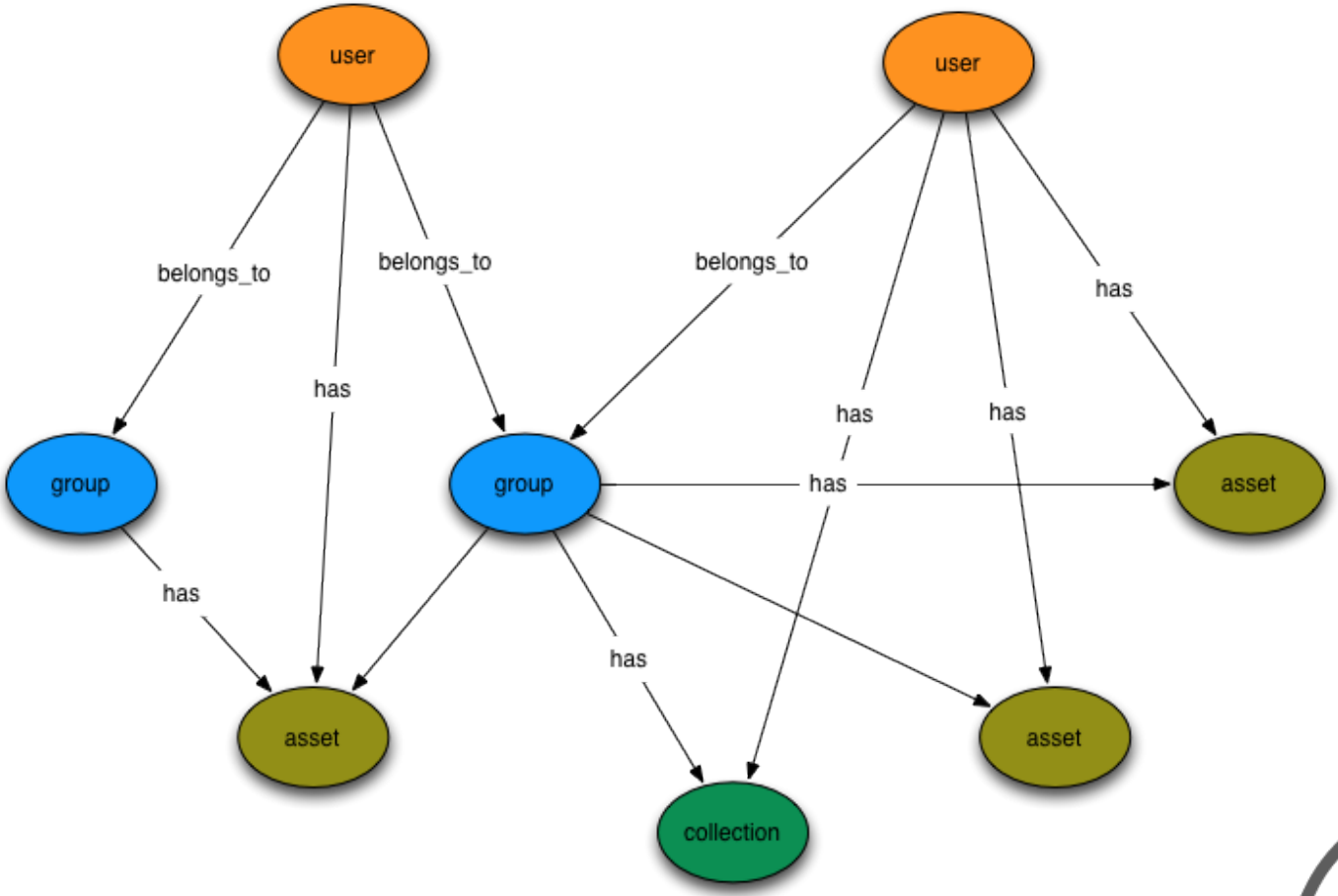


#neo4j

Logistics

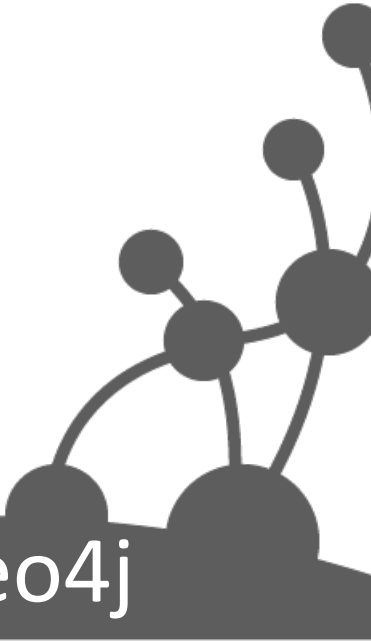

package network
package network
package network
parcel centre
parcel centre
delivery base
delivery base
delivery area
delivery area
delivery area
delivery area
delivery segment
delivery segment
delivery segment
delivery segment
delivery segment
delivery segment
delivery segment
delivery segment
street segment
street segment
street segment
street segment
street segment
street segment
street segment
street segment
street segment
street segment
street segment
street segment

Social Network & Recommendations

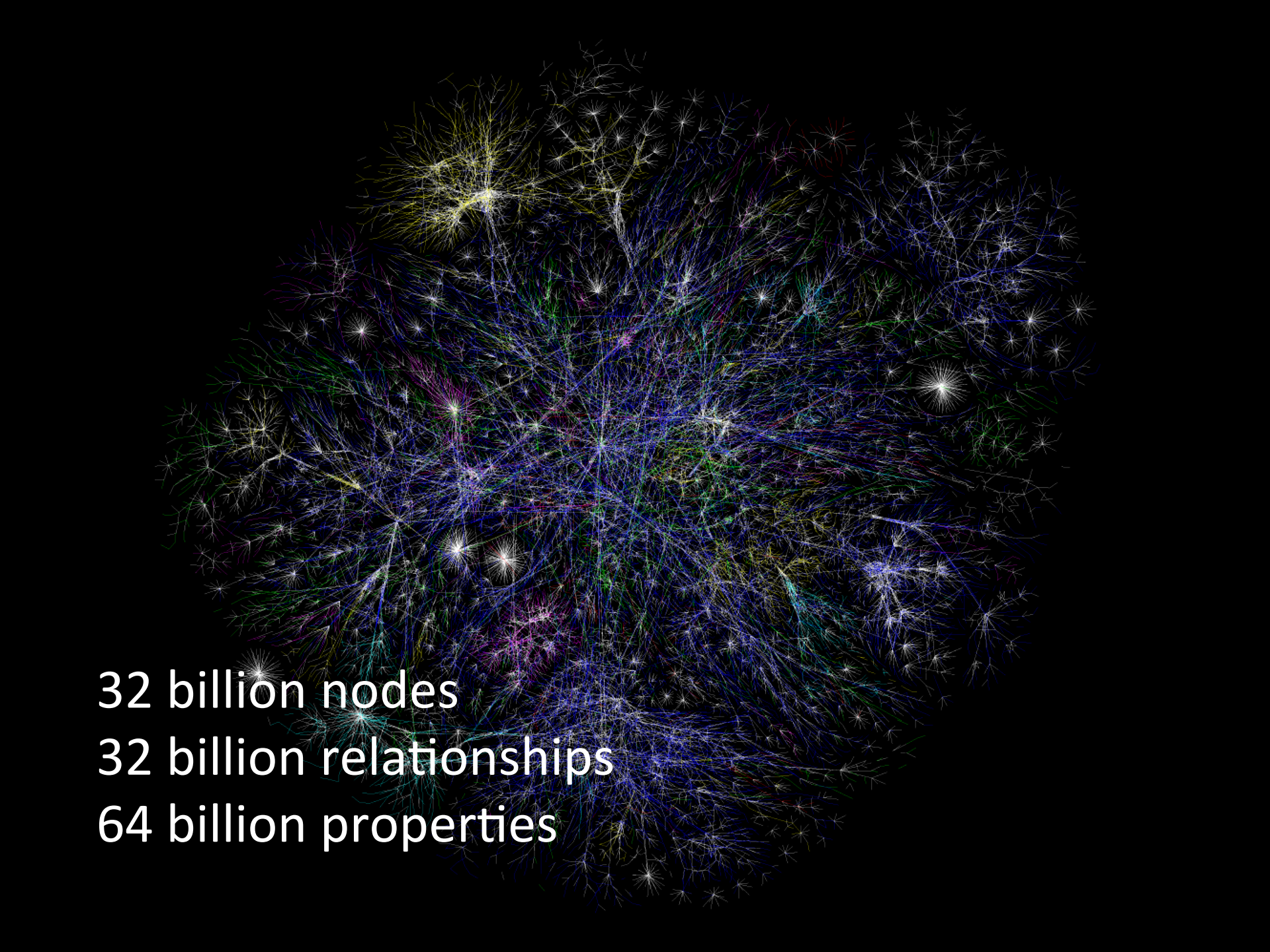


Questions

@iansrobinson
ian.robinson@neotechnology.com



#neo4j


32 billion nodes
32 billion relationships
64 billion properties