Getting Started With Neo4j

Pseudocode to create our 'Toy' Network

==========

Five Nodes

N1 = Tom

N2 = Harry

N3 = Julian

N4 = Michele

N5 = Josephine

Five Edges

e1 = Harry 'is known by' Tom

e2 = Julian 'is co-worker of' Harry

e3 = Michele 'is wife of' Harry

e4 = Josephine 'is wife of' Tom

e5 = Josephine 'is friend of' Michele

==========

A simple text description of a graph

N1 - e1 -> N2

N2 - e2 -> N3

2 - e3 -> N4

N1 - e4 -> N5

N4 - e5 -> N5

==========

A more technical text description of a graph

N1:ToyNode - e1 -> N2:ToyNode

N2 - e2 -> N3:ToyNode

N2 - e3 -> N4:ToyNode

N1 - e4 -> N5:ToyNode

N4 - e5 -> N5

==========

Even more technical pseudo-code

N1:ToyNode - ToyRelation -> N2:ToyNode

N2 - ToyRelation -> N3:ToyNode

N2 - ToyRelation -> N4:ToyNode

N1 - ToyRelation -> N5:ToyNode

N4 - ToyRelation -> N5

==========

Pseudo-code approximating CYPHER code

N1:ToyNode {name: 'Tom'} - ToyRelation {relationship: 'knows'} -> N2:ToyNode {name: 'Harry'}

N2 - ToyRelation {relationship: 'co-worker'} -> N3:ToyNode {name: 'Julian', job: 'plumber'} N2 - ToyRelation {relationship: 'wife'}-> N4:ToyNode {name: 'Michele', job: 'accountant'}

N1 - ToyRelation {relationship: 'wife'} -> N5:ToyNode {name: 'Josephine', job: 'manager'}

N4 - ToyRelation {relationship: 'friend'} -> N5

==========

The actual CYPHER code to create our 'Toy' network

create (N1:ToyNode {name: 'Tom'}) - [:ToyRelation {relationship: 'knows'}] -> (N2:ToyNode {name: 'Harry'}),

(N2) - [:ToyRelation {relationship: 'co-worker'}] -> (N3:ToyNode {name: 'Julian', job: 'plumber'}),

(N2) - [:ToyRelation {relationship: 'wife'}] -> (N4:ToyNode {name: 'Michele', job: 'accountant'}),

(N1) - [:ToyRelation {relationship: 'wife'}] -> (N5:ToyNode {name: 'Josephine', job: 'manager'}),

(N4) - [:ToyRelation {relationship: 'friend'}] -> (N5)

;

More code examples

==========

View the resulting graph

match (n:ToyNode)-[r]-(m) return n, r, m

==========

Delete all nodes and edges

match (n)-[r]-() delete n, r

==========

Delete all nodes which have no edges

match (n) delete n

==========

Delete only ToyNode nodes which have no edges

match (n:ToyNode) delete n

==========

Delete all edges

match (n)-[r]-() delete r

==========

Delete only ToyRelation edges

```
match (n)-[r:ToyRelation]-() delete r
```

//Selecting an existing single ToyNode node

```
match (n:ToyNode {name:'Julian'}) return n
```