Neo4J lab

1. Download and install Neo4j: <http://neo4j.com/download/>
2. Launch Fauxton: <http://localhost:7474/browser>
3. Create a Graph

CREATE (w:Wine {name:"Prancing Wolf", style: "ice wine", vintage: 2015})

View Nodes

$ MATCH (n) RETURN n;

CREATE (p:Publication {name: "Wine Expert Monthly"})

Create a relationship

MATCH (p:Publication {name: "Wine Expert Monthly"}),

(w:Wine {name: "Prancing Wolf", vintage: 2015})

CREATE (p)-[r:reported\_on]->(w)

View the relationship

MATCH ()-[r]-()

WHERE id(r) = 0

RETURN r

Use Postman to view the data

http://localhost:7474/db/data/relationship/0

Add a property to the relationship

MATCH ()-[r]-()

WHERE id(r) = 0

SET r.rating = 97

RETURN r

View graph

MATCH (n) RETURN n;

Create node

CREATE (g:GrapeType {name: "Riesling"})

Add relationship

$ MATCH (w:Wine {name: "Prancing Wolf"}),(g:GrapeType {name: "Riesling"})

CREATE (w)-[r:grape\_type]->(g)

See how to delete a node

CREATE (e:EphemeralNode {name: "short lived"})

MATCH (w:Wine {name: "Prancing Wolf"}),(e:EphemeralNode {name: "short lived"})

CREATE (w)-[r:short\_lived\_relationship]->(e)

MATCH ()-[r:short\_lived\_relationship]-()

DELETE r

MATCH (e:EphemeralNode)

DELETE e

Don’t do this (deletes everything)

MATCH (n)

OPTIONAL MATCH (n)-[r]-()

DELETE n, r

Create more nodes and relationships

CREATE (wr:Winery {name: "Prancing Wolf Winery"})

MATCH (w:Wine {name: "Prancing Wolf"}),(wr:Winery {name: "Prancing Wolf Winery"})

CREATE (wr)-[r:produced]->(w)

CREATE (w:Wine {name:"Prancing Wolf", style: "Kabinett", vintage: 2002});

CREATE (w:Wine {name: "Prancing Wolf", style: "Spätlese", vintage: 2010});

MATCH (wr:Winery {name: "Prancing Wolf Winery"}),(w:Wine {name: "Prancing Wolf"})

CREATE (wr)-[r:produced]->(w)

MATCH (w:Wine),(g:GrapeType {name: "Riesling"})

CREATE (w)-[r:grape\_type]->(g)

MATCH (n) return n;

Add People

CREATE (p:Person {name: "Alice"})

MATCH (p:Person {name: "Alice"}),

(w:Wine {name: "Prancing Wolf", style: "ice wine"})

CREATE (p)-[r:likes]->(w)

CREATE (p: Person {name: "Tom"})

MATCH (p:Person {name: "Tom"}),(w:Wine {name: "Prancing Wolf", style: "ice wine"})

CREATE (p)-[r:likes]->(w)

MATCH (p:Person {name: "Tom"}),

(pub:Publication {name: "Wine Expert Monthly"})

CREATE (p)-[r:trusts]->(pub)

CREATE (p:Person {name: "Patty"})

MATCH (p1:Person {name: "Patty"}),

(p2:Person {name: "Tom"})

CREATE (p1)-[r:friends]->(p2)

MATCH (p1:Person {name: "Patty"}),

(p2:Person {name: "Alice"})

CREATE (p1)-[r:friends]->(p2)

Use -> operator

MATCH (p:Person {name: "Alice"})-->(n)

RETURN n;

See all of the people that Alice is friends with, except let’s return only the name property of those nodes:

MATCH (p:Person {name: "Alice"})--(other:Person)

RETURN other.name;

See which nodes with the label Person are in the graph, but excluding

Patty (boo, Patty!). Note the <> operator, which is used instead of != in Cypher:

MATCH (p:Person)

WHERE p.name <> 'Patty'

RETURN p;

More people

CREATE (p1:Person {name: "Ahmed"}), (p2:Person {name: "Kofi"});

MATCH (p1:Person {name: "Ahmed"}),(p2:Person {name: "Alice"})

CREATE (p1)-[r:friends]->(p2);

MATCH (p1:Person {name: "Kofi"}),(p2:Person {name: "Tom"})

CREATE (p1)-[r:friends]->(p2);

Friends of friends of Alice?

MATCH (fof:Person)-[:friends]-(f:Person)-[:friends]-(p:Person {name: "Alice"})

RETURN fof.name;

Can you find friends of friends of friends of friends?

Use Curl or Postman to create some more nodes – Curl shown below

**curl -i -X POST** [**http://localhost:7474/db/data/node**](http://localhost:7474/db/data/node) **^  
 -H "Content-Type: application/json" ^  
 -d "{\"name\":\"P G Wodehouse\", \"genre\": \"British Humour\"}"**

Use the correct node number below

**curl** [**http://localhost:7474/db/data/node/0/properties/genre**](http://localhost:7474/db/data/node/0/properties/genre)

More nodes

**curl -i -X POST** [**http://localhost:7474/db/data/node**](http://localhost:7474/db/data/node) **^  
 -H "Content-Type: application/json" ^  
 -d "{\"name\":\"Jeeves Takes Charge\", \"style\": \"British Humour\"}"**

Make a relationship (use correct node numbers)

**curl -i -X POST** [**http://localhost:7474/db/data/node/0/relationships**](http://localhost:7474/db/data/node/0/relationships) **^  
-H "Content-Type: application/json" ^  
-d "{\"to\":\"http://localhost:7474/db/data/node/1\",\"type\":\"WROTE\",**\"data\":{\"published\":\"November 28, 1916\"}}**"**

Shortest Path (use correct node numbers)

For example, here we’re looking for a path following relationships of the type WROTE from node 9 using the shortestPath algorithm and capping out at a depth of 10.

**curl -X POST** [**http://localhost:7474/db/data/node/9/paths**](http://localhost:7474/db/data/node/9/paths) **^  
-H "Content-Type: application/json" ^  
-d "{\"to\":\"http://localhost:7474/db/data/node/10\",**\"relationships\":{\"type\":\"WROTE\"},\"algorithm\":\"shortestPath\",\"max\_dept\":10}**"**

Indexing (have a different path) – use correct node number

**curl -X POST** [**http://localhost:7474/db/data/index/node/authors ^**](http://localhost:7474/db/data/index/node/authors%20%5e)

**-H "Content-Type: application/json" ^**

**-d "{\"uri\":\"http://localhost:7474/db/data/node/0\",\**"key\":\"name\", \"value\":\"P.G.+Wodehouse\"}**"**

**curl** [**http://localhost:7474/db/data/index/node/authors/name/P.G.+Wodehouse**](http://localhost:7474/db/data/index/node/authors/name/P.G.+Wodehouse)

Full text search

**curl -X POST** [**http://localhost:7474/db/data/index/node**](http://localhost:7474/db/data/index/node) **^  
-H "Content-Type: application/json" ^  
-d "{\"name\":\"fulltext\", \"config\":{\"type\":\"fulltext\",\"provider\":\"lucene\"}}"**

Now if we add Wodehouse to the full-text index, we get this:

curl -X POST <http://localhost:7474/db/data/index/node/fulltext> ^  
-H "Content-Type: application/json" ^  
-d "{\"uri\":\"http://localhost:7474/db/data/node/0\",\"key\":\"name\",\"value\":\"P.G.+Wodehouse\"}"

Then a search is as easy as a Lucene syntax query on the index URL.

**curl** [**http://localhost:7474/db/data/index/node/fulltext?query=name:P\***](http://localhost:7474/db/data/index/node/fulltext?query=name:P*)

See if you can create a cluster using the Enterprise version – see lecture notes.