IST769 Unit H - Key Value

## Agenda

1. Your Questions
2. Go over Problem Set
3. Unit Coursework Activities

## FRONT MATTER

Start your engines!

PS> **docker-compose down**

PS> **docker-compose up -d neo4j jupyter drill**

Neo4J: [http://localhost:7474](http://localhost:8882)

**Neo4j:**

* What is the storage model? (relational, document, etc…)
  + Graph
    - Nodes / Edges
    - Labels
    - Attributes
* What are the logical model metaphors? (tables, keys constraints, etc…)
  + Node ID (globally unique number auto increments)
* How does it scale horizontally? CAP? (after all - its a big data course!)
  + Shards / Replica
  + Partitioning ???
* Purpose of database? OLTP (CRUD) or OLAP/Analytics (CR)?
  + Complex relationship-oriented data with recursive (related to self)
  + Graph stores the join in the node.
* How do you design it?
  + ER Modeling works
* How to connect to it from:
  + Client / Native Scripting Language
    - Own Libs / UI / Query Language Cypher
  + Drill / SQL
    - No
  + Spark / PySpark
    - Yes

## 1. Your Questions

Ask any questions you have here!

* Why is the relationship in graph data models not two ways? - not necessary
* Does neo4j store relationships within nodes - yes
* Why is it important to return codes or attributes before sorting?
* Do certain spark transformations execute on neo4j? - all of them
* Is neo4j compatible with spark sql? - yes.

## 2. Coursework Activities

Let’s build a simple graph, using Cypher

merge(e:Employee {name:'Mike', title:"CEO"});

merge(e1:Employee {name:'Alice', title: 'Store Manager'});

merge(e2:Employee {name: 'Betty', title:'Department Manager'});

merge(e3:Employee {name: 'Charles', title:'Department Manager'});

**Nodes:**

Employee

name: Mike

title: CEO

Employee

name: Alice

title: Store Manager

Employee

name: Betty

title: Dept Manager

Employee

name: Charles

title: Dept Manager

match(em:Employee {name :'Mike'})

match(ea:Employee {name: 'Alice'})

match(eb:Employee {name: 'Betty'})

match(ec:Employee {name: 'Charles'})

merge(em)-[:SUPERVISES]->(ea)

merge(em)-[:SUPERVISES]->(eb)

merge(em)-[:SUPERVISES]->(ec);

**Relationships:**

Supervisors

Mike -> Alice

Alice -> Betty

Alice -> Charles

match(em:Employee {name :'Mike'})

match(ea:Employee {name: 'Alice'})

match(eb:Employee {name: 'Betty'})

match(ec:Employee {name: 'Charles'})

merge(c1:Committee {type:'Personnel'})

merge(c2:Committee {type:'Employee Picnic'})

merge(c1)<-[:MEMBER\_OF]-(em)

merge(c1)<-[:MEMBER\_OF]-(ec)

merge(c2)<-[:MEMBER\_OF]-(em)

merge(c2)<-[:MEMBER\_OF]-(ea)

merge(c2)<-[:MEMBER\_OF]-(eb);

Committees:

Personnel

Members: Mike, Charles

Employee Picnic

Members: Mike, Alice, Betty

// Dete alice

match(e:Employee {name:'Betty'})-[m:MEMBER\_OF]->(c:Committee {type: "Employee Picnic"})

delete m;

**Questions:**

1. Show name title of members of the Employee Picnic committee
2. Show direct reports of Mike and their reports and their direct reports
3. For mike's direct reports, what committees are they on?