Neo4j Lab

Be prepared to demonstrate your knowledge of all of the following concepts and operations.  Keep track of your scripts and any learning and notes in this document.

On the due date you will be required to demo your work and discuss what you have learned. You may use this document as reference during the pass off.

* Get Neo4j up and running on your machine. No clustering or sharding is required.
* You can download Neo4j to windows
* What are the Neo4j equivalents of an (if there are equivalents):
* RDBMS database: Database
* RDBMS table: Graph
* RDBMS row: Node
* What is the case sensitivity of Neo4j?

Data is case-sensitive, queries are not.

* Import all 10,000 people records from Lab 1 into your Neo4j instance.
* Are you required to have any relationships between these new people records?

No

* What relationships might we want to record about these employee records (think about common business scenarios.)

Bosses, who is in a team with each other, who leads the group

* How are these relationships different from how we’d store them in a relational database or MongoDB?

In a relational database, we’d create a separate table for the relationships and use foreign keys for the connections. Here, the connections are kind of their own thing.

* Demonstrate all four CRUD operations on a single record by key on the people collection in your Neo4j instance.

Create (n:Person {id: 001, name: “Greeble Gonky”, age: 25})

Match (n:Person {id: 001}) return n

Match (n:Person {id:001})

Set n.age = 26

Match (n:Person {id:001}) detach delete n

* Complete the following queries on your new data set:
* Return a single record by it’s key. (In SQL: …Where PrimaryKey = x)

Match (a:Record {primaryKey: 5318008}) return a

OR

Match (a:Record) where a.primaryKey = 5318008 return a

* Return all the records with the same value for a field. (Where SomeField = x)

Match (b:Record {name: “Yimmy”}) return b

* Return all the records that participate in a particular relationship (i.e. Person is Married to Person)

Match (a:Person)-[r:Friends]->(b:person) return a, r, b

* Is there a new4j way to….
* Return all the records \_having\_ a particular field. (Is there a SQL equivalent for this?)

Match (n) where exists(valueWeWant) return n

* List the records missing a particular field.

Match (n) where not exists(valueWeWant) return n

* Return the count of the records that have fields with a particular value. (Count(\*) …. Where Field = ‘test’)

match (n:Item {value: “WhatWeWant”}) return count(n)

* Does neo4j have support for robust indexes? If so, ho would you create an index on the people records you imported.

Yeah, we can make indexes and they’re pretty dope, they make things fast  
Create index NameOfIndex for NODE/RELATIONSHIP on VALUE  
Create index people for (n:Person) on n.name

* Summarize the steps to create a cluster in neo4j. You don’t need to carry out the steps unless you want to… just be able to describe what would be required for setting up a cluster and scaling it up or down horizontally.

Set the settings of your clusters  
dbms.default\_advertised\_address - sets the default ip address all replica servers must connect to  
dbms.mode - sets whether the server is primary or secondary  
dbms.clustering.enable=true - allows single instances to cluster  
causal\_clustering.initial\_discovery\_members - must be set to set cluster as the same value as others  
You need to set up a primary server, then configure the replica servers  
Compare and contrast the strengths and weaknesses of Neo4j vs

* Relational Databases

Neo4j as a graph database is created for highly connected data, meaning it’s “joins” are much faster and more optimized than those found in relational databases. However, neo4j doesn’t have a strict format for each item like those found in relational databases, meaning that it’s harder to keep data strictly the same.

* MongoDB

Both databases allow you to input data without following a strict layout. Mongo only allows for JSON objects which contain variables, neo4j has nodes instead.

* Serialized Files

Storing/reading connections between data in Serialized files would be a major hassle to setup and would require deserialization for every query. Serialized files could be stored on your own system without needing to connect to a database.

* HashMap/HashTable

Hashes take up more storage space, but are fairly fast. However, Hashmaps are designed for single record access, and neo4j shines when accessing the relationships between many records at once.

* Be prepared to discuss three examples of software systems for which Neo4j would be a wise database choice.

A system where you have many connections between many nodes is where neo4j would excel

A database for a company’s network configuration.

A database for a tournament of some kind to display the matchups.

* Be prepared to discuss examples of software systems for which Neo4j would be a poor database choice.

A system where you have very few connections would be easier to setup, use and maintain with a different database. Neo4j is slower at accessing individual records.