**Neo4j Practical Overview**

Neo4j was first created as an embedded Java database; this is where that “4j” in the name comes from. It has since evolved beyond that, so the “4j” Java aspect is no longer really relevant today, but is a nod to its history.

Neo4j was created to address some problems that the founders were having in building a Content Management System (CMS), specifically for some of the rights and metadata around the usage of photos. They found it very difficult to represent in a relational database because of all of the different connections and relationships, and the richness of the data. So that’s why Neo4j was first created.

The Cypher query language is used with Neo4j, a graph database, to create, query, update, and delete data represented as nodes and relationships in a graph. Here's an end-to-end example covering key aspects of Cypher and Neo4j usage:

1. Setting up the Environment

Description about the project

This project is build using technologies Docker, Flask (Web Framework in python), Swagger UI and Neo4j. These all components are we sorted and connected to each other, using docker internal network, to start the application we just need to do following.

Clone this repository

navigate to the below directory you cloned the git repo to using the terminal or Command Prompt and run below command

$ docker-compose up

NOTE: Before running the above command please make sure that the docker daemon is running.

Once you see the bunch command called started, you can access the neo4j application url which shows you the screen for login to the neo4j website.

we can use the default username "neo4j" and default password "neo4j" initally and you end up with password change form please change the password to "SLU@2024".

Now as we set up our database we can access the Swagger UI from here.

from Swagger we can hit the requests and get responses for the course and student and subject nodes or tables.

NOTE: This application is showing very limited basic features of the neo4j like creating, reading , update and delete (CRUD) operations.

2. Creating Nodes

Nodes represent entities. Let's create some nodes representing people and a city:

*CREATE (alice:Person {name: 'Alice', age: 30})*

*CREATE (bob:Person {name: 'Bob', age: 25})*

*CREATE (chicago:City {name: 'Chicago'})*

3. Creating Relationships

Relationships connect nodes and can have properties. Let's connect our people with relationships indicating they live in Chicago and know each other:

*MATCH (alice:Person {name: 'Alice'}), (bob:Person {name: 'Bob'}), (chicago:City {name: 'Chicago'})*

*CREATE (alice)-[:KNOWS {since: 2020}]->(bob)*

*CREATE (alice)-[:LIVES\_IN]->(chicago)*

*CREATE (bob)-[:LIVES\_IN]->(chicago)*

4. Querying Data

To find data, use `MATCH` statements. Let's find everyone who lives in Chicago:

*MATCH (p:Person)-[:LIVES\_IN]->(c:City {name: 'Chicago'})*

*RETURN p.name, p.age*

5. Updating Data

To update, use `SET`. Let's update Alice's age:

*MATCH (alice:Person {name: 'Alice'})*

*SET alice.age = 31*

*RETURN alice.name, alice.age*

6. Deleting Data

To delete, use `DELETE`. If you want to delete a node and its relationships, use `DETACH DELETE`. Let's delete Bob:

*MATCH (bob:Person {name: 'Bob'})*

*DETACH DELETE bob*

7. Advanced Queries

Cypher allows complex queries, such as finding paths or using aggregation. Let's find the names of people who know someone in Chicago, and the length of their acquaintance:

*MATCH (p:Person)-[r:KNOWS]->(friend:Person)-[:LIVES\_IN]->(c:City {name: 'Chicago'})*

*RETURN p.name, friend.name, r.since*

8. Indexing for Performance

To improve query performance, create indexes on frequently queried properties:

*CREATE INDEX ON :Person(name)*

*CREATE INDEX ON :City(name)*

9. Importing Data

For bulk data, you can use `LOAD CSV` to import data from a CSV file:

*LOAD CSV WITH HEADERS FROM 'file:///people.csv' AS line*

*CREATE (:Person {name: line.name, age: toInteger(line.age)})*

**Theoretical Questions**

1. What is Neo4j and why is it used?

2. Explain the concept of graph databases versus traditional relational databases.

3. What are nodes, relationships, properties, and labels in Neo4j?

4. How does Neo4j's data model differ from SQL-based models?

5. Describe the advantages of using a graph database for handling complex relationships.

6. What is Cypher? How does it compare to SQL?

7. Explain the concept of index-free adjacency in Neo4j.

8. How does Neo4j ensure data integrity and transaction management?

9. Discuss the importance of relationship types in Neo4j.

10. What is the role of property graphs in Neo4j?

**Practical Query-Based Questions**

1. Write a Cypher query to create a node with labels `Person` that has `name` and `age` properties.

2. How would you find nodes with a specific label and property value using Cypher?

3. Write a query to create a relationship of type `FRIENDS\_WITH` between two `Person` nodes.

4. How can you update the `age` property of a `Person` node?

5. Write a Cypher query to delete a node by its `name` property, considering it has no relationships.

6. Construct a query to return all nodes connected by a `WORKS\_WITH` relationship.

7. How do you use the `LIMIT` clause in a Cypher query to restrict the number of returned results?

8. Write a query to find the shortest path between two nodes.

9. How can you use Cypher to create a graph pattern that does not exist yet using the `MERGE` keyword?

10. Construct a query that returns all `Person` nodes sorted by their `age` in descending order.

11. Write a Cypher query to find all `Person` nodes that have a `name` property starting with 'A'.

12. Construct a query to find the total number of relationships coming out of a node with label `Employee`.

13. How can you find the node with the highest number of outgoing relationships in a graph?

14. Write a Cypher query to return all pairs of `Person` nodes that share a common `LIKES` relationship with the same `Movie` node.

15. Create a query that finds the longest path in a graph between two `Person` nodes.

16. How would you modify the properties of a relationship in Neo4j using Cypher?

17. Write a query to delete all nodes with a specific label that have no relationships.

18. Construct a Cypher query to merge two nodes, keeping all their distinct relationships.

19. How can you find cycles in a graph using Neo4j Cypher queries?

20. Write a query to list all nodes and their relationships, including relationship properties.

31. Create a query to find the common friends between two `Person` nodes.

32. How can you implement pagination in results returned by a Cypher query?

33. Write a query to clone a subgraph within the same Neo4j database.

34. Construct a query to return the count of each type of relationship in the database.

35. How would you find orphan nodes (nodes without any relationships) in your graph?

**Advanced Questions**

1. Explain how you would model a recommendation system using Neo4j.

2. How can Neo4j be integrated with other technologies for a full-stack application?

3. Discuss the concept of graph centrality and how it can be calculated in Neo4j.

4. How does Neo4j handle large-scale data migrations?

5. Describe the process of securing a Neo4j database.

6. How do you perform a full-text search in Neo4j?

7. Explain the concept of graph algorithms in Neo4j and give an example use case.

8. Discuss the role of Neo4j in real-time data analysis.

9. How can you optimize Cypher queries for performance?

10. Explain how Neo4j's clustering and replication features support high availability and scalability.