NEO4J SALE CONSULTANT EXERCISE - BUILD A SIMPLE MUTUAL FUND GRAPH

Goal

This exercise is meant to give a prospective Neo4j Sales Engineer a platform to demonstrate their client engagement and technical sales abilities by using a simple scenario that involves building a Neo4j graph. The result is expected to be a short presentation on the value of Neo4j as if you're presenting to a sales prospect, along with a working graph database and example queries integrated into the presentation as you see fit. Feel free to embellish your scenario with common implementation architectures and technologies you've encountered at large enterprises with complex systems. The presentation should be no longer than 30 – 40 minutes including time for questions.

At the end of this exercise, you should be able to ask yourself if you're excited about the Neo4j technology and could see yourself as being effective in positioning and selling it to complex, enterprise scale accounts. If so, then Neo4j would be a very rewarding work environment for you.

Hint: *The presentation should include identifying and addressing a business problem, introducing why the graph is unique to solving the problem, some of the Neo4j Label Property Graph functionality that uniquely address the problem, and finally anticipating any objections that could be raised by a prospect.*

Please reach out to Neo4j with if you'd like to validate any assumptions or have questions.

Good luck and good selling!

Exercise Technical Goal

This exercise introduces some core Neo4j graph database design and query concepts. Working through this example is a microcosm of what a Neo4j Pre-sales System Engineer could be expected to perform as part of a technical positioning the value of Neo4j to a prospect. The goal is to generate excitement for the power of the Neo4j graph database, while exposing core technologies and concepts. The exercise involves loading data into a graph structure and running some simple Cypher queries. It does not represent the breadth and complexity of a typical prospect or client Neo4j implementation. Take a look at the topics in the Neo4j downloads page and the different deployment and operational options (documentation) to get a sense of the Neo4j technologies and the environments that it operates in.

Exercise Description

To create a graph based on investments in individual stocks and mutual funds. The demo data set can easily answer questions such as:

- Which accounts hold a particular stock?
- What is value of my total holdings/account/individual investment on a specificd date?
- Which of my accounts hold individual stocks?

Suggested Resources

For this exercise, the following resources are suggested:

- If you're not familiar with the Neo4j graph database and the concepts behind implementing a graph, then take some of the free self-paced, short online trainings available from the Neo4j GraphAcademy.
- You'll be using the Neo4j Browser for development. The Neo4j Browser Development Guide will get you familiar with this easy-to-use tool.
- Neo4j Documentation:
    - Where to place the .csv files if using Neo4j Desktop (suggested technology to use).
    - Using the Cypher language LOAD CSV command. References for the LOAD CSV Cypher command can be found in the documentation and developer guide materials.
- Cypher Reference Card. Major Cypher commands used for this exercise include LOAD CSV, CREATE, MATCH, and potentially MERGE and UNION / CALL.
- The Neo4j community website to ask questions and look for techniques.

What To Do with Your Results

Expect to be asked to present your results in a form that will be discussed as you complete the exercise. It is advised to keep any code to be able to recreate the database and run the queries built, typically by saving your Cypher statements.

General Tips On Working With The Graph

- There are multiple ways to start working with the Neo4j database, including the Neo4j Desktop (download), a local server install (download), the Neo4j Sandbox (time limited cloud instance), the Neo4j Aura Free Tier (blank cloud instance), or by using Neo4j Docker / Kubernetes environments. The suggested option is to use the Neo4j Desktop.
- The Neo4j Developer Guides augment the Neo4j documentation on how to use Neo4j, along with the short form Neo4j Graph Academy online training courses.
- Use the arrows.app to visually experiment with creating graph models. The model can be exported as Cypher and run against a Neo4j database. This is not required to run the exercise but some say it helps them formulate their design.
- Reach out to your Neo4j contact with any questions or concerns.
- Relational database design thinking is going to cause issues as relationships are instantiated first class citizens in a graph as compared to second class, runtime generated joins supported by primary and foreign keys. An example of relational design is if you see yourself creating queries in a pattern like the one below where *two data items are related, or joined at runtime*:

```
// Query pattern that mimics a RDBMS join instead of a graph relationship / traversal
MATCH (s:Stock)
MATCH (d:DailyClose) WHERE d.ticker = s.ticker
RETURN s, d
```

Tips About the Data

1. All data is ingested as strings when using the LOAD CSV command unless explicitly converted to another data type.
2. Neo4j is a schema-less or schema-last NoSQL database. Input data can be used to define the structure of the graph and be used properties on nodes and relationships.
3. Key file data interrelationships:
   a. account.csv records link accounts to customers.
   b. account_purchases.csv records are the individual stocks and funds a customer holds. NOTE: there can be multiple purchases for a stock or fund.
   c. funds.csv contains fund information.
   d. fund_holdings.csv records are the individual stocks or funds a fund holds by ticker.
   e. daily_close.csv contains the daily date, volume, and prices for by ticker symbol for stocks and funds.

Data Set

The data set is based on customers who hold various accounts (401(k), 529, individual, etc.) and five Vanguard funds and their top 10 holdings available via public domain. The ten-year historical quotes were pulled from the Nasdaq website.

Input File Examples

customers.csv

customer_id	owner_name	address1	address2	city	state	zip
AJ123	Alex Jones	12 1st St.		Boston	MA	2432

accounts.csv

customer_id	account_id	owner_name	address1	address2	city	state	zip	account_type	channel
AJ123	123456	Alex Jones	12 1st St.		Boston	MA	2432	401k	Corporate

account_purchases.csv

account_id	ticker	number_of_shares	purchase_date
123456	VMGAX	200	1/12/15
923457	MSFT	1000	01/12/2011

funds.csv

fund_name	ticker	assets	manager	inception_date	company	expense_ratio
Vanguard Dividend Growth Fund	VDIGX	32570000000	Donald J. Kilbride	3/15/92	Vanguard	0.26

fund_holdings.csv

fund_ticker	holding_company	holding_ticker	percent
VDIGX	Nike Inc.	NKE	3.89

daily_close.csv

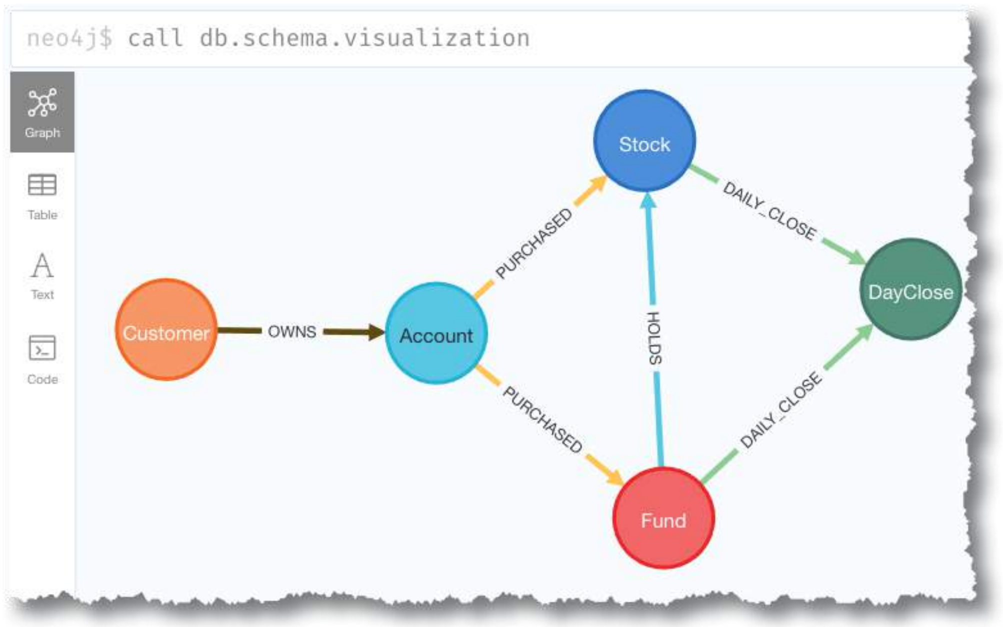
ticker	date	close	volume	open	high	low
GOOGL	6/22/18	1169.29	1710355	1171.495	1175	1159.65

stock_ticker.csv

ticker	holding_company
AAPL	Apple Inc.

Graph Data Model

The following graph schema diagram reflects how this data should be modeled together. The CSV files also contain additional attributes to be used as node or relationship properties.



```
neo4j$ call db.schema.visualization
```

Screen shot is from the Neo4j Browser

INSTRUCTIONS

1. Load Data

NOTE: Theses instructions are for using a local Neo4j database.

- With your Neo4j environment, place provided .csv files in appropriate directory for import.
- Using Neo4j Browser create and run load scripts to import data from each .csv file.
- Please save your scripts in a document, for our later review.
- As an example, the load scripts for Customers and Accounts are provided, including thestatement to establish relationships between Customers and Accounts (1a, 1b, 1c):

1a)
```
// 1a – Load customers
LOAD CSV WITH HEADERS FROM 'file:///customers.csv' AS row
CREATE (c:Customer)
SET c += row
```

1b)
```
// 1b – Load accounts
LOAD CSV WITH HEADERS FROM 'file:///accounts.csv' AS row
CREATE (:Account {
        account_id : row.account_id,
        customer_id : row.customer_id,
        account_type : row.account_type,
        channel:row.channel })
```

1c)
```
// 1c – Link customer to accounts
LOAD CSV WITH HEADERS FROM 'file:///accounts.csv' AS row
MATCH (account : Account {account_id : row.account_id })
MATCH (customer:Customer) WHERE customer.customer_id = row.customer_id
CREATE (customer)-[:HAS]->(account)
```

To be completed:

1d) Load funds data, similar to 1a or 1b.

1e) Create script similar to 1c to establish relationships betweenAccounts and Funds.

1f) Load Ticker data.

1g) Establish relationships for Holdings.

```
    -   e.g. (account)-[:PURCHASED]->(Stock)
             (account)-[:PURCHASED]->(Fund)
    -  include quantity & purchase date on relationship.
```

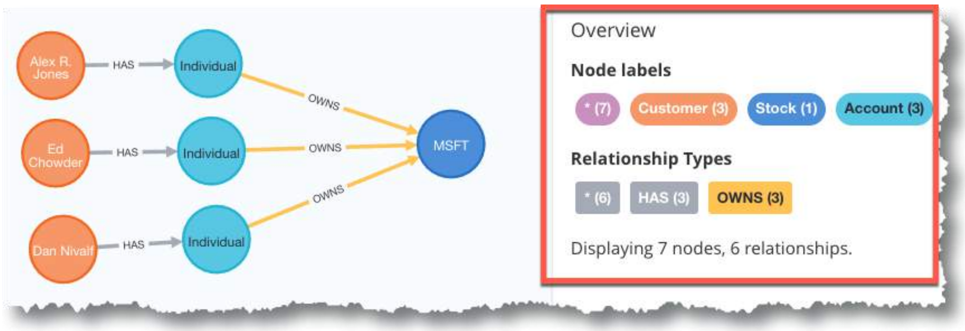
1h) Establish relationships for Mutual Funds and Holdings, with Percentage of Assets as a relationship property.

1i) Load historical daily close data, link to individual stocks int: Employ periodic commit to facilitate higher volume load**.**

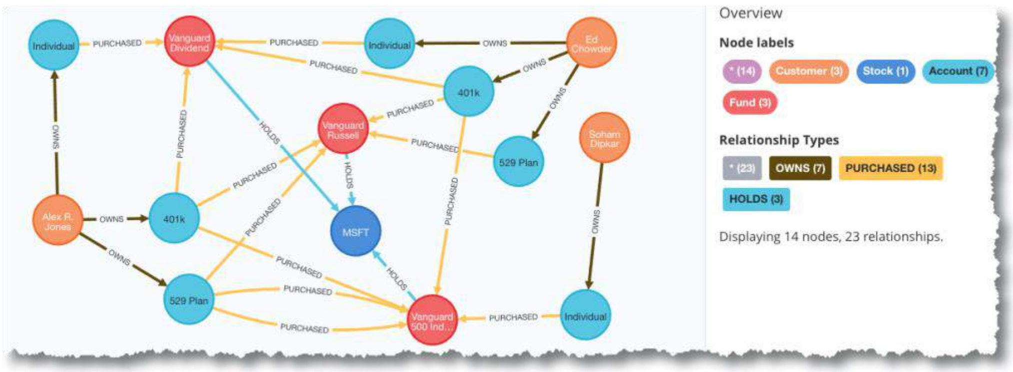
2. Create Cypher Queries:

Use Cypher queries to answer the following questions. Diagrams are provided to show sample results. The first two queries you can verify your results by looking at the overview pane in the Neo4j Browser.

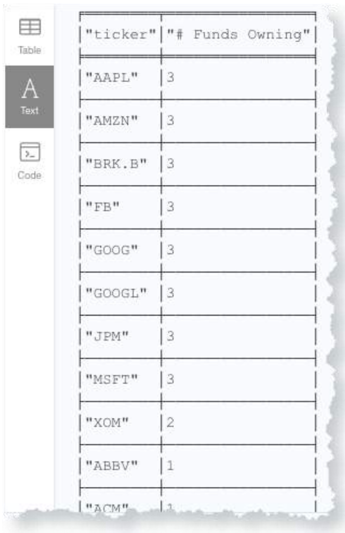
2a) Find all Accounts that own MSFT stock directly through an individual account.



2b) Find all Accounts that hold MSFT through a mutual fund.

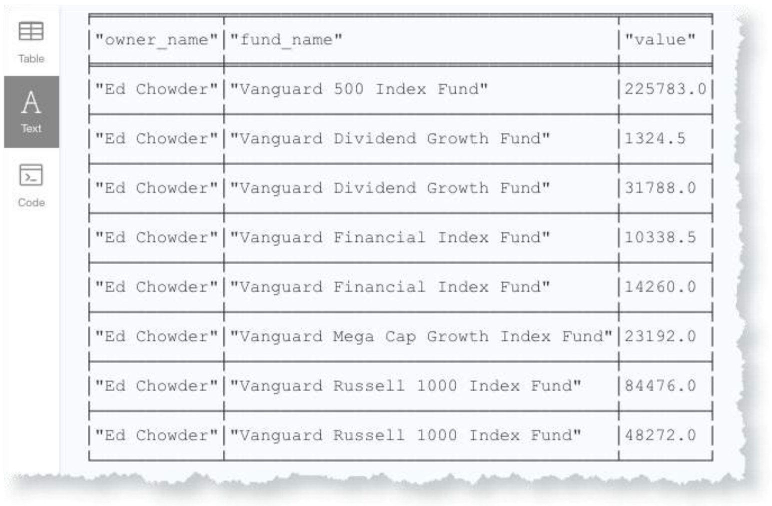


2c) Return a count of the number of times a fund holds a stock, sorted in descending count order.



```
│"ticker"│"# Funds Owning"│
│"AAPL"  │3               │
│"AMZN"  │3               │
│"BRK.B" │3               │
│"FB"    │3               │
│"GOOG"  │3               │
│"GOOGL" │3               │
│"JPM"   │3               │
│"MSFT"  │3               │
│"XOM"   │2               │
│"ABBV"  │1               │
│"ACM"   │1               │
```

2d) Return the value of mutual fund holdings owned by ‘Ed Chowder’ close date = 5/15/18. Calculate the value of the fund holdings and order by fund name.

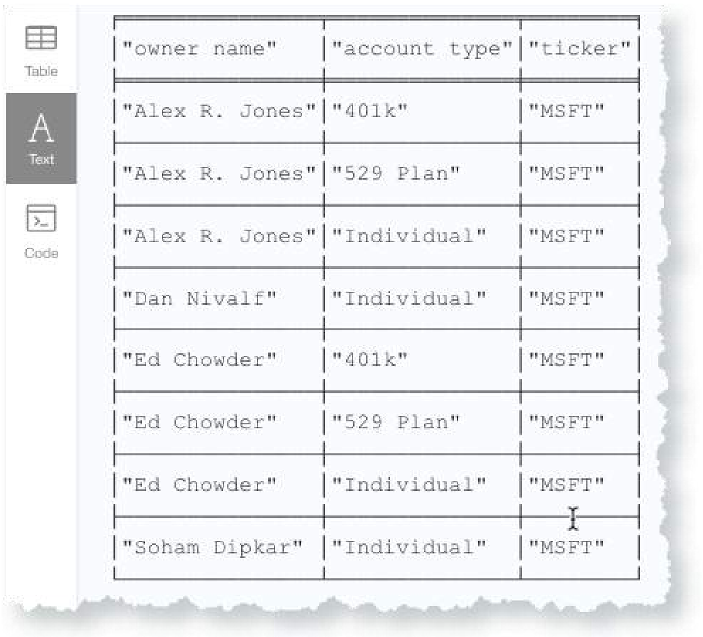


```
╒═════════════╤═══════════════════════════════════╤═════════╕
│"owner_name" │"fund_name"                        │"value"  │
╞═════════════╪═══════════════════════════════════╪═════════╡
│"Ed Chowder" │"Vanguard 500 Index Fund"          │225783.0 │
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Dividend Growth Fund"    │1324.5   │
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Dividend Growth Fund"    │31788.0  │
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Financial Index Fund"    │10338.5  │
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Financial Index Fund"    │14260.0  │
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Mega Cap Growth Index Fund"│23192.0│
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Russell 1000 Index Fund" │84476.0  │
├─────────────┼───────────────────────────────────┼─────────┤
│"Ed Chowder" │"Vanguard Russell 1000 Index Fund" │48272.0  │
└─────────────┴───────────────────────────────────┴─────────┘
```

3. Optional Bonus Cypher Queries

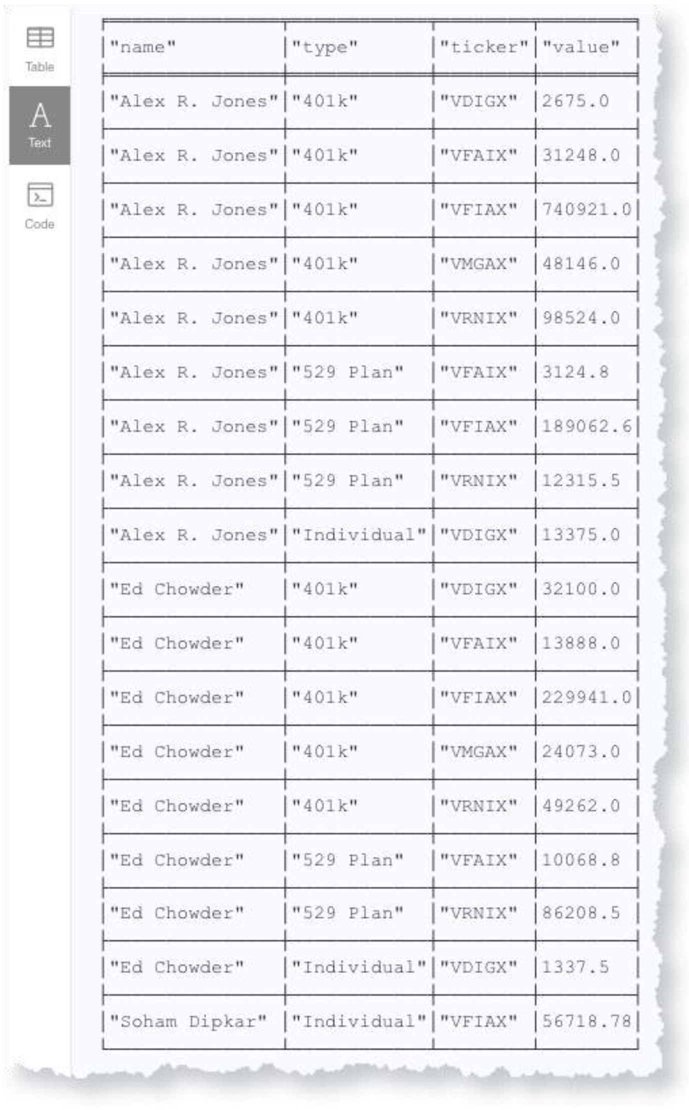
These questions are optional and involve slightly more complicated Cypher queries. The objective is only to help investigate how you approach building your model and what you’ve been able to absorb about applying the Cypher pattern based query language.

3a) Return account owner name(s) and account type(s) that own MSFT stock directly through an individual account or through a mutual fund. You do not have to count the number of account types a person owns (e.g. owner has more than one account type “Individual”)



```
╒═══════════════╤══════════════╤════════╕
│"owner name"   │"account type"│"ticker"│
╞═══════════════╪══════════════╪════════╡
│"Alex R. Jones"│"401k"        │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Alex R. Jones"│"529 Plan"    │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Alex R. Jones"│"Individual"  │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Dan Nivalf"   │"Individual"  │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Ed Chowder"   │"401k"        │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Ed Chowder"   │"529 Plan"    │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Ed Chowder"   │"Individual"  │"MSFT"  │
├───────────────┼──────────────┼────────┤
│"Soham Dipkar" │"Individual"  │"MSFT"  │
└───────────────┴──────────────┴────────┘
```

3b) Return account owner name(s), account type(s), the fund or stock they own and total the value for the last day in the daily trading data. Do not hard code the last day in the query.



```
╒═══════════════╤════════════╤════════╤════════╕
│"name"         │"type"      │"ticker"│"value" │
╞═══════════════╪════════════╪════════╪════════╡
│"Alex R. Jones"│"401k"      │"VDIGX" │2675.0  │
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"401k"      │"VFAIX" │31248.0 │
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"401k"      │"VFIAX" │740921.0│
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"401k"      │"VMGAX" │48146.0 │
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"401k"      │"VRNIX" │98524.0 │
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"529 Plan"  │"VFAIX" │3124.8  │
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"529 Plan"  │"VFIAX" │189062.6│
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"529 Plan"  │"VRNIX" │12315.5 │
├───────────────┼────────────┼────────┼────────┤
│"Alex R. Jones"│"Individual"│"VDIGX" │13375.0 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"401k"      │"VDIGX" │32100.0 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"401k"      │"VFAIX" │13888.0 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"401k"      │"VFIAX" │229941.0│
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"401k"      │"VMGAX" │24073.0 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"401k"      │"VRNIX" │49262.0 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"529 Plan"  │"VFAIX" │10068.8 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"529 Plan"  │"VRNIX" │86208.5 │
├───────────────┼────────────┼────────┼────────┤
│"Ed Chowder"   │"Individual"│"VDIGX" │1337.5  │
├───────────────┼────────────┼────────┼────────┤
│"Soham Dipkar" │"Individual"│"VFIAX" │56718.78│
└───────────────┴────────────┴────────┴────────┘
```

3c) Any other interesting queries or observations about the model you built and the types of queries that it entails.