Lesson 6: Expanding the SEC Knowledge Graph ¶

Note: This notebook takes about 30 seconds to be ready to use. Please wait until the "Kernel starting, please wait..." message clears from the top of the notebook before running any cells. You may start the video while you wait.

Import packages and set up Neo4j

In [1]:
```
from dotenv import load_dotenv
import os
import textwrap

# Langchain
from langchain_community.graphs import Neo4jGraph
from langchain_community.vectorstores import Neo4jVector
from langchain_openai import OpenAIEmbeddings
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import RetrievalQAWithSourcesChain
from langchain_openai import ChatOpenAI

# Warning control
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Load from environment
load_dotenv('.env', override=True)
NEO4J_URI = os.getenv('NEO4J_URI')
NEO4J_USERNAME = os.getenv('NEO4J_USERNAME')
NEO4J_PASSWORD = os.getenv('NEO4J_PASSWORD')
NEO4J_DATABASE = os.getenv('NEO4J_DATABASE') or 'neo4j'

# Global constants
VECTOR_INDEX_NAME = 'form_10k_chunks'
VECTOR_NODE_LABEL = 'Chunk'
VECTOR_SOURCE_PROPERTY = 'text'
VECTOR_EMBEDDING_PROPERTY = 'textEmbedding'
```

In [3]:
```
kg = Neo4jGraph(
    url=NEO4J_URI,
    username=NEO4J_USERNAME,
    password=NEO4J_PASSWORD,
    database=NEO4J_DATABASE
)
```

Read the collection of Form 13s

- Investment management firms must report on their investments in companies to the SEC by filing a document called **Form 13**
- You'll load a collection of Form 13 for managers that have invested in NetApp

- You can check out the CSV file by navigating to the data directory using the File menu at the top of the notebook

In [4]:
```
import csv

all_form13s = []

with open('./data/form13.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    for row in csv_reader: # each row will be a dictionary
      all_form13s.append(row)
```

- Look at the contents of the first 5 Form 13s

In [5]:
```
all_form13s[0:5]
```

```
[{'source': 'https://sec.gov/Archives/edgar/data/1000275/0001140361-23-03
9575.txt',
  'managerCik': '1000275',
  'managerAddress': 'ROYAL BANK PLAZA, 200 BAY STREET, TORONTO, A6, M5J2J
5',
  'managerName': 'Royal Bank of Canada',
  'reportCalendarOrQuarter': '2023-06-30',
  'cusip6': '64110D',
  'cusip': '64110D104',
  'companyName': 'NETAPP INC',
  'value': '64395000000.0',
  'shares': '842850'},
 {'source': 'https://sec.gov/Archives/edgar/data/1002784/0001387131-23-00
9542.txt',
  'managerCik': '1002784',
  'managerAddress': '1875 Lawrence Street, Suite 300, Denver, CO, 80202-1
805',
  'managerName': 'SHELTON CAPITAL MANAGEMENT',
  'reportCalendarOrQuarter': '2023-06-30',
  'cusip6': '64110D',
  'cusip': '64110D104',
  'companyName': 'NETAPP INC',
  'value': '2989085000.0',
  'shares': '39124'},
 {'source': 'https://sec.gov/Archives/edgar/data/1007280/0001007280-23-00
0008.txt',
  'managerCik': '1007280',
  'managerAddress': '277 E TOWN ST, COLUMBUS, OH, 43215',
  'managerName': 'PUBLIC EMPLOYEES RETIREMENT SYSTEM OF OHIO',
  'reportCalendarOrQuarter': '2023-06-30',
  'cusip6': '64110D',
  'cusip': '64110D104',
  'companyName': 'Netapp Inc',
  'value': '8170000.0',
  'shares': '106941'},
 {'source': 'https://sec.gov/Archives/edgar/data/1007399/0001007399-23-00
0004.txt',
  'managerCik': '1007399',
  'managerAddress': '150 WEST MAIN STREET, SUITE 1700, NORFOLK, VA, 2351
0',
  'managerName': 'WILBANKS SMITH & THOMAS ASSET MANAGEMENT LLC',
  'reportCalendarOrQuarter': '2023-06-30',
  'cusip6': '64110D',
  'cusip': '64110D104',
  'companyName': 'NETAPP INC',
  'value': '505539000.0',
  'shares': '6617'},
 {'source': 'https://sec.gov/Archives/edgar/data/1008894/0001172661-23-00
3025.txt',
  'managerCik': '1008894',
  'managerAddress': '250 Park Avenue South, Suite 250, Winter Park, FL, 3
2789',
  'managerName': 'DEPRINCE RACE & ZOLLO INC',
  'reportCalendarOrQuarter': '2023-06-30',
  'cusip6': '64110D',
  'cusip': '64110D104',
  'companyName': 'NETAPP INC',
  'value': '24492389000.0',
  'shares': '320581'}]
```

In [6]:
```
len(all_form13s)
```

```
561
```

Create company nodes in the graph

- Use the companies identified in the Form 13s to create `Company` nodes
- For now, there is only one company - NetApp

In [7]:
```
# work with just the first form fow now
first_form13 = all_form13s[0]

cypher = """
MERGE (com:Company {cusip6: $cusip6})
  ON CREATE
    SET com.companyName = $companyName,
        com.cusip = $cusip
"""

kg.query(cypher, params={
    'cusip6':first_form13['cusip6'],
    'companyName':first_form13['companyName'],
    'cusip':first_form13['cusip']
})
```

```
[]
```

In [8]:
```
cypher = """
MATCH (com:Company)
RETURN com LIMIT 1
"""

kg.query(cypher)
```

```
[{'com': {'cusip': '64110D104',
   'companyName': 'NETAPP INC',
   'cusip6': '64110D'}}]
```

- Update the company name to match Form 10-K

In [9]:
```
cypher = """
  MATCH (com:Company), (form:Form)
    WHERE com.cusip6 = form.cusip6
  RETURN com.companyName, form.names
"""

kg.query(cypher)
```

```
[{'com.companyName': 'NETAPP INC', 'form.names': ['Netapp Inc', 'NETAPP I
NC']}]
```

In [10]:
```
cypher = """
  MATCH (com:Company), (form:Form)
    WHERE com.cusip6 = form.cusip6
  SET com.names = form.names
"""

kg.query(cypher)
```

```
[]
```

- Create a `FILED` relationship between the company and the Form-10K node

In [11]:
```
kg.query("""
  MATCH (com:Company), (form:Form)
    WHERE com.cusip6 = form.cusip6
  MERGE (com)-[:FILED]->(form)
""")
```

```
[]
```

Create manager nodes

- Create a `manager` node for companies that have filed a Form 13 to report their investment in NetApp
- Start with the single manager who filed the first Form 13 in the list

In [12]:
```
cypher = """
  MERGE (mgr:Manager {managerCik: $managerParam.managerCik})
    ON CREATE
        SET mgr.managerName = $managerParam.managerName,
            mgr.managerAddress = $managerParam.managerAddress
"""

kg.query(cypher, params={'managerParam': first_form13})
```

```
[]
```

In [13]:
```
kg.query("""
  MATCH (mgr:Manager)
  RETURN mgr LIMIT 1
""")
```

```
[{'mgr': {'managerCik': '1000275',
   'managerAddress': 'ROYAL BANK PLAZA, 200 BAY STREET, TORONTO, A6, M5J2
J5',
   'managerName': 'Royal Bank of Canada'}}]
```

- Create a uniquness constraint to avoid duplicate managers

In [14]:
```
kg.query("""
CREATE CONSTRAINT unique_manager
  IF NOT EXISTS
  FOR (n:Manager)
  REQUIRE n.managerCik IS UNIQUE
""")
```

```
[]
```

- Create a fulltext index of manager names to enable text search

In [15]:
```
kg.query("""
CREATE FULLTEXT INDEX fullTextManagerNames
  IF NOT EXISTS
  FOR (mgr:Manager)
  ON EACH [mgr.managerName]
""")
```

```
[]
```

In [16]:
```
kg.query("""
  CALL db.index.fulltext.queryNodes("fullTextManagerNames",
      "royal bank") YIELD node, score
  RETURN node.managerName, score
""")
```

```
[{'node.managerName': 'Royal Bank of Canada', 'score': 0.261529147624969
5}]
```

- Create nodes for all companies that filed a Form 13

In [17]:
```
cypher = """
  MERGE (mgr:Manager {managerCik: $managerParam.managerCik})
    ON CREATE
        SET mgr.managerName = $managerParam.managerName,
            mgr.managerAddress = $managerParam.managerAddress
"""
# Loop through all Form 13s
for form13 in all_form13s:
  kg.query(cypher, params={'managerParam': form13 })
```

In [18]:
```
kg.query("""
    MATCH (mgr:Manager)
    RETURN count(mgr)
""")
```

```
[{'count(mgr)': 561}]
```

Create relationships between managers and companies

- Match companies with managers based on data in the Form 13
- Create an `OWNS_STOCK_IN` relationship between the manager and the company

- Start with the single manager who filed the first Form 13 in the list

In [19]:
```
cypher = """
  MATCH (mgr:Manager {managerCik: $investmentParam.managerCik}),
        (com:Company {cusip6: $investmentParam.cusip6})
  RETURN mgr.managerName, com.companyName, $investmentParam as invest
"""

kg.query(cypher, params={
    'investmentParam': first_form13
})
```

```
[{'mgr.managerName': 'Royal Bank of Canada',
  'com.companyName': 'NETAPP INC',
  'investment': {'shares': '842850',
   'source': 'https://sec.gov/Archives/edgar/data/1000275/0001140361-23-0
39575.txt',
   'managerName': 'Royal Bank of Canada',
   'managerAddress': 'ROYAL BANK PLAZA, 200 BAY STREET, TORONTO, A6, M5J2
J5',
   'value': '64395000000.0',
   'cusip6': '64110D',
   'cusip': '64110D104',
   'reportCalendarOrQuarter': '2023-06-30',
   'companyName': 'NETAPP INC',
   'managerCik': '1000275'}}]
```

In [20]:
```
cypher = """
MATCH (mgr:Manager {managerCik: $ownsParam.managerCik}),
        (com:Company {cusip6: $ownsParam.cusip6})
MERGE (mgr)-[owns:OWNS_STOCK_IN {
    reportCalendarOrQuarter: $ownsParam.reportCalendarOrQuarter
}]->(com)
ON CREATE
    SET owns.value  = toFloat($ownsParam.value),
        owns.shares = toInteger($ownsParam.shares)
RETURN mgr.managerName, owns.reportCalendarOrQuarter, com.companyName
"""

kg.query(cypher, params={ 'ownsParam': first_form13 })
```

```
[{'mgr.managerName': 'Royal Bank of Canada',
  'owns.reportCalendarOrQuarter': '2023-06-30',
  'com.companyName': 'NETAPP INC'}]
```

In [21]:
```
kg.query("""
MATCH (mgr:Manager {managerCik: $ownsParam.managerCik})
-[owns:OWNS_STOCK_IN]->
        (com:Company {cusip6: $ownsParam.cusip6})
RETURN owns { .shares, .value }
""", params={ 'ownsParam': first_form13 })
```

```
[{'owns': {'shares': 842850, 'value': 64395000000.0}}]
```

- Create relationships between all of the managers who filed Form 13s and the company

In [22]:
```
cypher = """
MATCH (mgr:Manager {managerCik: $ownsParam.managerCik}),
        (com:Company {cusip6: $ownsParam.cusip6})
MERGE (mgr)-[owns:OWNS_STOCK_IN {
    reportCalendarOrQuarter: $ownsParam.reportCalendarOrQuarter
    }]->(com)
  ON CREATE
    SET owns.value  = toFloat($ownsParam.value),
        owns.shares = toInteger($ownsParam.shares)
"""

#loop through all Form 13s
for form13 in all_form13s:
  kg.query(cypher, params={'ownsParam': form13 })
```

In [23]:
```
cypher = """
  MATCH (:Manager)-[owns:OWNS_STOCK_IN]->(:Company)
  RETURN count(owns) as investments
"""

kg.query(cypher)
```

```
[{'investments': 561}]
```

In [24]:
```
kg.refresh_schema()
print(textwrap.fill(kg.schema, 60))
```

```
Node properties are the following: Chunk {textEmbedding:
LIST, f10kItem: STRING, chunkSeqId: INTEGER, text: STRING,
cik: STRING, cusip6: STRING, names: LIST, formId: STRING,
source: STRING, chunkId: STRING},Form {cusip6: STRING,
names: LIST, formId: STRING, source: STRING},Company
{cusip6: STRING, names: LIST, companyName: STRING, cusip:
STRING},Manager {managerName: STRING, managerCik: STRING,
managerAddress: STRING} Relationship properties are the
following: SECTION {f10kItem: STRING},OWNS_STOCK_IN {shares:
INTEGER, reportCalendarOrQuarter: STRING, value: FLOAT} The
relationships are the following: (:Chunk)-[:NEXT]-
>(:Chunk),(:Chunk)-[:PART_OF]->(:Form),(:Form)-[:SECTION]-
>(:Chunk),(:Company)-[:FILED]->(:Form),(:Manager)-
[:OWNS_STOCK_IN]->(:Company)
```

Determine the number of investors

- Start by finding a form 10-K chunk, and save to use in subsequent queries

In [25]:
```
cypher = """
    MATCH (chunk:Chunk)
    RETURN chunk.chunkId as chunkId LIMIT 1
    """

chunk_rows = kg.query(cypher)
print(chunk_rows)
```

```
[{'chunkId': '0000950170-23-027948-item1-chunk0000'}]
```

In [26]:
```
chunk_first_row = chunk_rows[0]
print(chunk_first_row)
```

```
{'chunkId': '0000950170-23-027948-item1-chunk0000'}
```

In [27]:
```
ref_chunk_id = chunk_first_row['chunkId']
ref_chunk_id
```

```
'0000950170-23-027948-item1-chunk0000'
```

- Build up path from Form 10-K chunk to companies and managers

In [28]:
```
cypher = """
    MATCH (:Chunk {chunkId: $chunkIdParam})-[:PART_OF]->(f:Form)
    RETURN f.source
    """

kg.query(cypher, params={'chunkIdParam': ref_chunk_id})
```

```
[{'f.source': 'https://www.sec.gov/Archives/edgar/data/1002047/0000950170
23027948/0000950170-23-027948-index.htm'}]
```

In [29]:
```
cypher = """
MATCH (:Chunk {chunkId: $chunkIdParam})-[:PART_OF]->(f:Form),
    (com:Company)-[:FILED]->(f)
RETURN com.companyName as name
"""

kg.query(cypher, params={'chunkIdParam': ref_chunk_id})
```

```
[{'name': 'NETAPP INC'}]
```

In [30]:
```
cypher = """
MATCH (:Chunk {chunkId: $chunkIdParam})-[:PART_OF]->(f:Form),
        (com:Company)-[:FILED]->(f),
        (mgr:Manager)-[:OWNS_STOCK_IN]->(com)
RETURN com.companyName,
        count(mgr.managerName) as numberOfinvestors
LIMIT 1
"""

kg.query(cypher, params={
    'chunkIdParam': ref_chunk_id
})
```

```
[{'com.companyName': 'NETAPP INC', 'numberOfinvestors': 561}]
```

Use queries to build additional context for LLM

- Create sentences that indicate how much stock a manager has invested in a company

In [31]:
```
cypher = """
    MATCH (:Chunk {chunkId: $chunkIdParam})-[:PART_OF]->(f:Form),
        (com:Company)-[:FILED]->(f),
        (mgr:Manager)-[owns:OWNS_STOCK_IN]->(com)
    RETURN mgr.managerName + " owns " + owns.shares +
        " shares of " + com.companyName +
        " at a value of $" +
        apoc.number.format(toInteger(owns.value)) AS text
    LIMIT 10
    """
kg.query(cypher, params={
    'chunkIdParam': ref_chunk_id
})
```

```
[{'text': 'CSS LLC/IL owns 12500 shares of NETAPP INC at a value of $955,
000,000'},
 {'text': 'BOKF, NA owns 40774 shares of NETAPP INC at a value of $3,115,
134,000'},
 {'text': 'BANK OF NOVA SCOTIA owns 18676 shares of NETAPP INC at a value
of $1,426,847,000'},
 {'text': 'Jefferies Financial Group Inc. owns 23200 shares of NETAPP INC
at a value of $1,772,480,000'},
 {'text': 'DEUTSCHE BANK AG\\ owns 929854 shares of NETAPP INC at a value
of $71,040,845,000'},
 {'text': 'TORONTO DOMINION BANK owns 183163 shares of NETAPP INC at a va
lue of $13,984,000'},
 {'text': 'STATE BOARD OF ADMINISTRATION OF FLORIDA RETIREMENT SYSTEM own
s 265756 shares of NETAPP INC at a value of $20,303,759,000'},
 {'text': 'NISA INVESTMENT ADVISORS, LLC owns 67848 shares of NETAPP INC
at a value of $5,183,587,000'},
 {'text': 'ONTARIO TEACHERS PENSION PLAN BOARD owns 7290 shares of NETAPP
INC at a value of $556,956,000'},
 {'text': 'STATE STREET CORP owns 9321206 shares of NETAPP INC at a value
of $712,140,138,000'}]
```

In [32]:
```
results = kg.query(cypher, params={
    'chunkIdParam': ref_chunk_id
})
print(textwrap.fill(results[0]['text'], 60))
```

```
CSS LLC/IL owns 12500 shares of NETAPP INC at a value of
$955,000,000
```

- Create a plain Question Answer chain
- Similarity search only, no augmentation by Cypher Query

In [33]:
```
vector_store = Neo4jVector.from_existing_graph(
    embedding=OpenAIEmbeddings(),
    url=NEO4J_URI,
    username=NEO4J_USERNAME,
    password=NEO4J_PASSWORD,
    index_name=VECTOR_INDEX_NAME,
    node_label=VECTOR_NODE_LABEL,
    text_node_properties=[VECTOR_SOURCE_PROPERTY],
    embedding_node_property=VECTOR_EMBEDDING_PROPERTY,
)
# Create a retriever from the vector store
retriever = vector_store.as_retriever()

# Create a chatbot Question & Answer chain from the retriever
plain_chain = RetrievalQAWithSourcesChain.from_chain_type(
    ChatOpenAI(temperature=0),
    chain_type="stuff",
    retriever=retriever
)
```

- Create a second QA chain
- Augment similarity search using sentences found by the investment query above

In [34]:
```
investment_retrieval_query = """
MATCH (node)-[:PART_OF]->(f:Form),
    (f)<-[:FILED]-(com:Company),
    (com)<-[owns:OWNS_STOCK_IN]-(mgr:Manager)
WITH node, score, mgr, owns, com
    ORDER BY owns.shares DESC LIMIT 10
WITH collect (
    mgr.managerName +
    " owns " + owns.shares +
    " shares in " + com.companyName +
    " at a value of $" +
    apoc.number.format(toInteger(owns.value)) + "."
) AS investment_statements, node, score
RETURN apoc.text.join(investment_statements, "\n") +
    "\n" + node.text AS text,
    score,
    {
      source: node.source
    } as metadata
"""
```

In [35]:
```
vector_store_with_investment = Neo4jVector.from_existing_index(
    OpenAIEmbeddings(),
    url=NEO4J_URI,
    username=NEO4J_USERNAME,
    password=NEO4J_PASSWORD,
    database="neo4j",
    index_name=VECTOR_INDEX_NAME,
    text_node_property=VECTOR_SOURCE_PROPERTY,
    retrieval_query=investment_retrieval_query,
)

# Create a retriever from the vector store
retriever_with_investments = vector_store_with_investment.as_retrieve

# Create a chatbot Question & Answer chain from the retriever
investment_chain = RetrievalQAWithSourcesChain.from_chain_type(
    ChatOpenAI(temperature=0),
    chain_type="stuff",
    retriever=retriever_with_investments
)
```

- Compare the outputs!

In [36]:
```
question = "In a single sentence, tell me about Netapp."
```

In [37]:
```
plain_chain(
    {"question": question},
    return_only_outputs=True,
)
```

```
{'answer': 'NetApp is a global cloud-led, data-centric software company t
hat provides customers the freedom to manage applications and data across
hybrid multicloud environments. \n',
 'sources': 'https://www.sec.gov/Archives/edgar/data/1002047/000095017023
027948/0000950170-23-027948-index.htm'}
```

In [38]:
```
investment_chain(
    {"question": question},
    return_only_outputs=True,
)
```

```
{'answer': 'NetApp is a global cloud-led, data-centric software company t
hat provides customers with the freedom to manage applications and data a
cross hybrid multicloud environments. \n',
 'sources': 'https://www.sec.gov/Archives/edgar/data/1002047/000095017023
027948/0000950170-23-027948-index.htm'}
```

- The LLM didn't make use of the investor information since the question didn't ask about investors
- Change the question and ask again

In [39]:
```
question = "In a single sentence, tell me about Netapp investors."
```

In [40]:
```
plain_chain(
    {"question": question},
    return_only_outputs=True,
)
```

```
{'answer': 'Netapp investors are diverse and include global enterprises,
local businesses, and government installations who look to NetApp and the
ir ecosystem of partners to maximize the business value of their IT and c
loud investments.\n',
 'sources': 'https://www.sec.gov/Archives/edgar/data/1002047/000095017023
027948/0000950170-23-027948-index.htm'}
```

In [41]:
```
investment_chain(
    {"question": question},
    return_only_outputs=True,
)
```

```
{'answer': 'Netapp investors include VANGUARD GROUP INC, BlackRock Inc.,
and PRIMECAP MANAGEMENT CO/CA/.\n',
 'sources': 'https://www.sec.gov/Archives/edgar/data/1002047/000095017023
027948/0000950170-23-027948-index.htm'}
```

Try for yourself

- Try changing the query above to retrieve other information
- Try asking different questions
- Note, if you change the Cypher query, you'll need to reset the retriever and QA chain