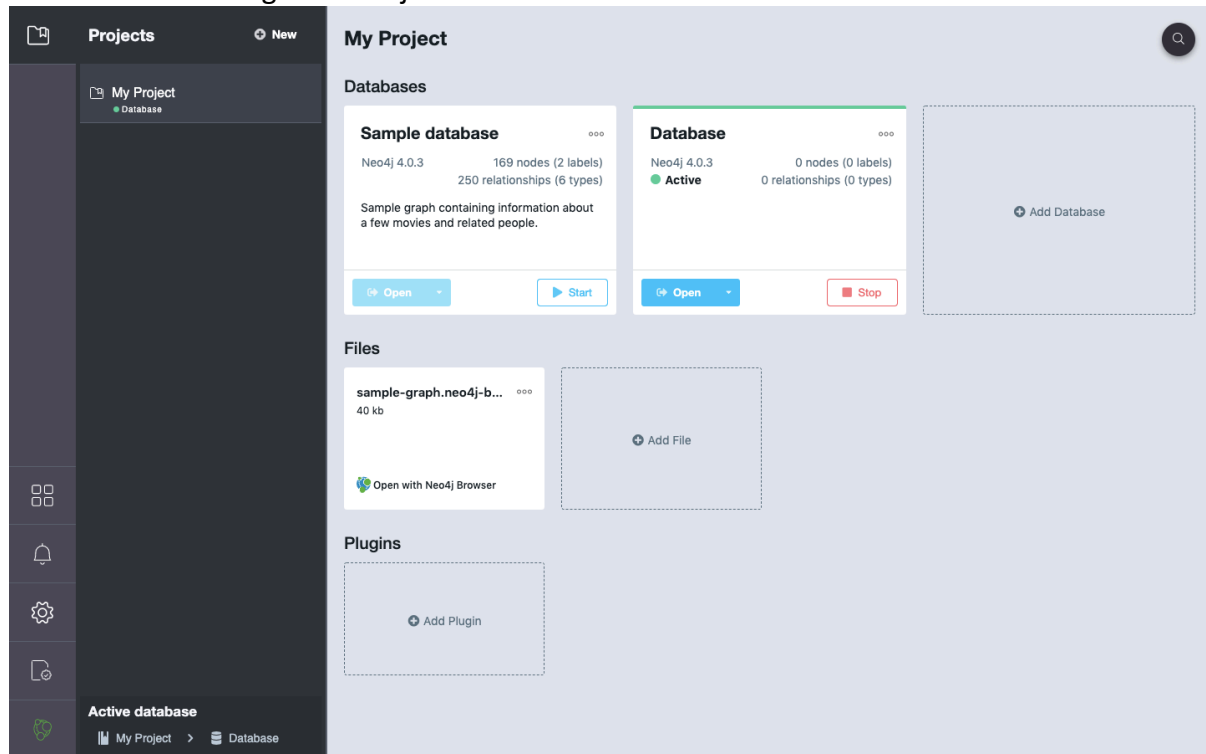


## 1. Downloading the neo4j server



2.1. What are some pros and cons of Graph Databases? Please give an example of when you would want to use a graph database, when you would want to use a traditional relational database.

Pros:

really fast queries when you are looking for relationships between nodes

really fast to traverse nodes

Can represent multiple dimensions

Cons:

Inappropriate for transactional information, like accounting records where relationships between records are simpler

Harder to do summing queries and max queries efficiently - counting queries not harder

Usually need to learn a new query language like CYPHER

Fewer vendors to choose from, and smaller user base, so harder to get support when you run into issues

Source: <https://www.quora.com/What-are-the-pros-and-cons-of-using-a-graph-database>

When to use a graph database?

When I need to build a recommendation engine, I will choose to use a graph database. It is important to know the relationship between each user for good recommendations.

When to use a relational database?

If join querying is used frequently, it is a good time to apply a relational database. I will use a relational database to store member information because it is easier for sales and IT to join some member data from different tables to meet their needs.

2.2. Please describe the following terms (2-3 sentences/bullets each):

**Nodes:** Nodes represent entities or instances such as people, businesses, accounts, or any other item to be tracked. They are roughly the equivalent of a record, relation, or row in a relational database, or a document in a document-store database. (source:

[https://en.wikipedia.org/wiki/Graph\\_database#Labeled-property\\_graph](https://en.wikipedia.org/wiki/Graph_database#Labeled-property_graph))

**Labels:** A label is a named graph construct that is used to group nodes into sets; all nodes labeled with the same label belongs to the same set. Many database queries can work with these sets instead of the whole graph, making queries easier to write and more efficient to execute. A node may be labeled with any number of labels, including none, making labels an optional addition to the graph. (source: <https://stackoverflow.com/questions/38479361/what-are-labels-in-graph-database>)

**Relationships:** relationships are the lines that connect nodes to other nodes; representing the relationship between them. Meaningful patterns emerge when examining the connections and interconnections of nodes, properties and edges. The edges can either be directed or undirected. In an undirected graph, an edge connecting two nodes has a single meaning. In a directed graph, the edges connecting two different nodes have different meanings, depending on their direction. Edges are the key concept in graph databases, representing an abstraction that is not directly implemented in a relational model or a document-store model. (source: [https://en.wikipedia.org/wiki/Graph\\_database#Labeled-property\\_graph](https://en.wikipedia.org/wiki/Graph_database#Labeled-property_graph))

**Properties:** Properties are information germane to nodes. For example, if Wikipedia were one of the nodes, it might be tied to properties such as website, reference material, or words that starts with the letter w, depending on which aspects of Wikipedia are germane to a given database. (source:

[https://en.wikipedia.org/wiki/Graph\\_database#Labeled-property\\_graph](https://en.wikipedia.org/wiki/Graph_database#Labeled-property_graph))

2.3. What query language does neo4j use? Please list some similarities and differences between this query language and SQL.

**Neo4j uses Cypher.**

**Similarities:** Both uses `WHERE` to set the selection condition

**Difference:** The syntaxes are different. SQL uses `SELECT` to query. However, Cyther uses `Match`. In Cyther, we have to add `Return` at the end of query in order to get the result.

3.1 What does the following query return? Run it on your neo4j instance and include a screenshot below.

MATCH (n:Person)-[r:ACTED\_IN]->(m:Movie) where n.name='Tom Hanks' return n.name, r.roles, m.title, m.released

The screenshot shows the Neo4j Desktop interface. On the left is a sidebar with 'Saved Scripts' and 'Sample Scripts'. The 'Sample Scripts' section is expanded, showing 'Basic Queries', 'Common Procedures', 'Data Profiling', and 'Example Graphs'. The 'Data Profiling' section is selected, showing various analysis options. The main panel displays a Cypher query: `neo4j$ MATCH (n:Person)-[r:ACTED_IN]->(m:Movie) where n.name='Tom Hanks' return n.name, r.roles, m.title, m.released`. Below the query, a table shows the results. The table has four columns: n.name, r.roles, m.title, and m.released. The results list Tom Hanks and his roles in various movies, including 'A League of Their Own', 'Cloud Atlas', 'The Da Vinci Code', 'Sleepless in Seattle', 'The Polar Express', 'The Green Mile', 'Cast Away', 'Charlie Wilson's War', and 'That Thing You Do!'. Below the table, a status bar indicates 'Started streaming 12 records after 2 ms and completed after 11 ms.'.

n.name	r.roles	m.title	m.released
"Tom Hanks"	["Jimmy Dugan"]	"A League of Their Own"	1992
"Tom Hanks"	["Zachry", "Dr. Henry Goose", "Isaac Sachs", "Dermot Hoggins"]	"Cloud Atlas"	2012
"Tom Hanks"	["Dr. Robert Langdon"]	"The Da Vinci Code"	2006
"Tom Hanks"	["Sam Baldwin"]	"Sleepless in Seattle"	1993
"Tom Hanks"	["Hero Boy", "Father", "Conductor", "Hobo", "Scrooge", "Santa Claus"]	"The Polar Express"	2004
"Tom Hanks"	["Paul Edgecomb"]	"The Green Mile"	1999
"Tom Hanks"	["Chuck Noland"]	"Cast Away"	2000
"Tom Hanks"	["Rep. Charlie Wilson"]	"Charlie Wilson's War"	2007
"Tom Hanks"	["Mr. White"]	"That Thing You Do"	1996

3.2 Write a query to return every movie that Kevin Bacon was in between 1990 and 2000. Include your query and screenshot.

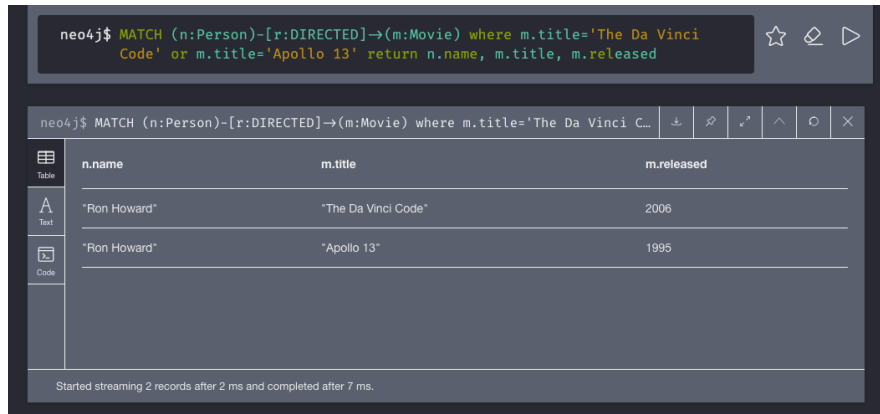
MATCH (n:Person)-[r:ACTED\_IN]->(m:Movie) where n.name='Kevin Bacon' and m.released > 1990 and m.released < 2000 return n.name, r.roles, m.title, m.released

The screenshot shows the Neo4j Desktop interface. The main panel displays a Cypher query: `neo4j$ MATCH (n:Person)-[r:ACTED_IN]->(m:Movie) where n.name='Kevin Bacon' and m.released > 1990 and m.released < 2000 return n.name, r.roles, m.title, m.released`. Below the query, a table shows the results. The table has four columns: n.name, r.roles, m.title, and m.released. The results list Kevin Bacon and his roles in 'A Few Good Men' (1992) and 'Apollo 13' (1995). Below the table, a status bar indicates 'Started streaming 2 records after 2 ms and completed after 5 ms.'.

n.name	r.roles	m.title	m.released
"Kevin Bacon"	["Capt. Jack Ross"]	"A Few Good Men"	1992
"Kevin Bacon"	["Jack Swigert"]	"Apollo 13"	1995

3.3 Find the director who directed both “The Da Vinci Code” and “Apollo 13”. Include your query and screenshot.

MATCH (n:Person)-[r:DIRECTED]->(m:Movie) where m.title='The Da Vinci Code' or m.title='Apollo 13' return n.name, m.title, m.released



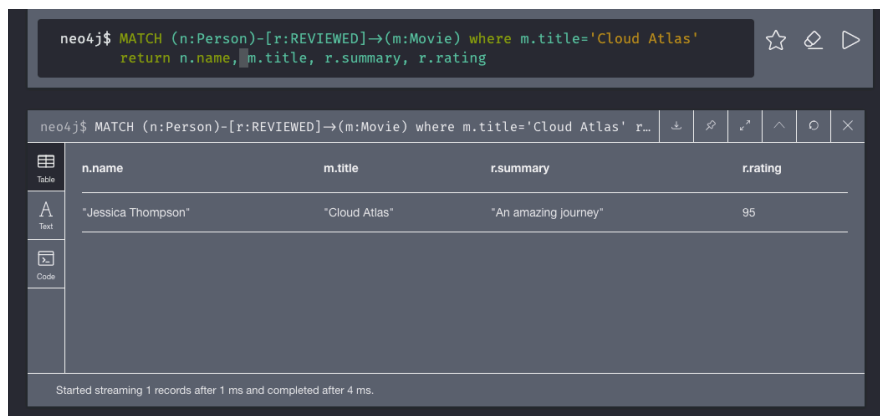
The screenshot shows the Neo4j query interface. The query entered is: `neo4j$ MATCH (n:Person)-[r:DIRECTED]->(m:Movie) where m.title='The Da Vinci Code' or m.title='Apollo 13' return n.name, m.title, m.released`. The results are displayed in a table with three columns: **n.name**, **m.title**, and **m.released**. The table contains two rows of data.

n.name	m.title	m.released
"Ron Howard"	"The Da Vinci Code"	2006
"Ron Howard"	"Apollo 13"	1995

Started streaming 2 records after 2 ms and completed after 7 ms.

3.4 Generate all of the reviews on “Cloud Atlas” include the name of the reviewer and what they said. Include your query and screenshot.

MATCH (n:Person)-[r:REVIEWED]->(m:Movie) where m.title='Cloud Atlas' return n.name, m.title, r.summary, r.rating



The screenshot shows the Neo4j query interface. The query entered is: `neo4j$ MATCH (n:Person)-[r:REVIEWED]->(m:Movie) where m.title='Cloud Atlas' return n.name, m.title, r.summary, r.rating`. The results are displayed in a table with four columns: **n.name**, **m.title**, **r.summary**, and **r.rating**. The table contains one row of data.

n.name	m.title	r.summary	r.rating
"Jessica Thompson"	"Cloud Atlas"	"An amazing journey"	95

Started streaming 1 records after 1 ms and completed after 4 ms.

3.5 Find all movies in which someone involved was both a director and an actor. List the names of these people and their movies. There may be more than one person per movie or more than one movie per person! Include your query and screenshot.

MATCH (m:Movie)<-[:ACTED\_IN]-(p:Person)-[:DIRECTED]->(m) RETURN p.name, m.title

p.name	m.title
"Tom Hanks"	"That Thing You Do"
"Clint Eastwood"	"Unforgiven"
"Danny DeVito"	"Hoffa"

Started streaming 3 records after 4 ms and completed after 36 ms.

3.6 Robin Williams wants to have a party with all of his co-actors across all of the movies that he has acted in (his A-list). He orders too much food and realizes he has to extend the guest list to make sure it is all eaten. His B-list of party invites now includes the co-actors of his co-actors. What actors are on the A-list and what actors are on the B-list. Be careful, no actor should be on both lists! Include your query and screenshot.

MATCH (a:Person{name:'Robin Williams'})-[:ACTED\_IN]->(m1:Movie)<-[:ACTED\_IN]-(A:Person) OPTIONAL MATCH(A:Person)-[:ACTED\_IN]->(m2:Movie)<-[:ACTED\_IN]-(B:Person) WHERE A.name<>B.name AND m1.title <> m2.title RETURN distinct A.name, B.name

A.name	B.name
"Cuba Gooding Jr."	"Regina King"
"Cuba Gooding Jr."	"Tom Cruise"
"Cuba Gooding Jr."	"Bonnie Hunt"
"Cuba Gooding Jr."	"Jonathan Lipnicki"
"Cuba Gooding Jr."	"Jerry O'Connell"
"Cuba Gooding Jr."	"Jay Mohr"
"Cuba Gooding Jr."	"Kelly Preston"
"Cuba Gooding Jr."	"Renee Zellweger"
"Cuba Gooding Jr."	"Aaron Sorkin"

Started streaming 36 records after 2 ms and completed after 9 ms.

4. For the final portion of the homework, come up with a dataset that you think would be best represented as a graph database. You must include at least two different types of nodes and four different types of relationships and create a database in neo4j! Each node should have at least three properties.

4.1 Why is it better to write this data in a graph database instead of a SQL database?  
Logistics data is better to write in graph database because it is easier to analyze the supply chain. The graph database makes it easier to evaluate the distance between nodes. Accordingly, we can give the best route to the drivers.

4.2 Write out the different types of nodes, properties and relationships

node	Property 1 (purpose)	Property 2 (employee)	Property 3 (location)
Factory	Manufacture	Technician	Countryside
Warehouse	Store	Logistics officer	Suburb
Store	Sale	Sales	City

Relations:

1. Distance (undirected): the distance between two nodes
2. Delivery capacity (undirected): the delivery capacity between two nodes
3. Demand (directed): the demand of product by one node to another one
4. Return (directed): the return of product by one node to another one

4.3 Write two example queries that someone looking at this dataset might need. Include the queries and the results that they yield.

the location between each node (factory, warehouse, and store)

The screenshot shows the Neo4j Browser interface. The query entered is:

```
neo4j$ MATCH (f:Factory)-[r1:DISTANCE]->(w:Warehouse)-[r2:DISTANCE]->(s:Store) RETURN f.location, r1.unit, w.location, r2.unit, s.location
```

The results are displayed in a table with 5 columns: f.location, r1.unit, w.location, r2.unit, and s.location. There are 8 rows of data.

f.location	r1.unit	w.location	r2.unit	s.location
"countryside"	60	"suburb"	30	"city"
"countryside"	20	"suburb"	30	"city"
"countryside"	60	"suburb"	15	"city"
"countryside"	20	"suburb"	15	"city"
"countryside"	25	"suburb"	10	"city"
"countryside"	40	"suburb"	10	"city"
"countryside"	25	"suburb"	45	"city"
"countryside"	40	"suburb"	45	"city"

At the bottom, it says: "Started streaming 8 records after 2 ms and completed after 5 ms."

See whether the delivery capacity covers the demand

```
MATCH (f:Factory)-[d1:DEMAND]->(w:Warehouse)-[d2:DEMAND]->(s:Store), (f)-[c1:DELIVERY_CAP]->(w)-[c2:DELIVERY_CAP]->(s) RETURN f.location, c1.unit-d1.unit, w.location, c2.unit-d2.unit, s.location
```



The image shows the Neo4j Cypher query interface. At the top, a query is entered in the editor. Below the editor, the query is repeated in a toolbar. The results are displayed in a table view with five columns: f.location, c1.unit-d1.unit, w.location, c2.unit-d2.unit, and s.location. The table contains eight rows of data. At the bottom, a status message indicates that 8 records were streamed after 1 ms and completed after 7 ms.

```
neo4j$ MATCH (f:Factory)-[d1:DEMAND]->(w:Warehouse)-[d2:DEMAND]->(s:Store), (f)-[c1:DELIVERY_CAP]->(w)-[c2:DELIVERY_CAP]->(s) RETURN f.location, c1.unit-d1.unit, w.location, c2.unit-d2.unit, s.location
```

neo4j\$ MATCH (f:Factory)-[d1:DEMAND]->(w:Warehouse)-[d2:DEMAND]...

f.location	c1.unit-d1.unit	w.location	c2.unit-d2.unit	s.location
"countryside"	-30	"suburb"	3	"city"
"countryside"	-5	"suburb"	3	"city"
"countryside"	-30	"suburb"	-9	"city"
"countryside"	-5	"suburb"	-9	"city"
"countryside"	-5	"suburb"	-21	"city"
"countryside"	-30	"suburb"	-21	"city"
"countryside"	-5	"suburb"	-3	"city"
"countryside"	-30	"suburb"	-3	"city"

Started streaming 8 records after 1 ms and completed after 7 ms.