**Graph Thinking**

**Module Overview**

In this module, you will learn about:

* Leonhard Euler, the Swiss mathematician who founded the study of graph theory.
* Elements that make up a graph and how they are used to solve real-world problems.
* The structure of the graphs and how they are traversed.
* Common use-cases for graphs.

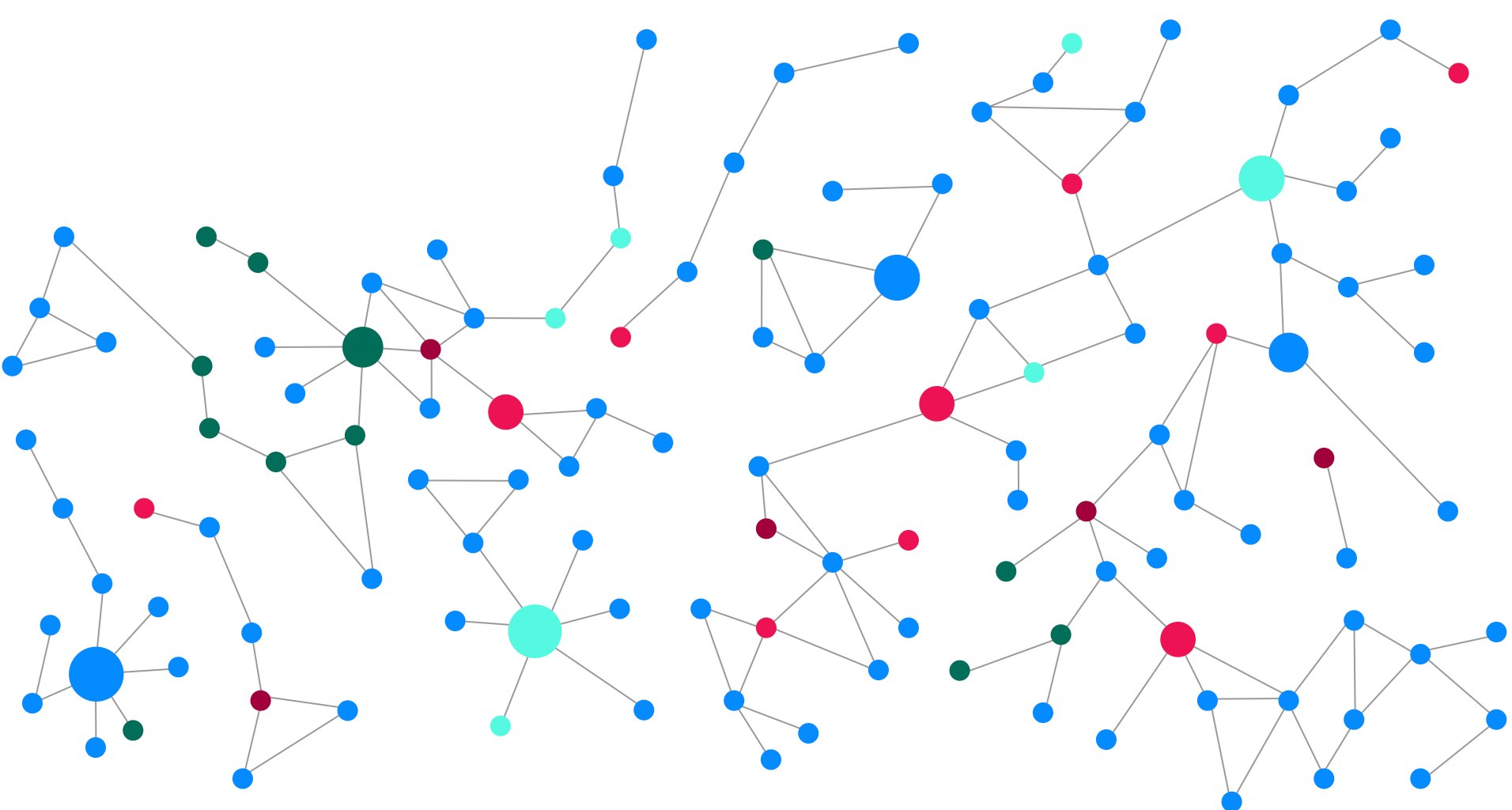
If you are ready, let’s get going!

# Graph Elements

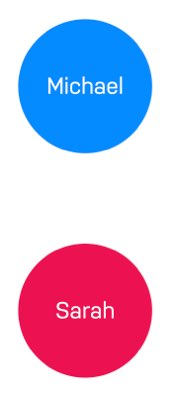
## Graph elements

Let’s take a closer look at the two elements that make up a graph:

* Nodes (also known as vertices)
* Relationships (also known as edges)



### Nodes



**Nodes** (or vertices) are the circles in a graph. Nodes commonly represent objects, entities, or merely things.

In the [**Seven Bridges of Königsberg**](https://graphacademy.neo4j.com/courses/neo4j-fundamentals/1-graph-thinking/1-seven-bridges/) example in the previous lesson, nodes were used to represent the land masses.

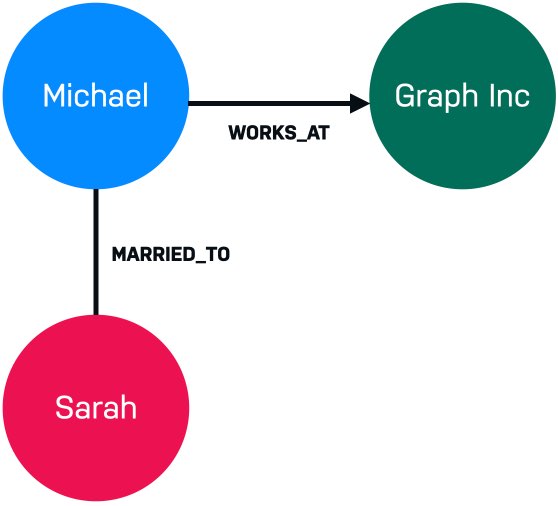
Another example that everyone can relate to is the concept of a social graph. People interact with each other and form relationships of varying strengths.

The diagram to the right has two nodes which represent two people, **Michael** and **Sarah**. On their own, these elements are uninspiring. But when we start to connect these circles together, things start to get interesting.

#### Nodes typically represent things

Examples of entities that could typically be represented as a node are: person, product, event, book or subway station.

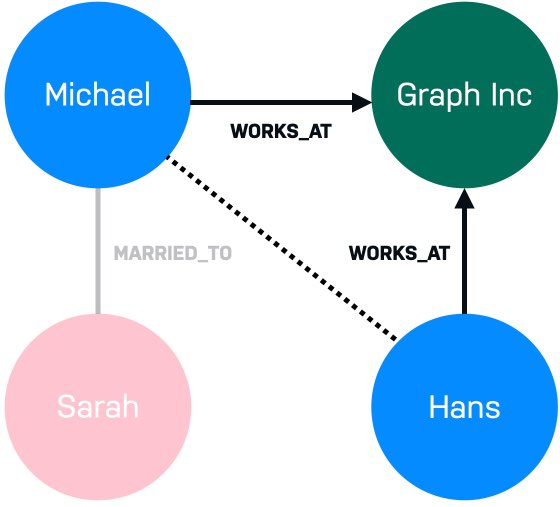
### Relationships



**Relationships** (or edges) are used to connect nodes. We can use relationships to describe how nodes are connected to each other. For example **Michael** has the **WORKS\_AT** relationship to **Graph Inc** because he works there. **Michael** has the **MARRIED\_TO** relationship to **Sarah** because he is married to her.

All of a sudden, we know that we are looking at the beginnings of some sort of social graph.

Now, let’s introduce a third person, **Hans**, to our Graph.



**Hans** also works for **Graph Inc** along with Michael. Depending on the size of the company and the properties of the relationship, we may be able to infer that Michael and Hans know each other.

If that is the case, how likely is it that Sarah and Hans know each other?

These are all questions that can be answered using a graph.

#### Relationships are typically verbs.

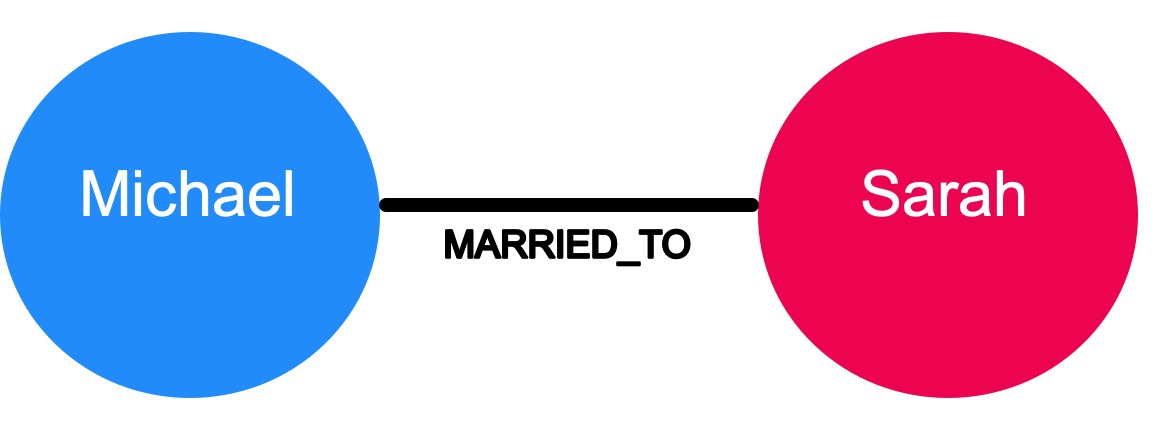
We could use a relationship to represent a personal or professional connection (Person ***knows*** Person, Person ***married to*** Person), to state a fact (Person ***lives in*** Location, Person ***owns*** Car, Person ***rated*** Movie), or even to represent a hierarchy (Parent ***parent of*** Child, Software ***depends on*** Library).

# Graph Structure

## Graph characteristics and traversal

There are a few types of graph characteristics to consider. In addition, there are many ways that a graph may be traversed to answer a question.

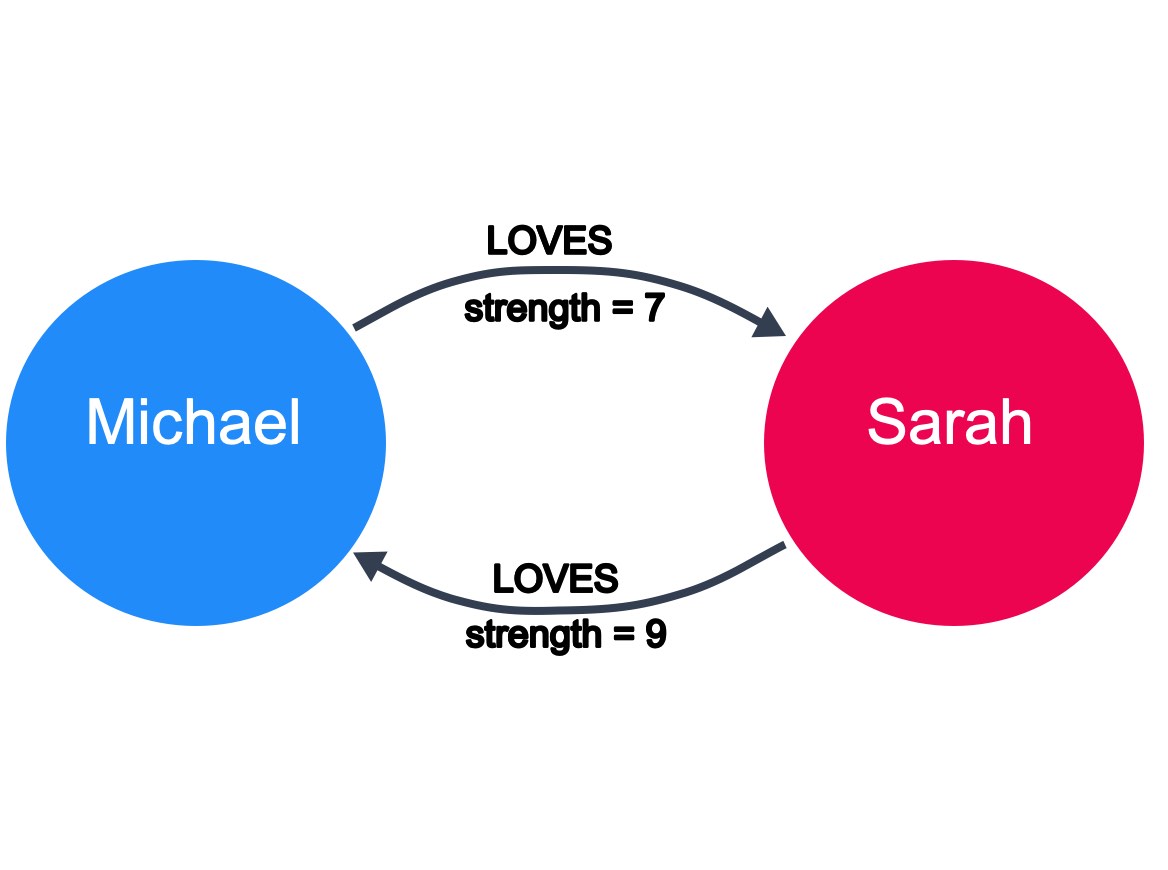
### Directed vs. undirected graphs



In an undirected graph, relationships are considered to be bi-directional or symmetric.

An example of an undirected graph would include the concept of marriage. If Michael is married to Sarah, then it stands to reason that Sarah is also married to Michael.

A directed graph adds an additional dimension of information to the graph. Relationships with the same type but in opposing directions carry a different semantic meaning.



For example, if marriage is a symmetrical relationship, then the concept of love is asymmetrical. Although two people may like or love each other, the amount that they do so may vary drastically. Directional relationships can often be qualified with some sort of weighting. Here we see that the strength of the LOVES relationship describes how much one person loves another.

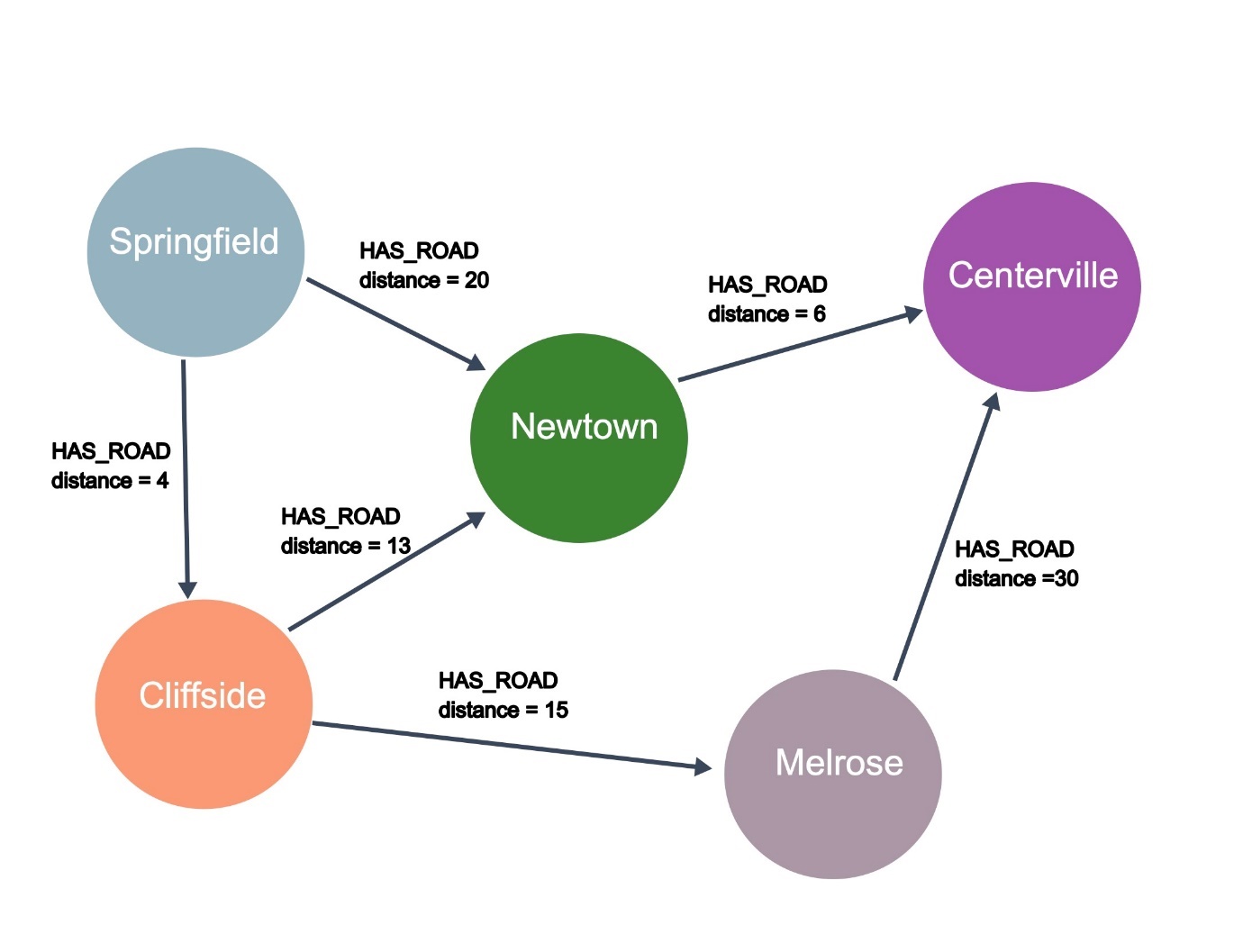
At a larger scale, a large network of social connections may also be used to understand network effects and predict the transfer of information or disease. Given the strength of connections between people, we can predict how information would spread through a network.

### Weighted vs. unweighted graphs

The concept of love is also an example of a weighted graph.

In a weighted graph, the relationships between nodes carry a value that represents a variety of measures, for example cost, time, distance or priority.

A basic shortest path algorithm would calculate the shortest distance between two nodes in the graph. This could be useful for finding the fastest walking route to the local store or working out the most efficient route to travel from city to city.



In this example, the question that we might have for this graph is: What is the shortest drive from Springfield to Centerville? Using the HAS\_ROAD relationships and the distance for these relationships, we can see that the shortest drive will be to start in Springfield, then go to Cliffside, then to Newtown, and finally arrive in Centerville.

More complex shortest path algorithms (for example, Dijkstra’s algorithm or A\* search algorithm) take a weighting property on the relationship into account when calculating the shortest path. Say we have to send a package using an international courier, we may prefer to send the package by air so it arrives quickly, in which case the weighting we would take into account is the time it takes to get from one point to the next.

Inversely, if cost is an issue we may prefer to send the package by sea and therefore use a property that represents cost to send the package.

### Graph traversal

How one answers questions about the data in a graph is typically implemented by traversing the graph. To find the shortest path between Springfield to Centerville, the application would need to traverse all paths between the two cities to find the shortest one.

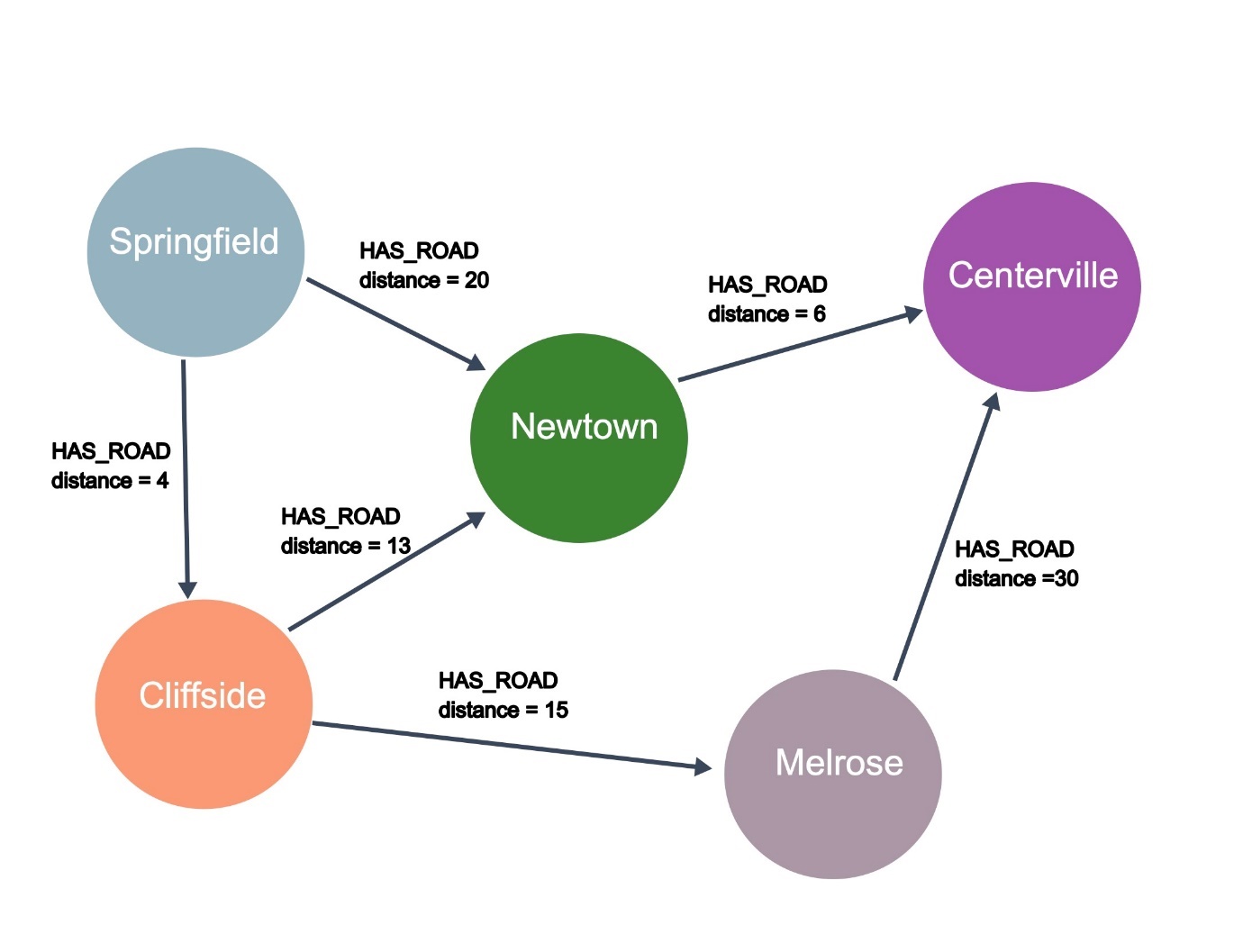
* Springfield-Newtown-Centerville = 26
* Springfield-Cliffside-Newtown-Centerville = 23
* Springfield-Cliffside-Melrose-Certerville = 49

Traversal implies that the relationships are followed in the graph. There are different types of traversals in graph theory that can impact application performance. For example, can a relationship be traversed multiple times or can a node be visited multiple times?

Neo4j’s Cypher statement language is optimized for node traversal so that relationships are not traversed multiple times, which is a huge performance win for an application.

## How many paths are traversed?

Given this graph:



Suppose we have this graph where a relationship can be traversed multiple times. How many unique paths are traversed to go from Springfield to Centerville when following the direction of the relationship?

* 

1

* 

2

* 

3

* 

4

**Properties of Graph**

Label the nodes (Male/Female, person/place/things, Employee/Doctor)

Properties of nodes :

Key value pair

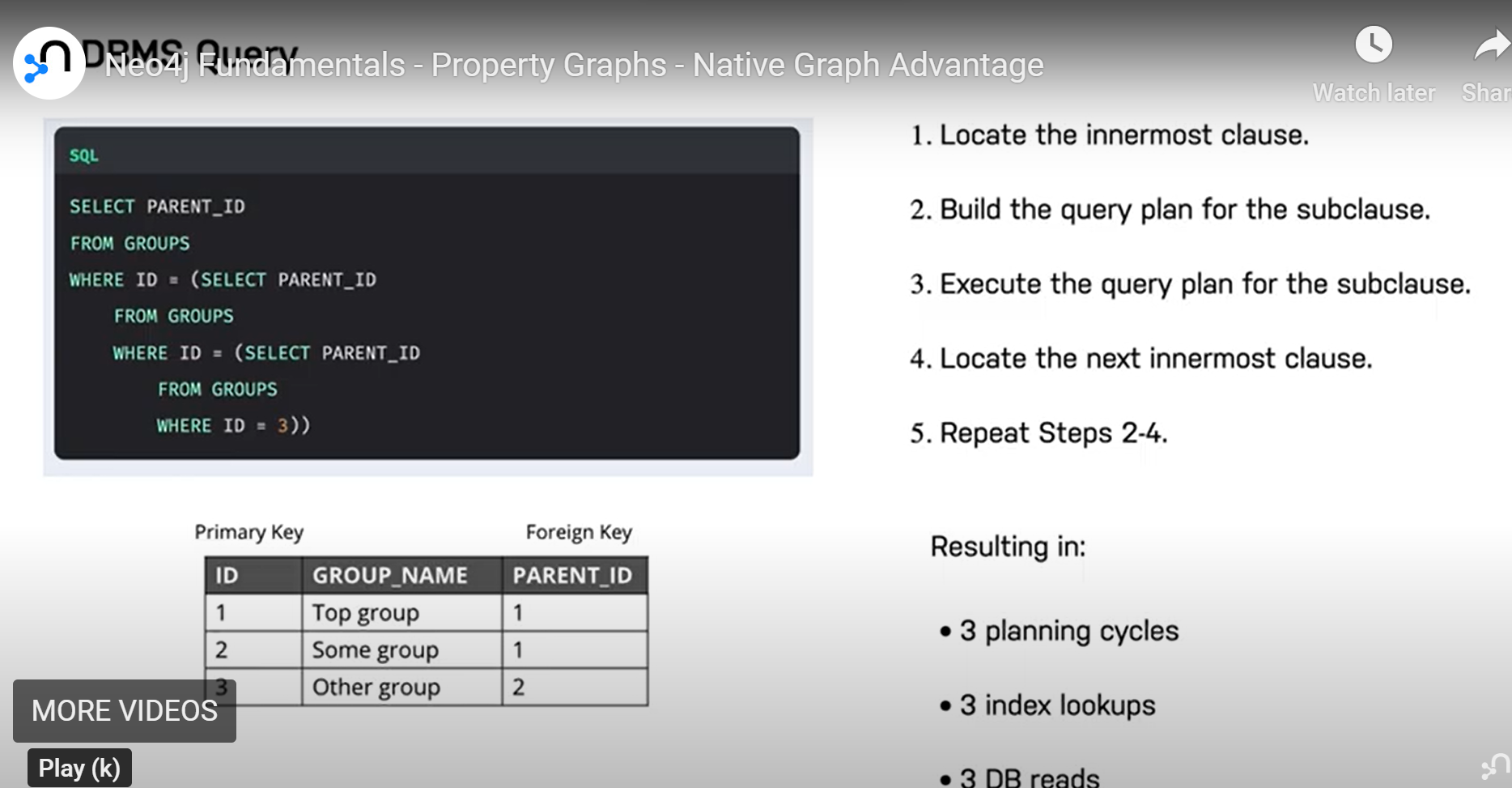
FirstName,LastName,DOB,City,

Property of Edges/Relations :

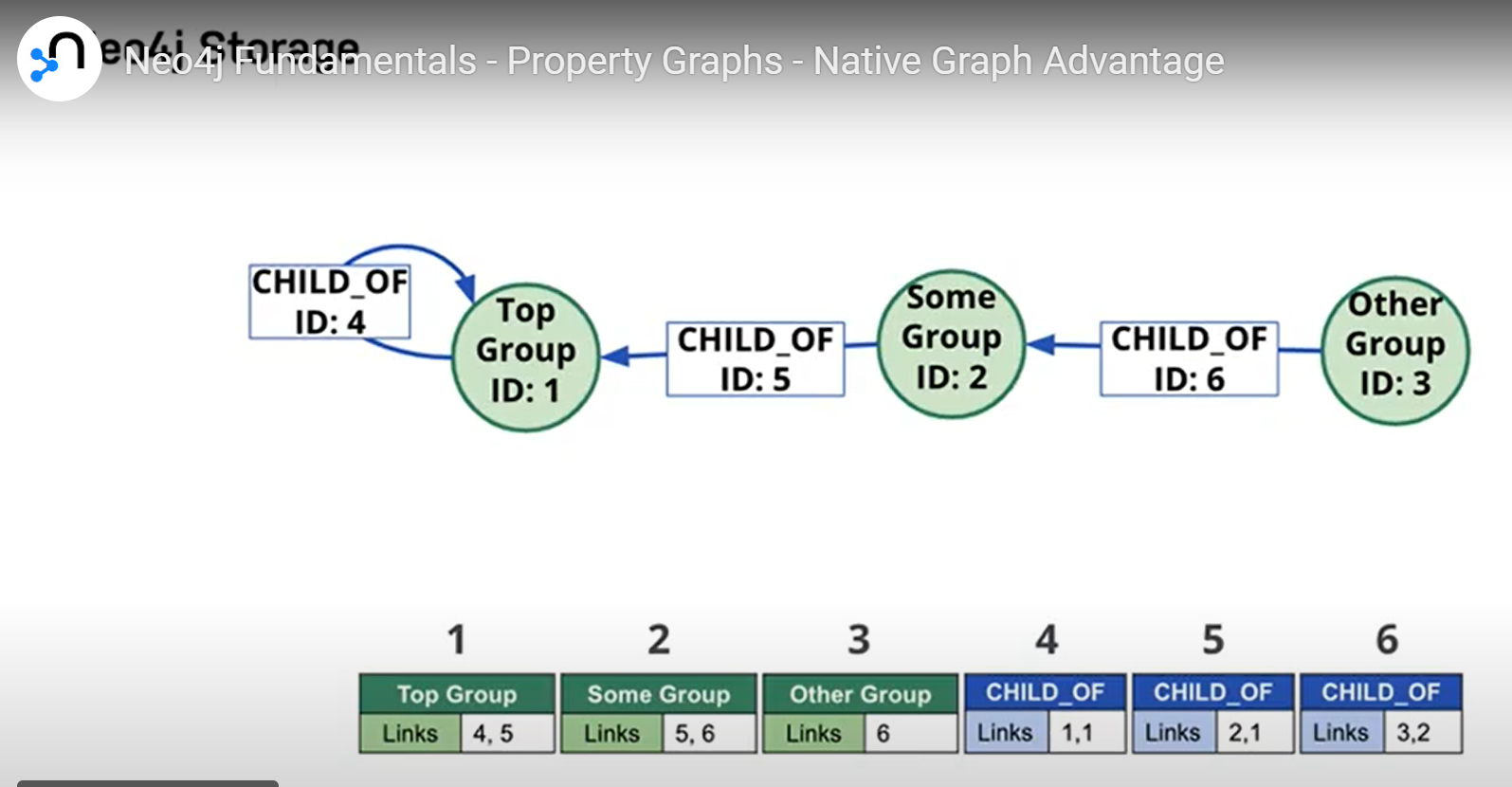
Directions

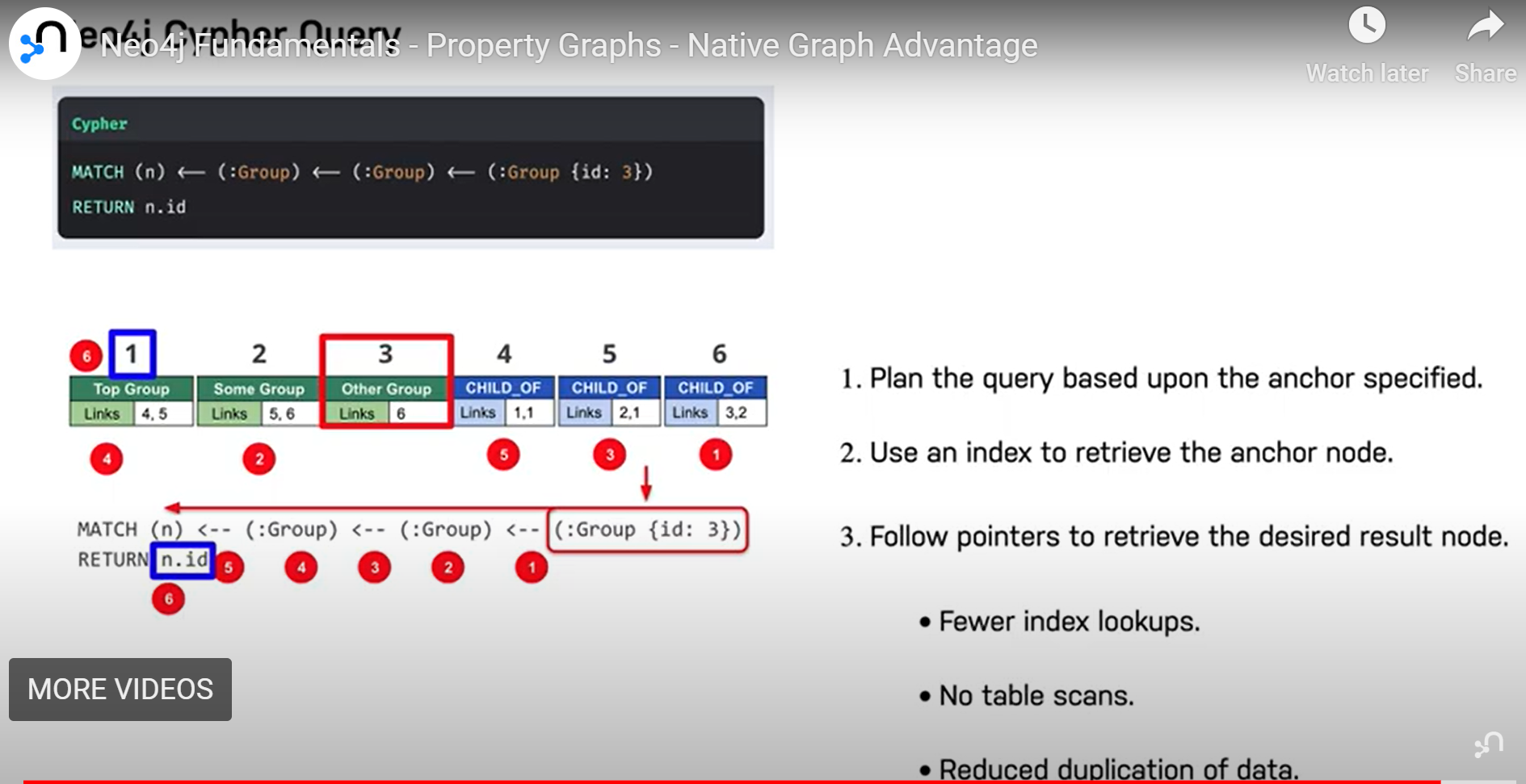
**Native Graph Advantage**

RDBMS

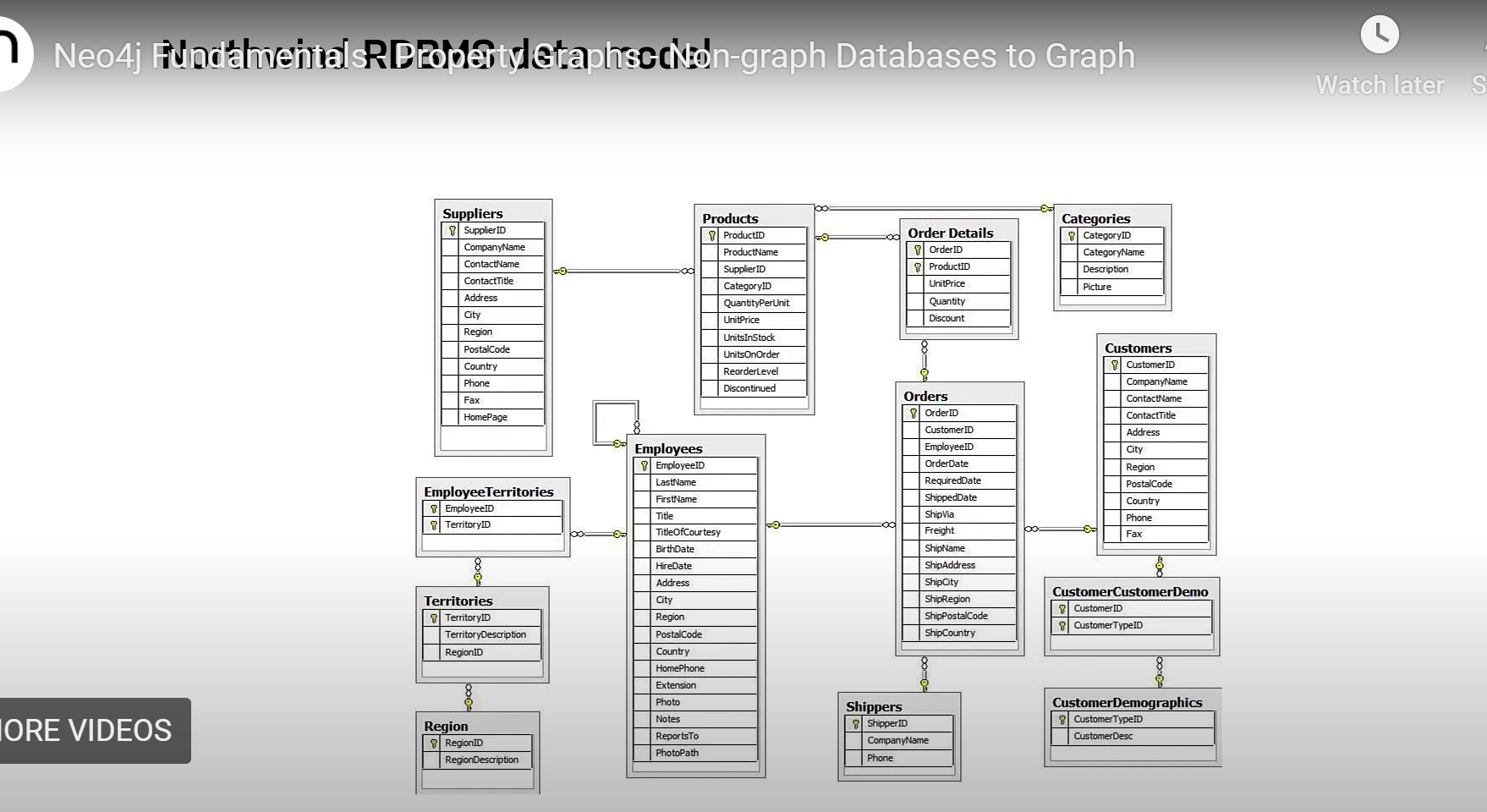


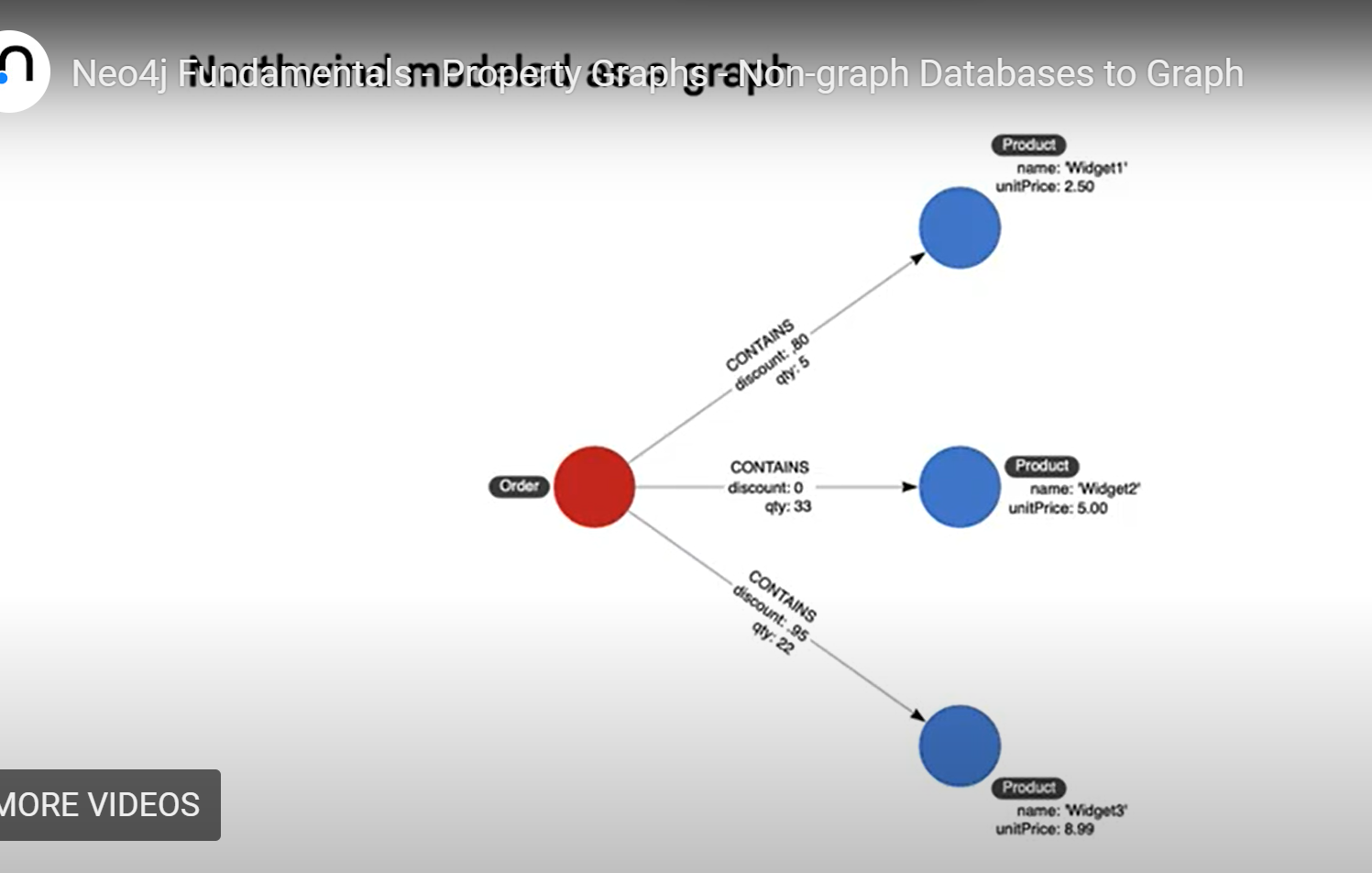
In Neo4j





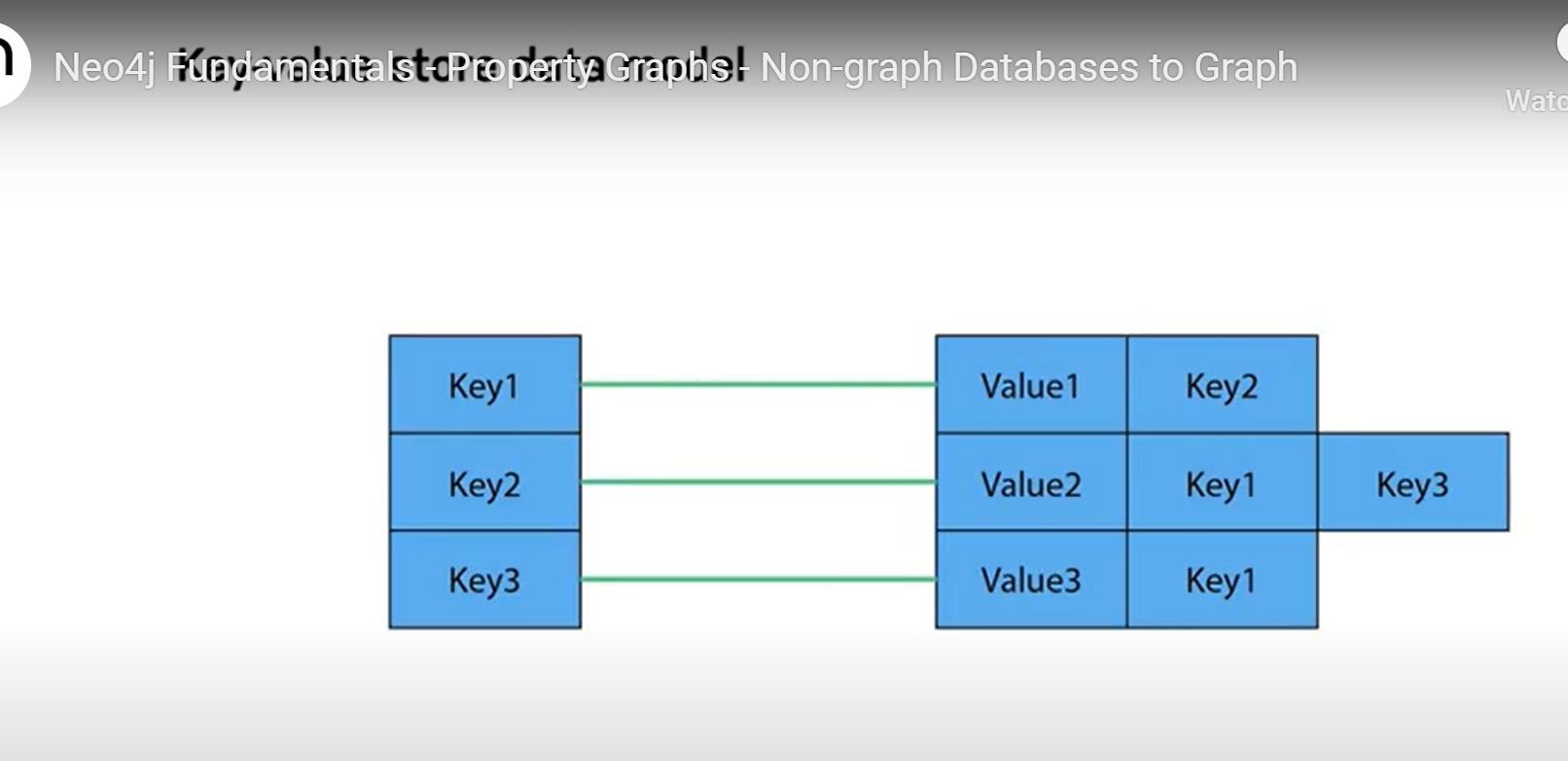
Non-graph DB into Graph

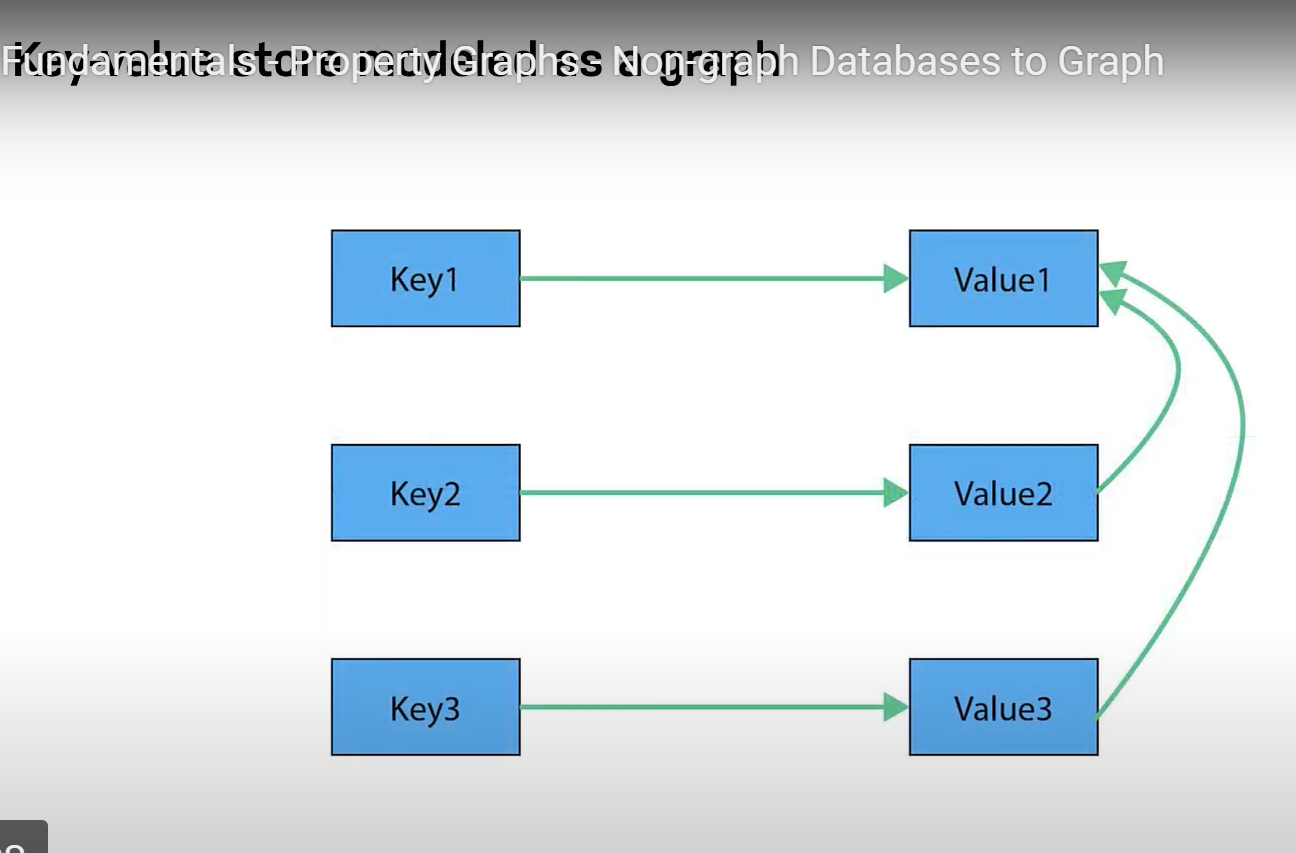


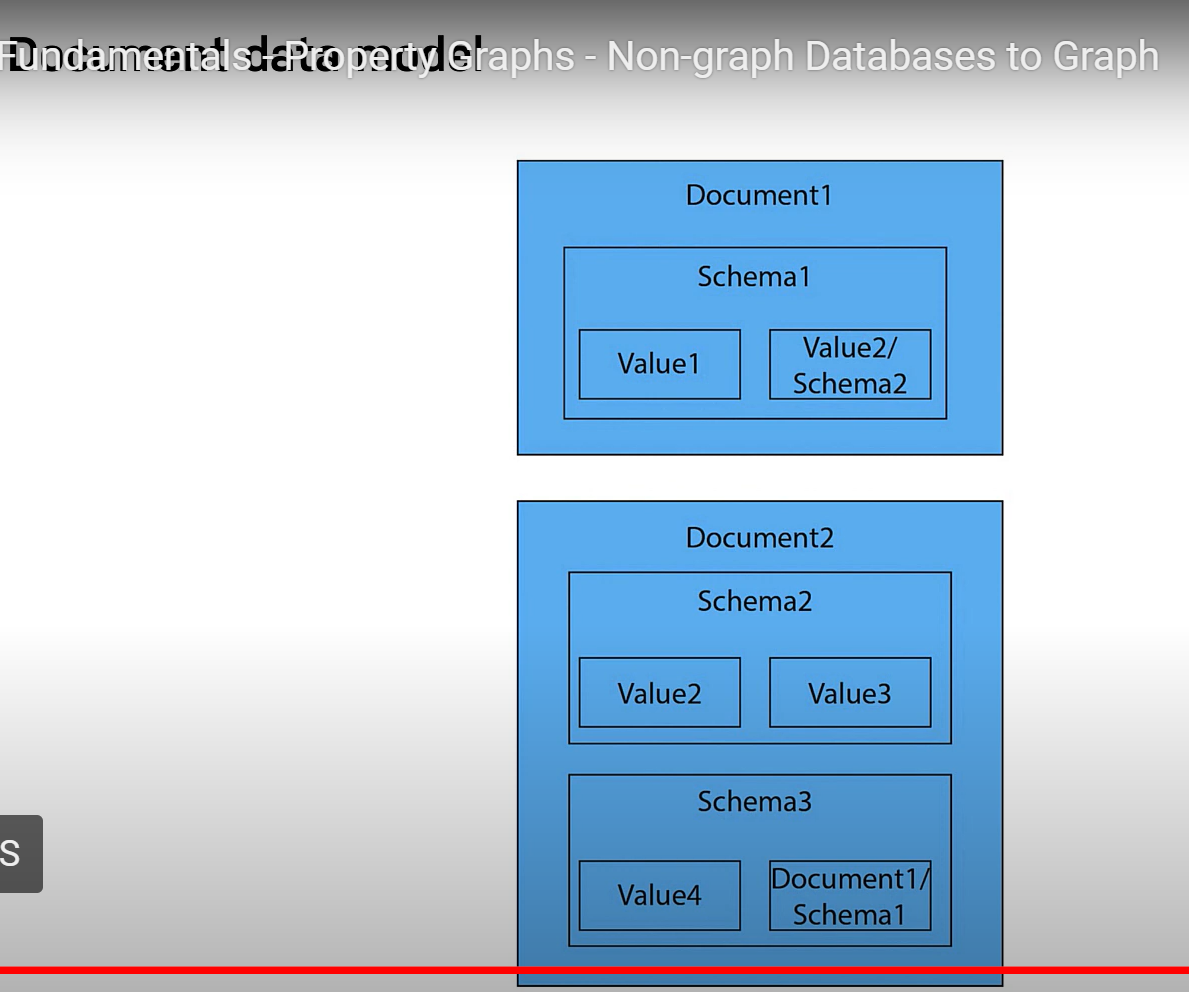


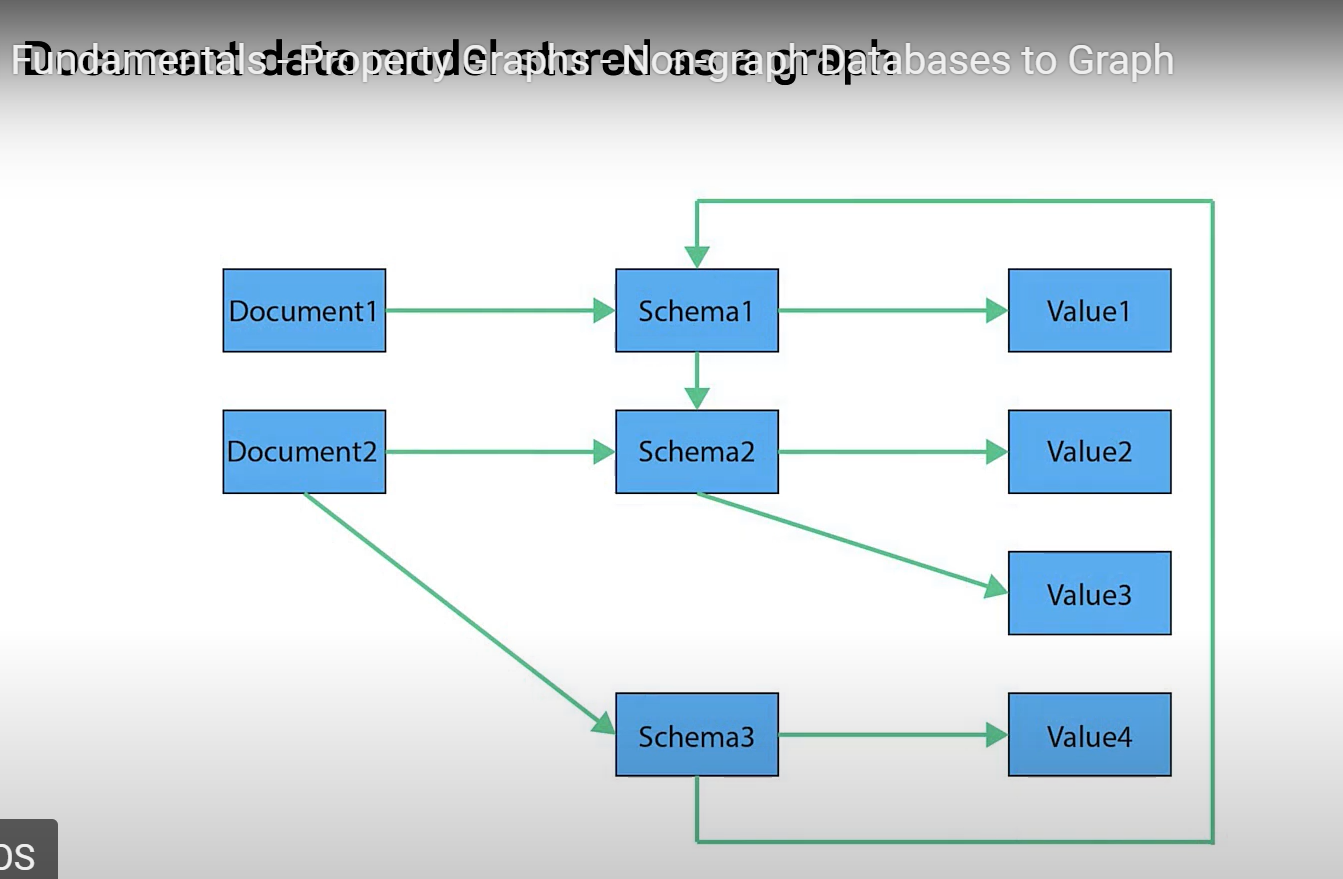
Some No-SQL DB

* Key-value stores
* Document Stores









# Graphs Are Everywhere

[**Video**](https://graphacademy.neo4j.com/courses/neo4j-fundamentals/1-graph-thinking/4-graphs-are-everywhere/#video)[**Transcript**](https://graphacademy.neo4j.com/courses/neo4j-fundamentals/1-graph-thinking/4-graphs-are-everywhere/#transcript)

## Use cases for graphs

As we discovered previously, the fundamental structure of a graph has applications far beyond mathematics. In fact, you may have seen the phrase **Graphs are Everywhere** across the [**neo4j.com**](https://neo4j.com/) website. It is our hope that you will start to see the connections between things everywhere.

Neo4j hosts a site that contains example graphs (data models) that have been designed by Neo4j engineers and Neo4j Community members. You can browse the graphgists by use case or industry. You can also use a graphgist as a starting point for your application’s graph.

[**Explore the Neo4j Graphgists**](https://neo4j.com/graphgists/).

Here are a some commonly-used use cases for Neo4j.

### E-commerce and real-time recommendations

Many online stores are traditionally built and run on relational databases. But by adding a graph database, either as a primary data store or as an additional data store, we can start to serve real time recommendations.

The first area that can be improved in e-commerce is the category hierarchy. To find products in a parent and subsequent child categories can be difficult in a traditional SQL query, or require the duplication of data. Conversely, this can be represented in a couple of lines of Cypher:

**cypher**

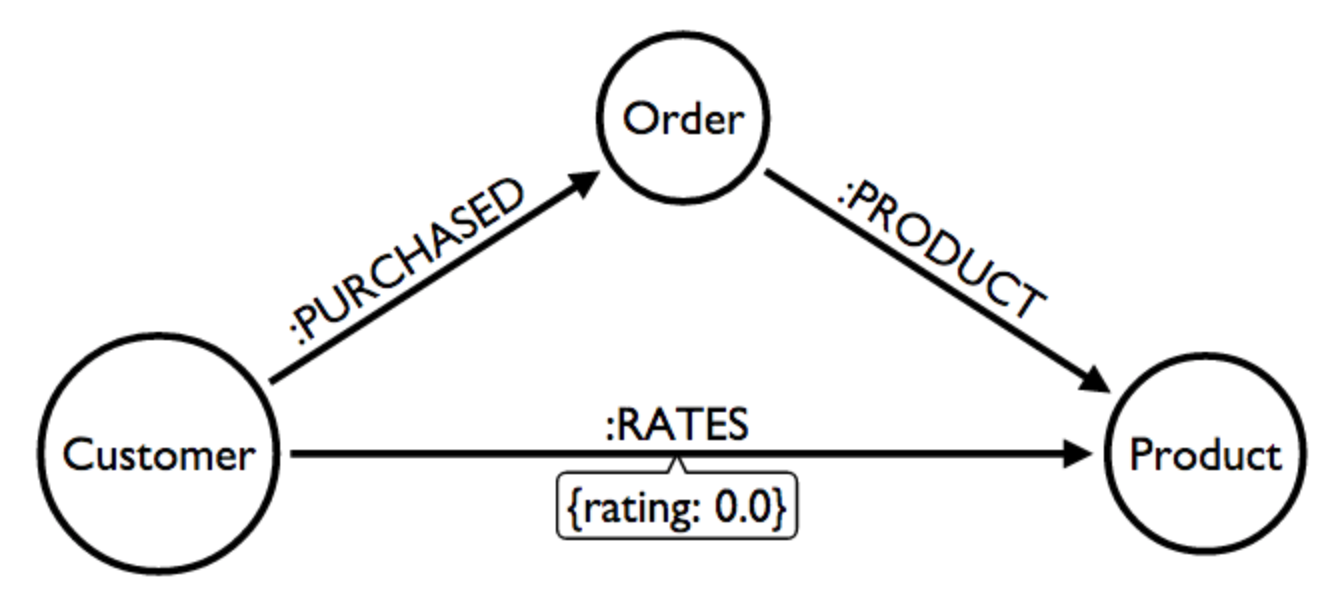
MATCH (c:Category)-[:HAS\_CHILD|HAS\_PRODUCT\*1..3]->(p:Product)

RETURN p.id, p.title, collect(c.name) AS categories

You may also be familiar with the **People who bought {Product A} also bought…​** sections on your favorite online store. These types of recommendations can be computationally expensive to generate due to the large amount of data that needs to be held in memory. This creates the need for batch processes to be deployed in order to generate the recommendations.

Where graph databases have the advantage in this use case, is that a much smaller proportion of the graph needs to be traversed in order to generate the recommendation. You can simply traverse from one Product node, through the users who have purchased that product and onwards to the subsequent products that they have bought.

Given the existing data in the graph about Customers, Orders, and Products, we can infer the rating for a product based upon the number of times the customer ordered a product.



This uses case is described in the Neo4j GraphGist site. [**View the Northwind Recommendation Engine example GraphGist**](https://neo4j.com/graphgists/northwind-recommendation-engine/)

### Investigative journalism

The most prominent user of Neo4j for investigative journalism is the International Consortium of Investigative Journalists ([**ICIJ**](https://icij.org/)). One such graph that was created by the ICIJ was the Panama Papers. The purpose of this graph was to identify possible corruption based upon the relationships between people, companies, and most importantly financial institutions.

We have a subset of the Panama Papers investigation in a [**Neo4j Graphgist**](https://neo4j.com/graphgists/the-panamapapers-example-dataset-president-of-azerbaijan/) representing the family of the Azerbaijan’s President Ilham Aliyev.



The purpose of this graph to enable one to answer these questions:

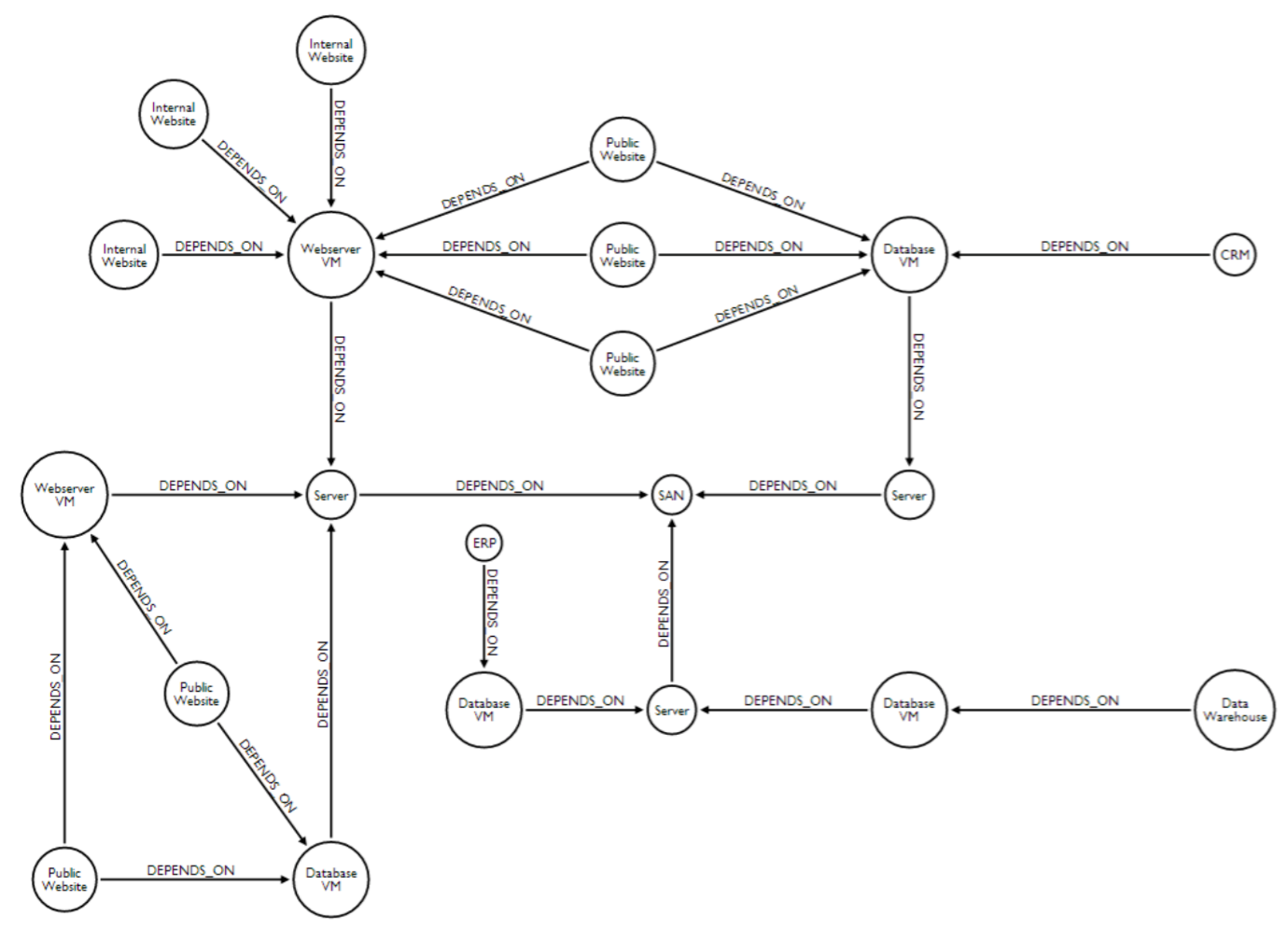
* What families with the name that contains the string 'aliye' are Officers of Companies?
* How is the family with the name that contains the string 'aliye' related to Companies?
* How are Officers related to each other?
* What are the connections between multiple companies and a family?

Another graph that has been created by the ICIJ contains information on almost 350,000 offshore entities that are part of the Paradise and Panama Papers and the Offshore Leaks investigations. The Offshore Leaks data exposes a set of connections between people and offshore entities. You can play with this graph by creating a [**Paradise Papers Sandbox**](https://sandbox.neo4j.com/?usecase=icij-paradise-papers/) and querying the data.

### Network and IT operations

Many enterprises use Neo4j to help them understand how information flows through a system and how components of a network are related. This is useful for planning, analysis of costs, and also to troubleshoot problems when a problem arises.

One of our Neo4j Community members contributed this sample data model to demonstrate how one might use a graph to identify network dependencies. Here is the data model:

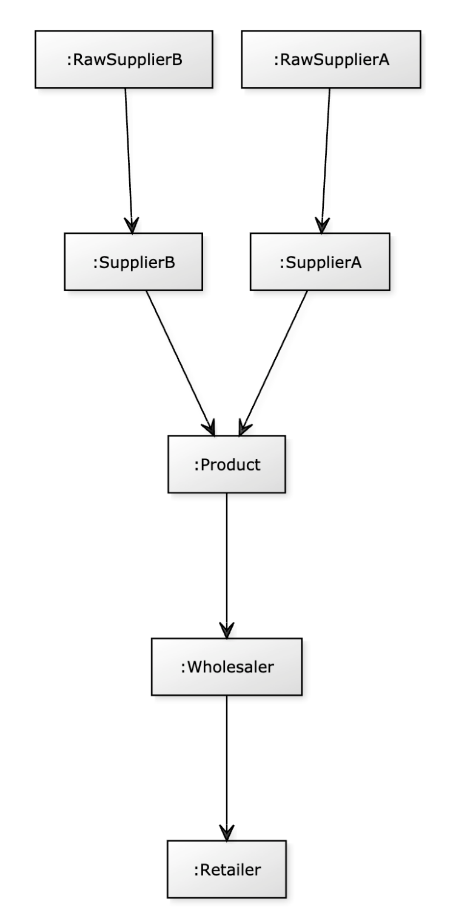


You can use this type of data model to answer:

* What are the direct dependencies of public websites?
* What are the direct dependencies of internal websites?
* What is the most depended-upon component?
* Find the dependency chain for a business critical component.
* What is the impact of removing a server?

[**View the Network Dependency Graphgist**](https://neo4j.com/graphgists/network-dependency-graph/).

### Transportation and logistics



Here is an example data model contributed by a Neo4j Community member related to supply chain management. Entities that are modeled include raw suppliers, suppliers, products, wholesalers, and retailers. All of these entities are located somewhere and there is a distance between them that will impact how quickly products can be transported.

With this graph, one can answer these questions:

* Who is the best wholesaler for each retailer based upon distance?
* Which raw supplier will give a particular retailer the freshest products?
* Which retailer provides locally grown products?
* How can we rate each supply chain?