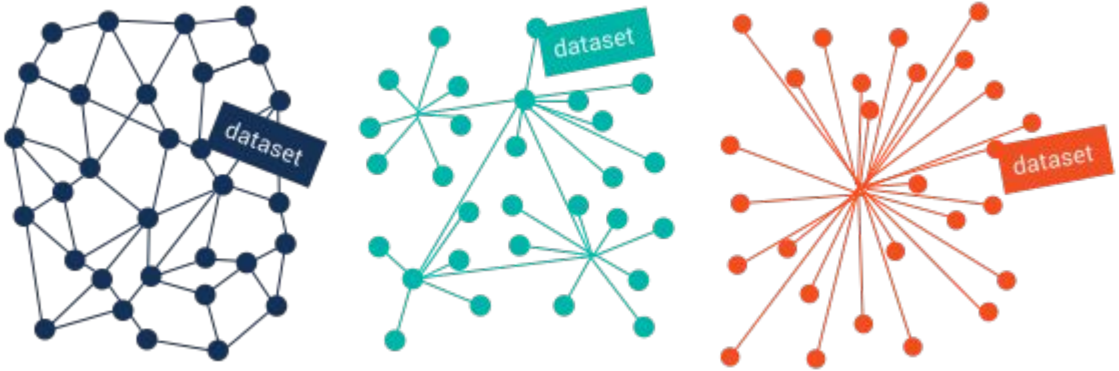
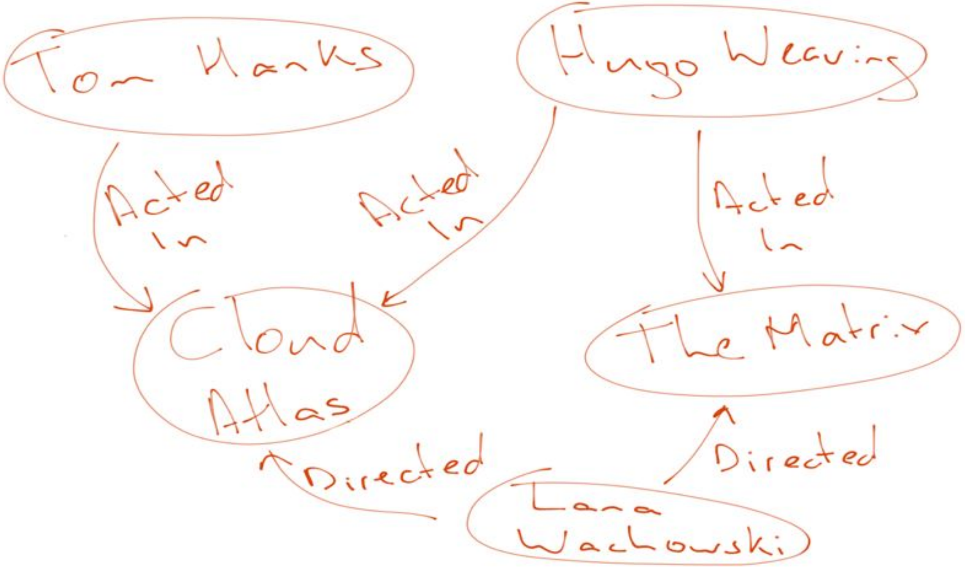
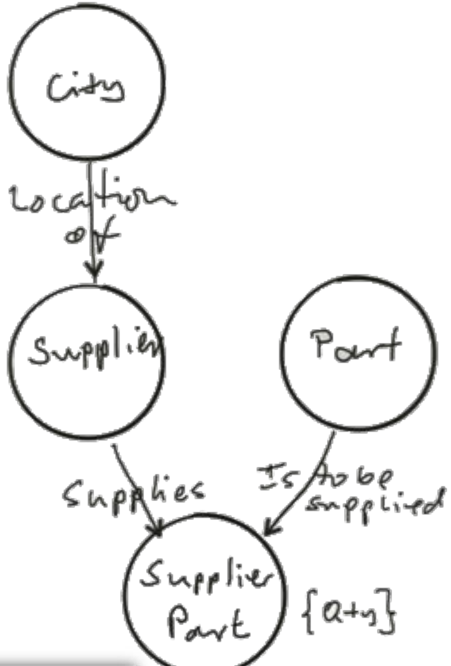
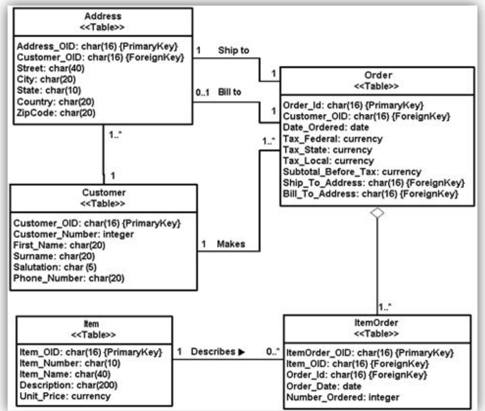
Introduction to Graph Databases



Graphs everywhere







Address
<<Table>>

Address_OID: char(16) {PrimaryKey}
Customer_OID: char(16) {ForeignKey}
Street: char(40)
City: char(20)
State: char(10)
Country: char(20)
ZipCode: char(20)

Order
<<Table>>

Order_Id: char(16) {PrimaryKey}
Customer_OID: char(16) {ForeignKey}
Date_Ordered: date
Tax_Federal: currency
Tax_State: currency
Tax_Local: currency
Subtotal_Before_Tax: currency
Ship_To_Address: char(16) {ForeignKey}
Bill_To_Address: char(16) {ForeignKey}

Customer
<<Table>>

Customer_OID: char(16) {PrimaryKey}
Customer_Number: integer
First_Name: char(20)
Surname: char(20)
Salutation: char (5)
Phone_Number: char(20)

Item
<<Table>>

Item_OID: char(16) {PrimaryKey}
Item_Number: char(10)
Item_Name: char(40)
Description: char(200)
Unit_Price: currency

ItemOrder
<<Table>>

ItemOrder_OID: char(16) {PrimaryKey}
Item_OID: char(16) {ForeignKey}
Order_Id: char(16) {ForeignKey}
Order_Date: date
Number_Ordered: integer

What is a graph anyway?

A set of things (nodes) that are connected to each other (edges)

What is a graph anyway?

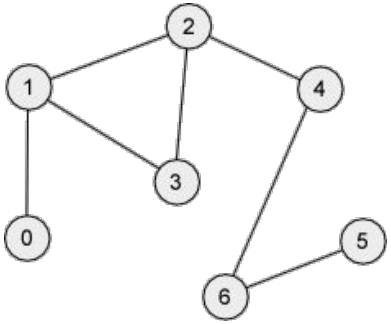
A set of things (nodes) that are connected to each other (edges)

There are many versions though:

What is a graph anyway?

A set of things (nodes) that are connected to each other (edges)

There are many versions though:

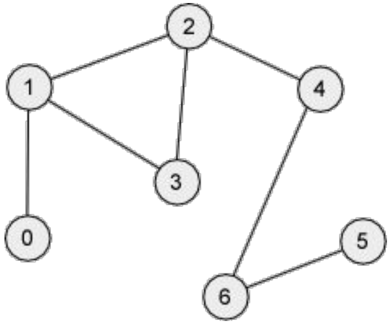


unweighted/undirected graph

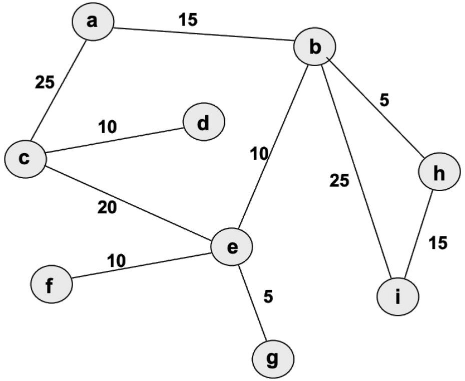
What is a graph anyway?

A set of things (nodes) that are connected to each other (edges)

There are many versions though:



unweighted/undirected graph

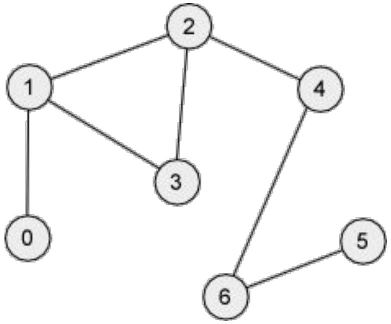


weighted graph

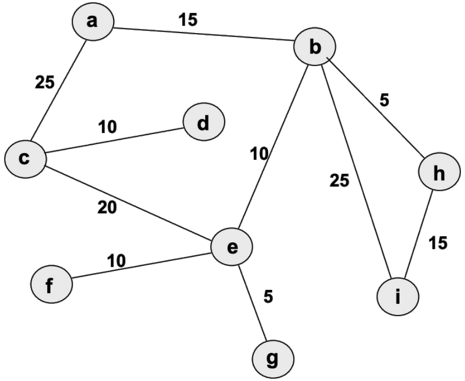
What is a graph anyway?

A set of things (nodes) that are connected to each other (edges)

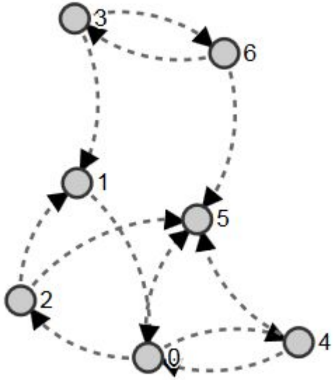
There are many versions though:



unweighted/undirected graph



weighted graph



directed graph

Everything is about labels and attributes

Everything is about labels and attributes

:Labels -> types of the node or edge

Everything is about labels and attributes

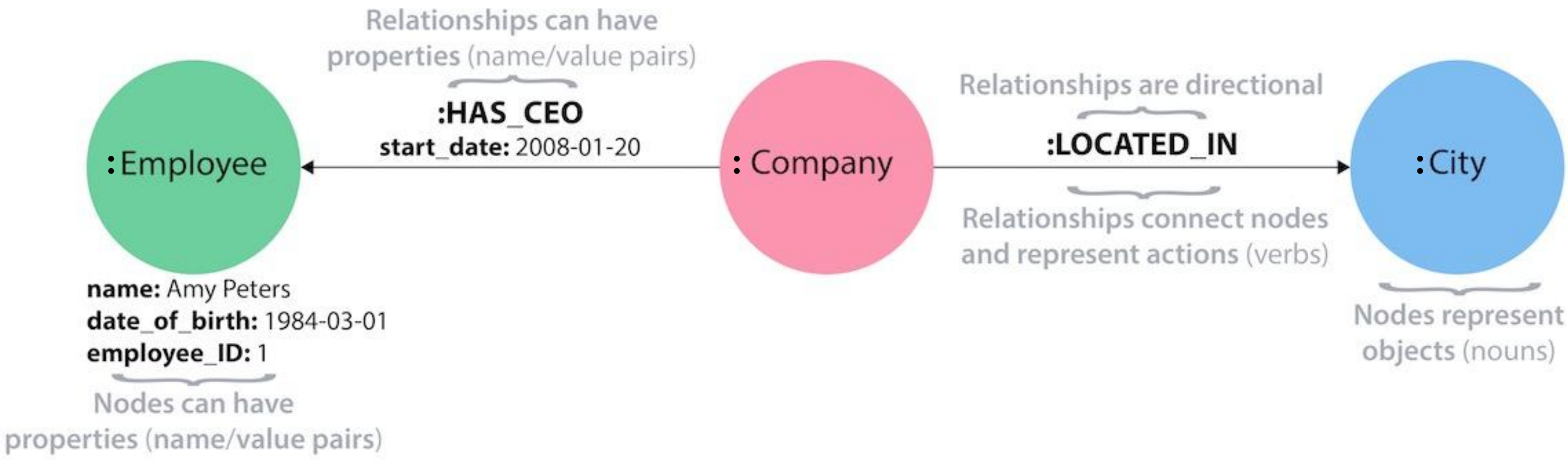
:Labels -> types of the node or edge

{attributes} -> json object of all the data in the node or edge

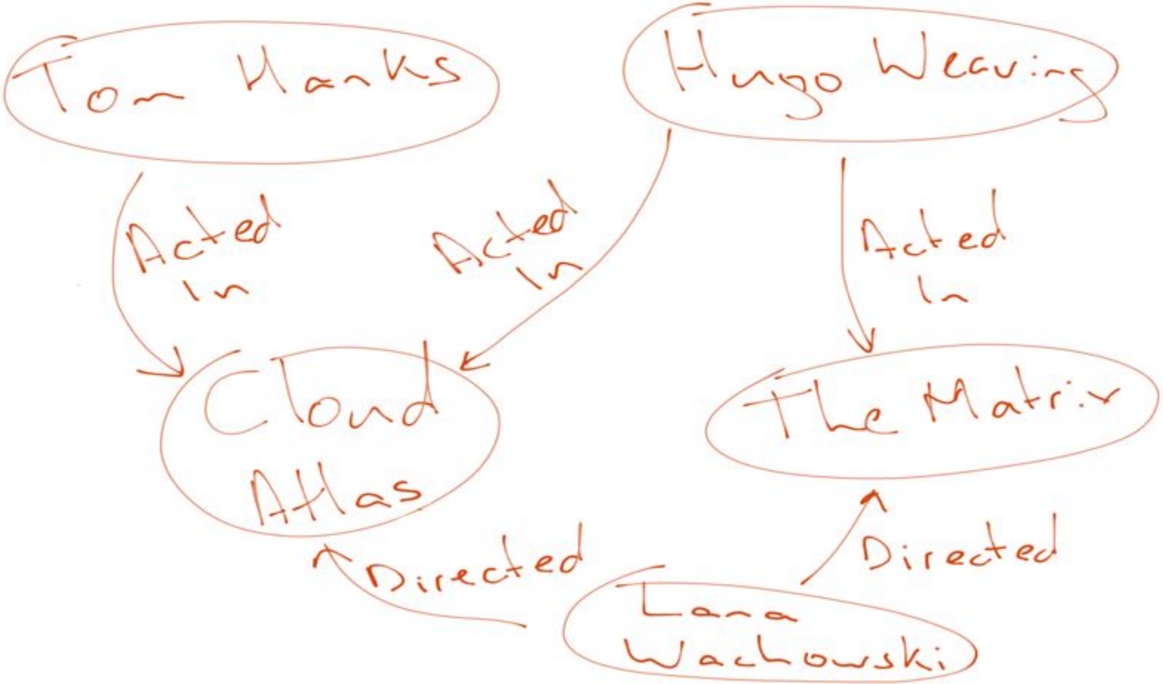
Everything is about labels and attributes

:Labels -> types of the node or edge

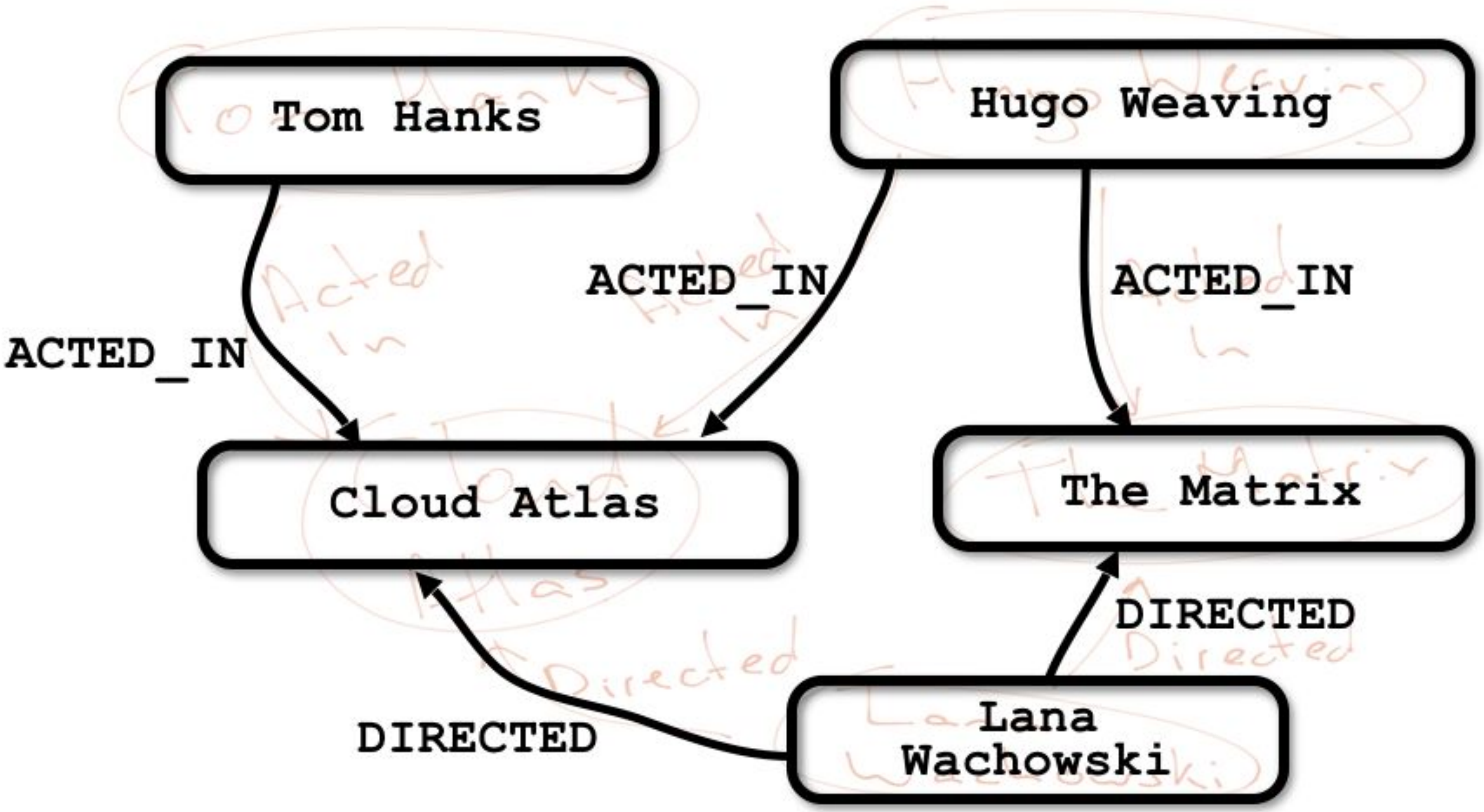
{attributes} -> json object of all the data in the node or edge


Relationships can have properties (name/value pairs)
:HAS_CEO
start_date: 2008-01-20
:Employee
: Company
Relationships are directional
:LOCATED_IN
:City
Relationships connect nodes and represent actions (verbs)
name: Amy Peters
date_of_birth: 1984-03-01
employee_ID: 1
Nodes can have properties (name/value pairs)
Nodes represent objects (nouns)

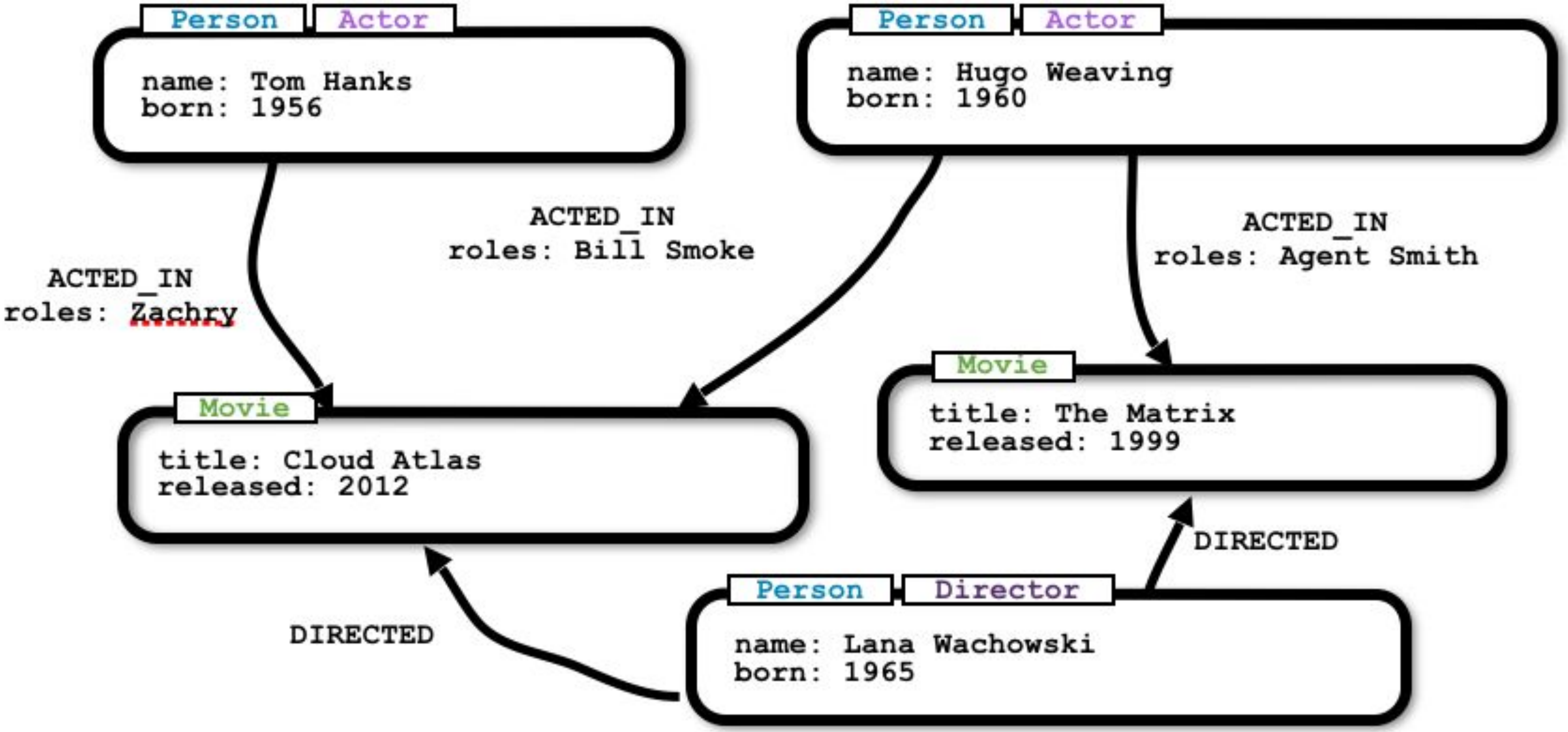
So we have


Tom Hanks
Hugo Weaving
Acted In
Acted In
Acted In
Cloud Atlas
The Matrix
Directed
Directed
Lana Wachowski

So we have

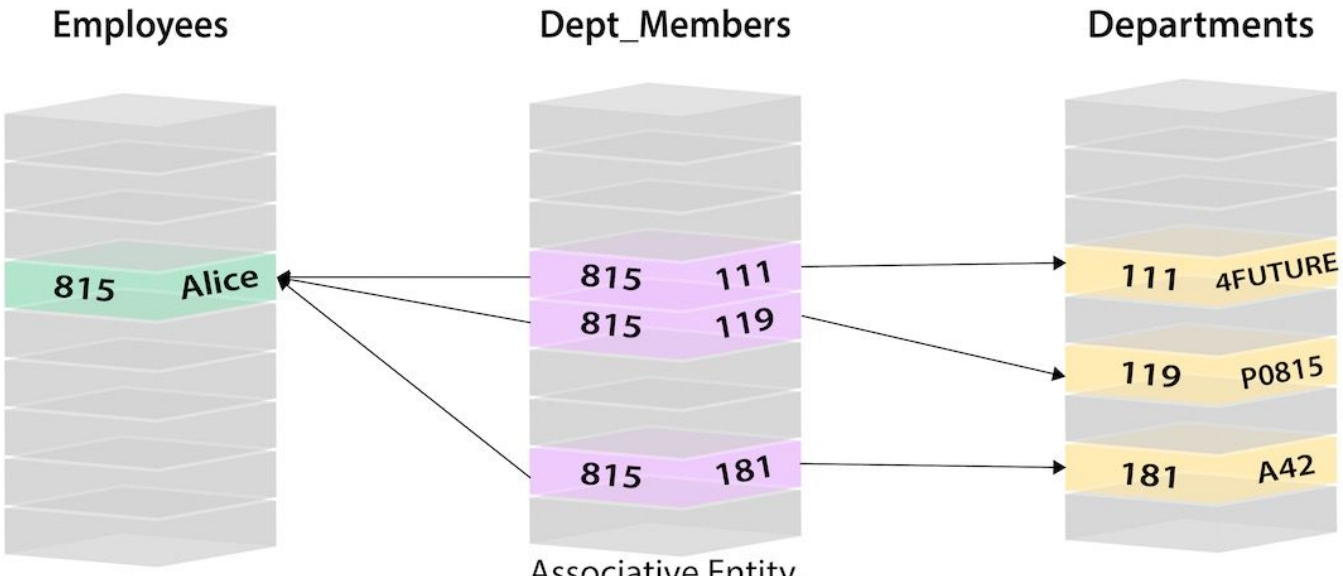

Tom Hanks
Hugo Weaving
ACTED_IN
ACTED_IN
ACTED_IN
Cloud Atlas
The Matrix
DIRECTED
DIRECTED
Lana Wachowski
Tom Hanks
Hugo Weaving
Acted In
Acted In
Acted In
Cloud Atlas
The Matrix
Directed
Directed
Lana Wachowski

So we have


Person
Actor
name: Tom Hanks
born: 1956
Person
Actor
name: Hugo Weaving
born: 1960
ACTED_IN
roles: Bill Smoke
ACTED_IN
roles: Zachry
ACTED_IN
roles: Agent Smith
Movie
title: The Matrix
released: 1999
Movie
title: Cloud Atlas
released: 2012
DIRECTED
DIRECTED
Person
Director
name: Lana Wachowski
born: 1965

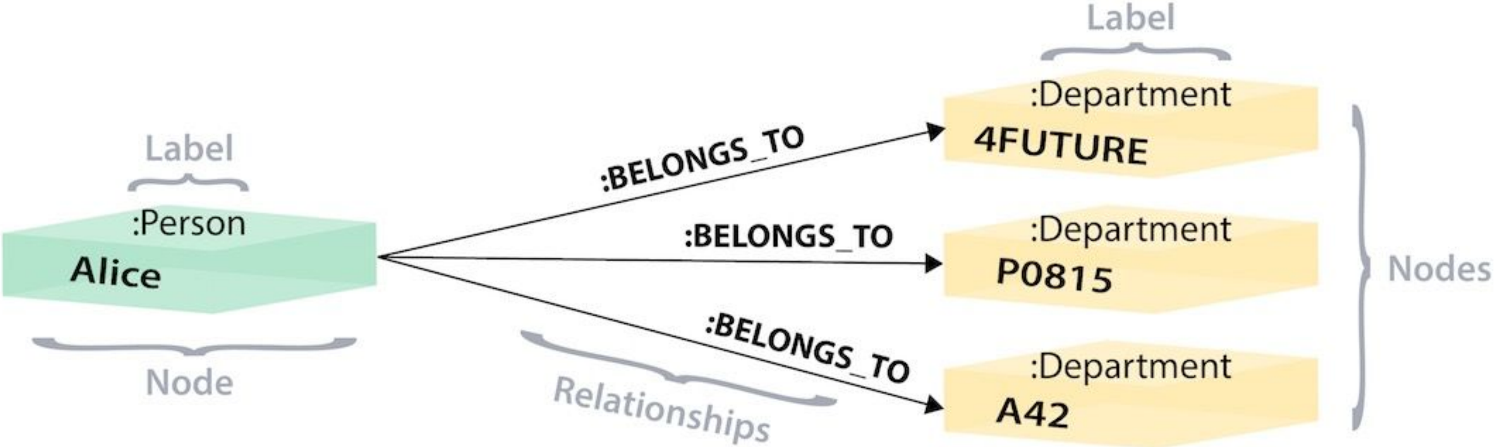
Relational database yeay, Graph DB, nayy.

Relational database yeay, Graph DB, nayy.

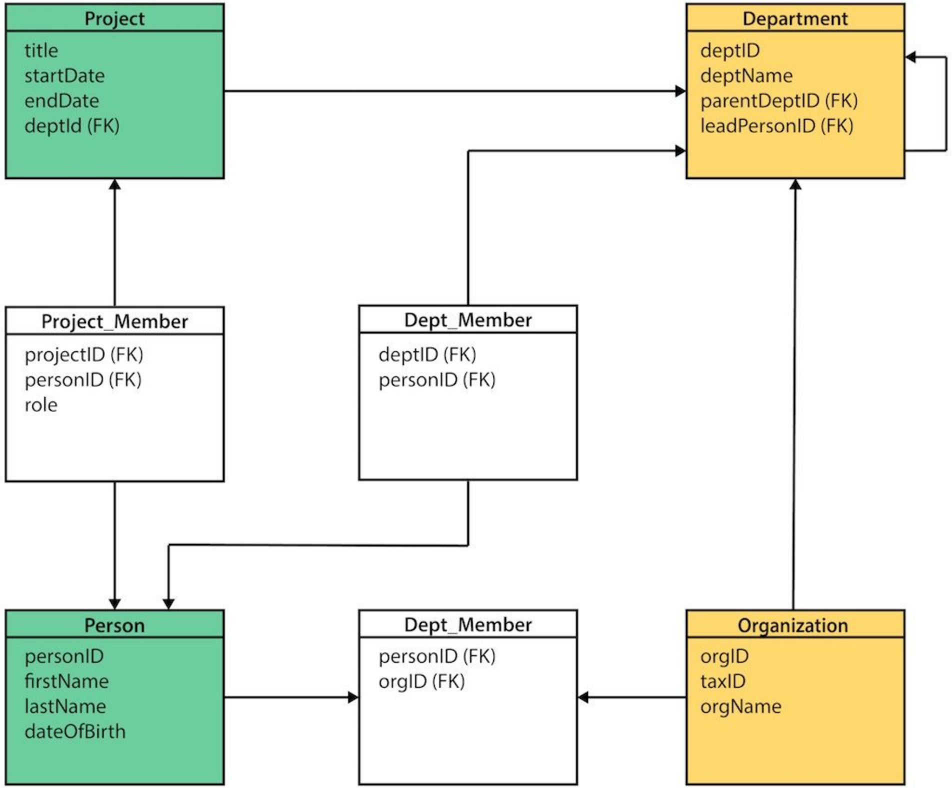


Associative Entity,
JOIN Table,
or Lookup Table

Relational database yeay, Graph DB, nayy.



Relational database yeay, Graph DB, nayy.



Project

title
startDate
endDate
deptId (FK)

Department

deptID
deptName
parentDeptID (FK)
leadPersonID (FK)

Project_Member

projectID (FK)
personID (FK)
role

Dept_Member

deptID (FK)
personID (FK)

Person

personID
firstName
lastName
dateOfBirth

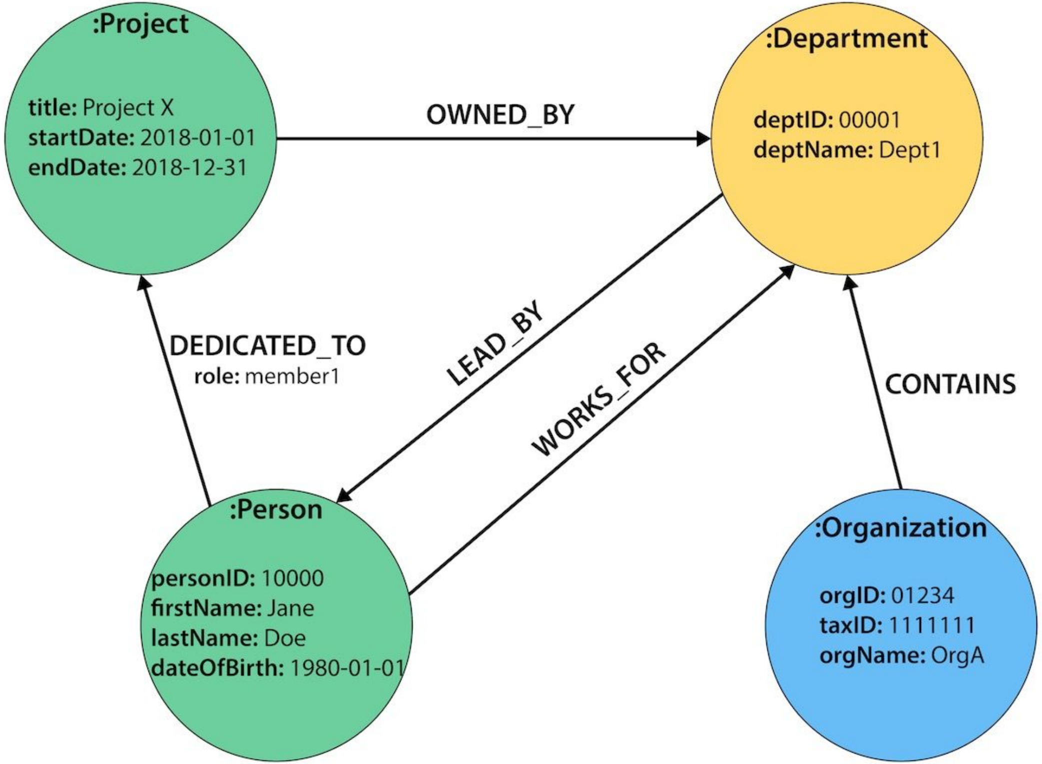
Dept_Member

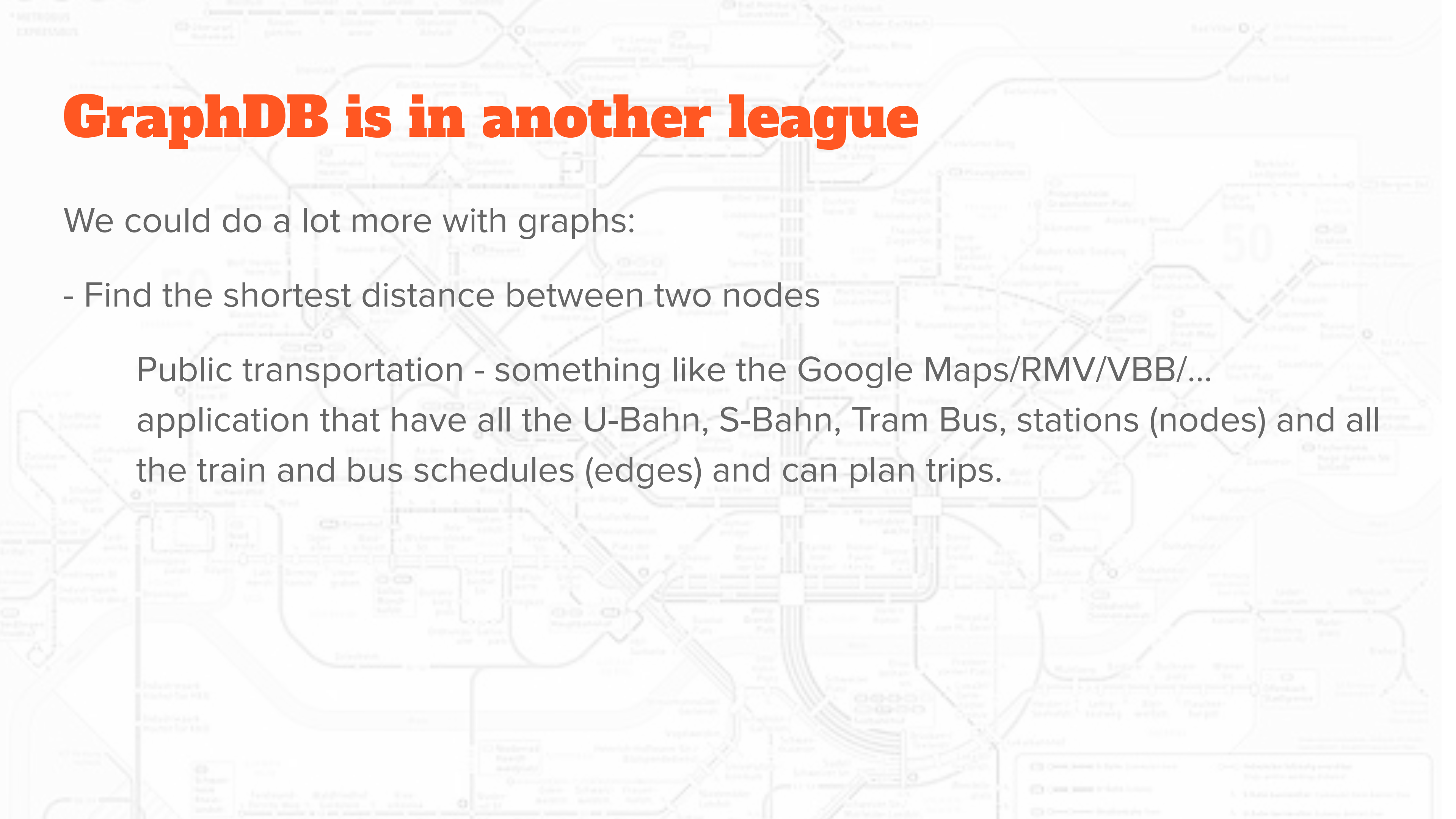
personID (FK)
orgID (FK)

Organization

orgID
taxID
orgName

Relational database yeay, Graph DB, nayy.





GraphDB is in another league

We could do a lot more with graphs:

- Find the shortest distance between two nodes

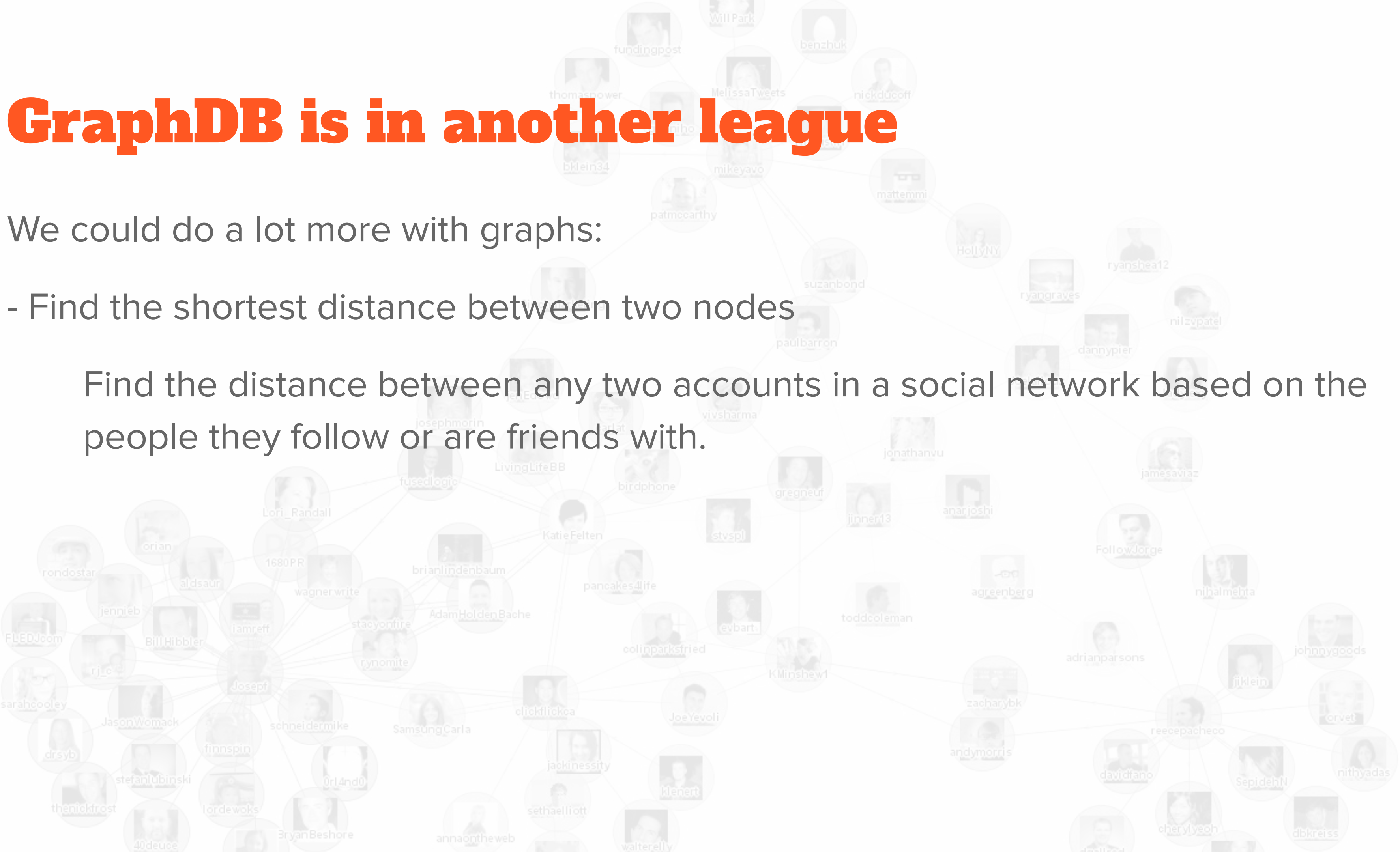
  Public transportation - something like the Google Maps/RMV/VBB/… application that have all the U-Bahn, S-Bahn, Tram Bus, stations (nodes) and all the train and bus schedules (edges) and can plan trips.

GraphDB is in another league

We could do a lot more with graphs:

- Find the shortest distance between two nodes

Find the distance between any two accounts in a social network based on the people they follow or are friends with.

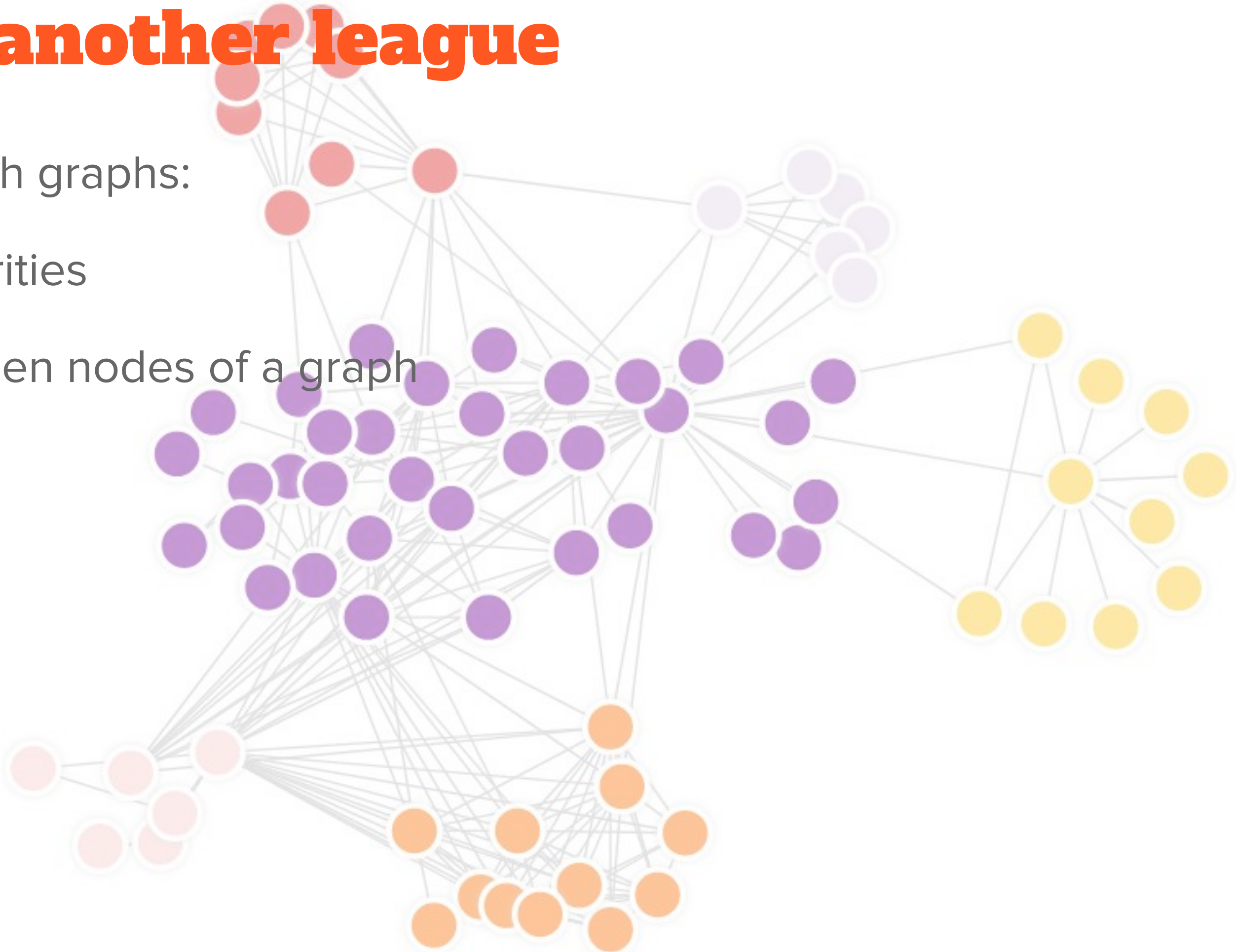


GraphDB is in another league

We could do a lot more with graphs:

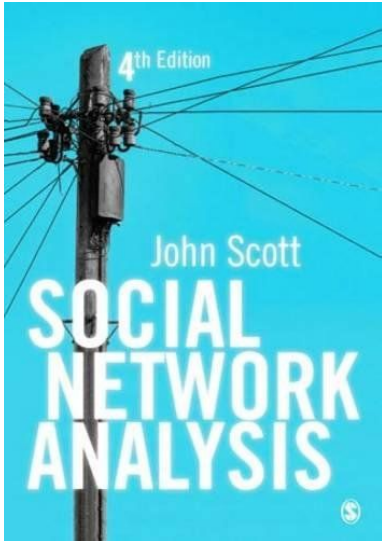
- Cluster nodes with similarities

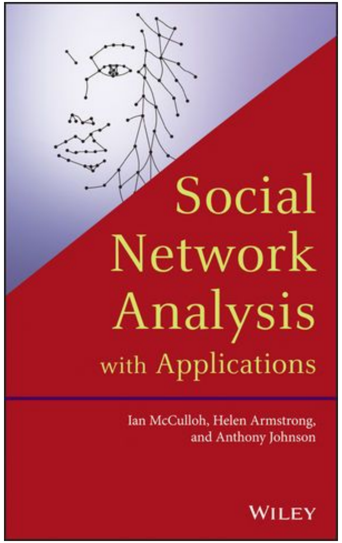
    Find similarities between nodes of a graph

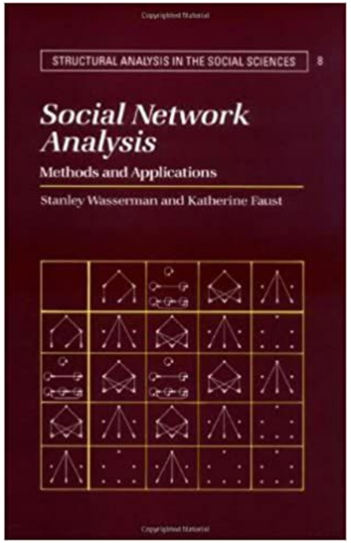


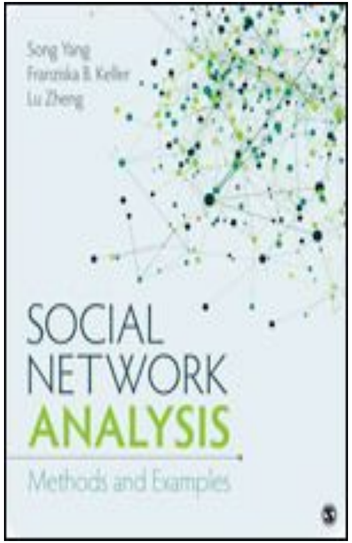
GraphDB is in another league

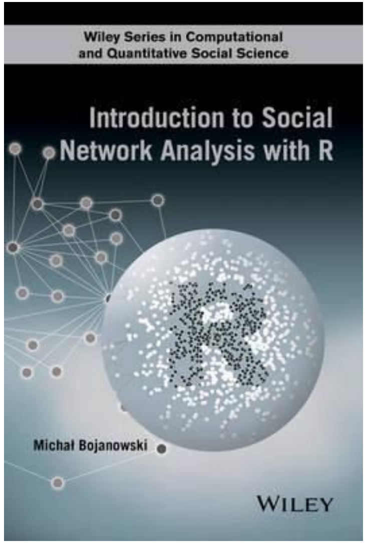
Actually there is a field called **SNA** (Social Network Analysis)









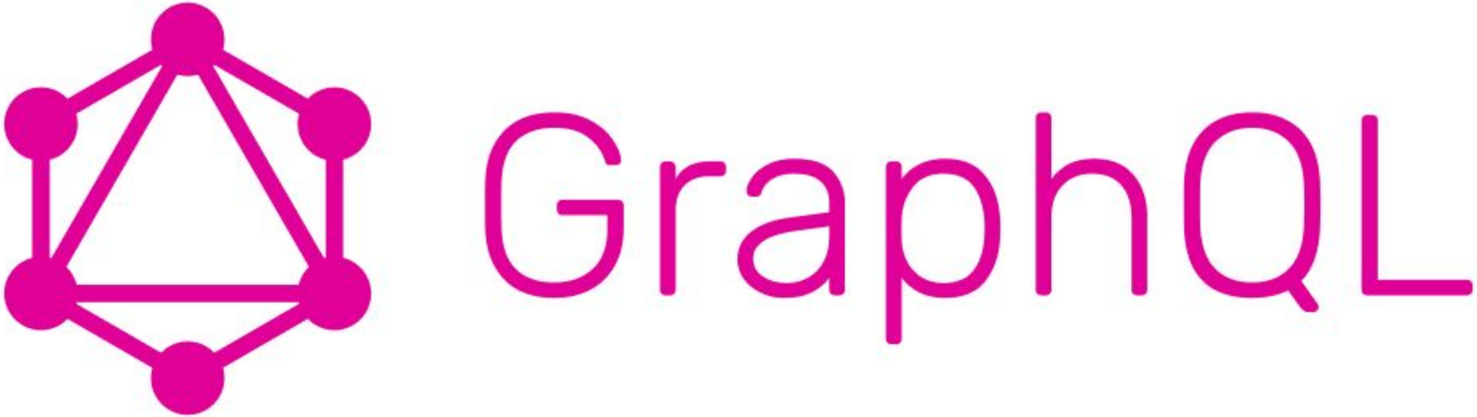

Wiley Series in Computational and Quantitative Social Science
Introduction to Social Network Analysis with R
Michał Bojanowski
WILEY

GraphDB is in another league

We could do a lot more with graphs:

- Or even a nicely implemented and optimized GraphQL API

https://medium.com/neo4j/introducing-graphql-architect-19b0f2035e21



So how could we use it?

So how could we use it?

The most famous graph database management system is **Neo4j**

So how could we use it?

The most famous graph database management system is **Neo4j**

And the most famous query language is **Cypher Query Language**

So how could we use it?

The most famous graph database management system is **Neo4j**

And the most famous query language is **Cypher Query Language**

We could use the **Neo4j** either with **HTTP** or **Bolt protocol**

So how could we use it?

The most famous graph database management system is **Neo4j**

And the most famous query language is **Cypher Query Language**

We could use the **Neo4j** either with **HTTP** or **Bolt protocol**

Oh, what is Bolt you ask?

Starting with Neo4j 3.0, it supports a binary protocol called Bolt. It is based on the PackStream serialization and supports the Cypher type system, protocol versioning, authentication and TLS via certificates. For Neo4j Clusters, Bolt provides smart client routing with load balancing and failover.

Lets see a small demo then

Thanks for your time and attention

Any questions?