Introduction to Graph query language Cypher

Véronique Gendner
e-tissage.net

Pink Programming Digital Sunday

P I N K
(programming)

Malmö, 25 Februrary 2024

Workshop presented *pro bono.* As a sponsor of Pink Programming, neo4j covered travel expenses

My path to graphs

- « Natural » languages : English, Chinese (French)
- Natural Languages Processing:
  - Linguistic formalisms *(trees)*
  - Data processing (Unix / Linux scripts)
- TheBrain : personal documentation structured in graph
- Data processing & visualization (Excel VBA)
- Web application full-stack dev for researchers (html css php js jQuery MySQL)
  - Modelling
  - UI design
- Graphs : DB & visualization (cypher python)

- let me first say a few words about how I got here today, talking about graphs
- my path to graphs started with what computer scientists call **natural languages**.
- **Languages** in their different aspects are the common thread in my different experiences : foreign languages, linguistics, NLP and of course formal languages for computer programming
- when I started NLP I was bold enough to believe I would do **NLP of Chinese** but it turned out that teaching computers a language I was not a native speakers of was too much for me
- at the time NLP was not just about LLM, we were working on **linguistic formalisms** that often used **tree structures** (we will be talking about that in a minute). I consider this the first step of my journey in the world of graphs
- in 2009, I started using TheBrain, which is a **personal documentation tool** that allows you to organize all your digital documents in a **graph structure**. It has a **graphical interface** that allows you to actually **trace relations with the mouse** and **visualize them on screen**. So this is how I started extensively using a graph structure, with an intuitive approach. The mathematical theory came in later, my first approach of graphs was intuitive
- at the time I was doing **data processing and visualizations for companies**
- then for 5 years I worked for a research team who was building a database for epidemiologists. my job was to **create and adjust web applications to help them process different kinds of data**. It was great because since the number of user was small, I could deal with the whole stack from user needs collection to front, including, db modeling and back end
- I first heard of Neo4j in a meetup in 2013 and rapidly became a huge fan of cypher's, so in 2019 I quitted my job to train myself and specialize on **graph technologies : both db and visualization**

Representing information: from ideographic writing to databases

- *In the beginning was the Word*
- -3300 CE first trace of writing (Egyptians and Sumerians)
- -2500 / -1000 CE alphabet (Phoenicians and maybe Egyptians)
- Papyrus (-3000 – 1000 CE)
- -150 CE parchment (paper in China 105 CE)
- 1200 paper reaches Europe, spreads during XIIIth century
- 1450 printing

Parchment delivery
Enluminure, mid XIIIth c.

- As amount of writings became large, the problem of finding information emerged
- ⇨ need to **organize information** in order **to be able to find it**
- ⇨ several **formal structures** emerged as most **commonly used**

- let's zoom out a little bit to see how human beings **represented information** over time and how the **invention of different storage medium** influenced the **quantity of stored information** which in turn, is **related to structuring and organizing information**
- Speech is the first way human beings communicate
- The problem with word is that
  - they are **gone after you say them** so **the information they conveyed is lost**
  - and you can shout over a valley, but if you have no technology, that is **pretty much as far as you can spread verbal words**
- Invention of writing allowed to **memorize information over time** and also to **spread information geographically**
- With the invention of the **alphabet** (that emerged in different places -> date unclear from what I could find, but not that long after first traces of writing), instead of having to learn hundreds of hieroglyphs or event thousands of Chinese characters, you can learn to write with just about 30 signs. On top of reducing the amount of memorization needed to write, the alphabet has had a **another major effect we will get back to in a minute**
- first traces of writing are on clay tablets. In the beginning, stone or metal was also used. But they are expensive and **not very practical**
- **Papyrus** invented by Egyptian around -3000 spread to Europe and has been used for about 4000 years, because it was easy to write on, stock and carry around, but it's production depended on one single plant that essentially grows in the Nile valley. Also, in humid regions, it was easily deteriorating
- **Parchment** made of animal skin was invented around the turn of the era and in Europe, it replaced papyrus in the VIIth century
- **Paper** invented in China, reaches Europe at the beginning of the XIIIth century and that has been a huge thing : much cheaper & convenient. It spread in Europe in just one century
- Mid XVth century Gutenberg **printing** took writings to the next level: that lead to a large multiplication of writings (previously, it had to be hand copied, typically by monks in monasteries)

Common formal structures

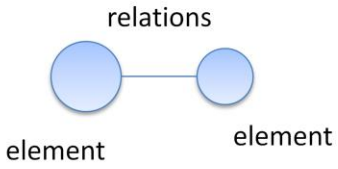
Définitions proposées par : Oxford Languages

structure

/ˈstrʌktʃə/

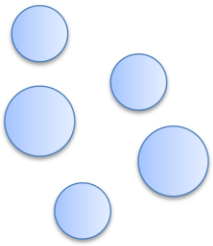
noun

1. the arrangement of and relations between the parts or elements of something complex.



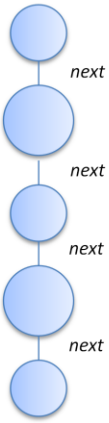
- **Structure** is about the arrangement of and the relations between parts or element of something complex
- So let's keep in mind that structure is about **elements related to other elements**

Set



In the early XXth c., the first phone switchboards had just a couple tens of persons (doctor – pharmacists). Operators did not even need to sort the names in a specific order, they could find people just by browsing through the unordered bunch of names to find the person who was requested

List



Most simple organization

- alphabetical order
- chronology
- …

Formal advantages :

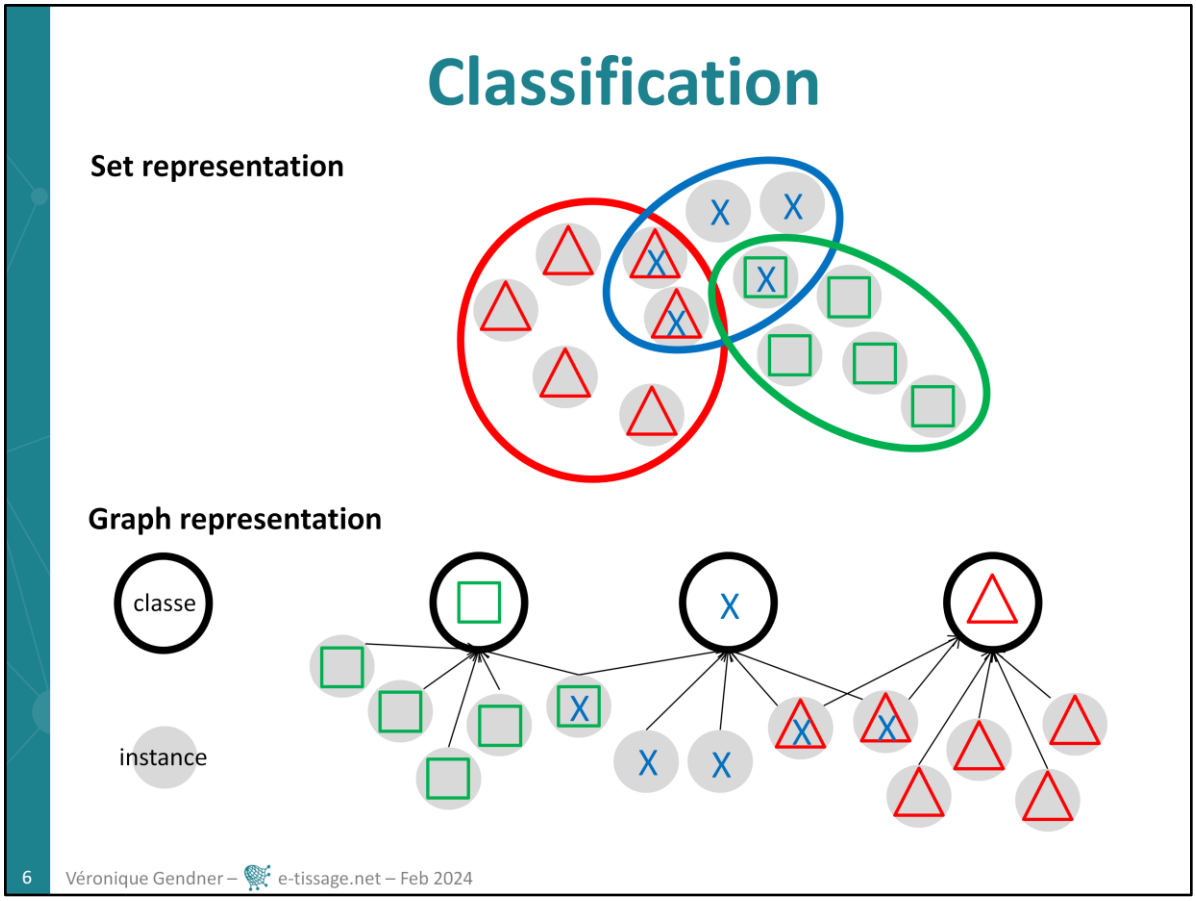
- ordered
- has a beginning / end : browsable in a finite time
- sortable

Street guides then phone books, etc ordered in alphabetical : massively used in the XIXth & XIXth centuries

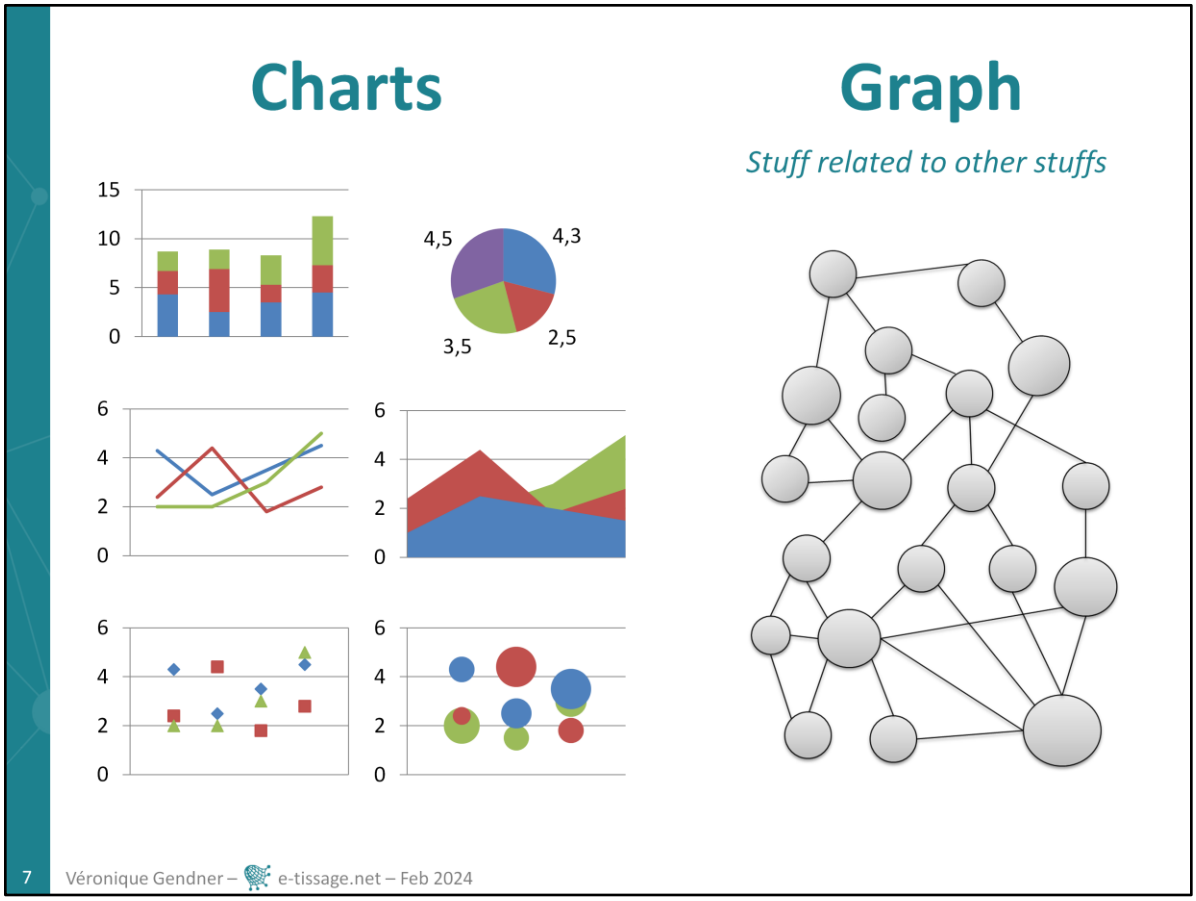
- If you have a limited number of elements, you can have a **set** of stuff, with **no organization**
- but as the quantity of information increases, you will need **lists** to find what you are looking for
- in the XIX et XXth century, the **alphabetical order** has become a reference that anyone could use to find what they search for with **no other knowledge** of what they are looking for than its name

(- as a side note, it is a **very western culture centered reference**, with the influence of western culture it reached countries with non alphabetical languages but it not as predominant there)

- In a list, there is an **implicit next relation** between elements
- lists have **formal advantages** :
  - ordered,
  - can be sort
  - and importantly, they have a formally obvious beginning and end

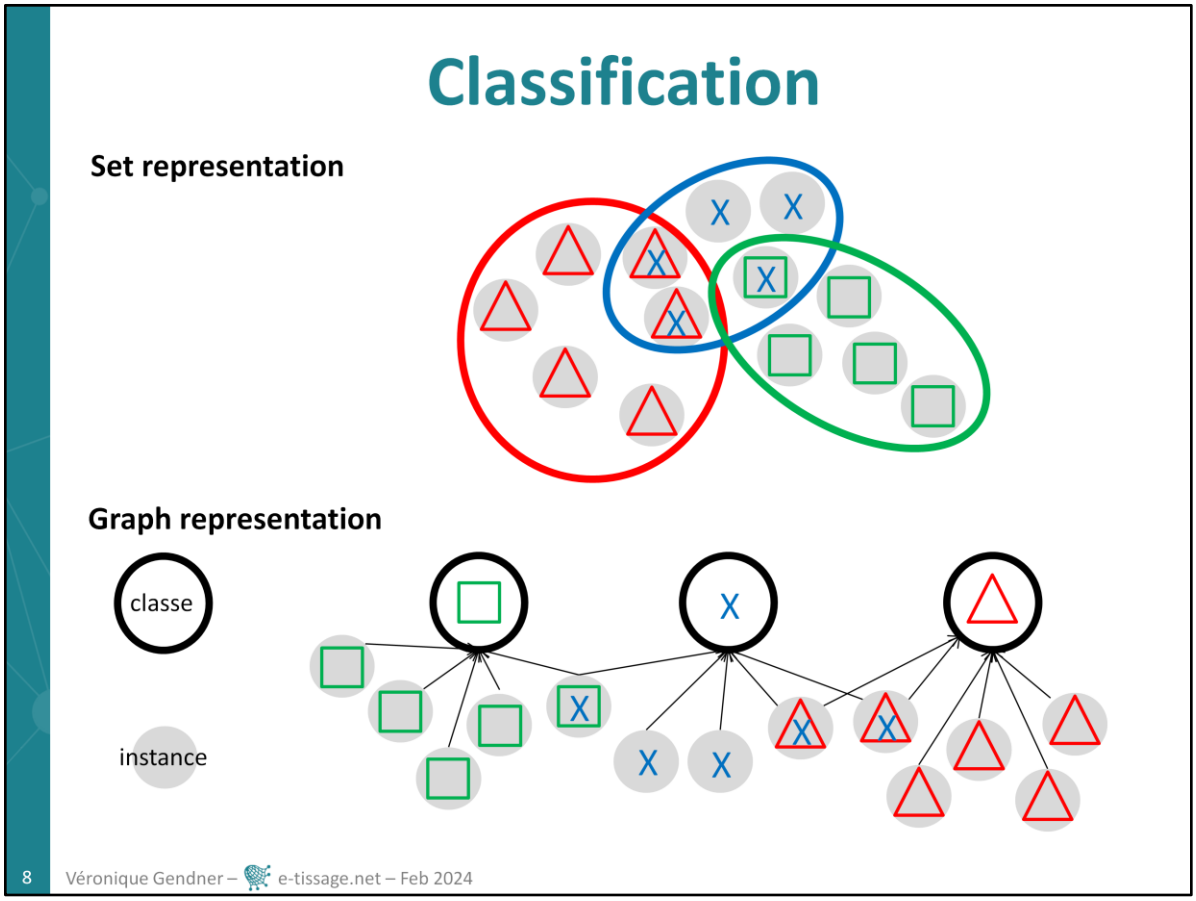


- When the **list** gets **too long**, instead of having everything in just one list
- You probably want to **classify** your items, in order to be able to find them
- For example, take those grey dots, that can have properties, represented here by a red triangle, blue cross or green square
- A **common representation of a classification** is with **sets**, here, the red, blue and green potato shapes
- But **a classification can also be represented a graph structure**
- [graph def]



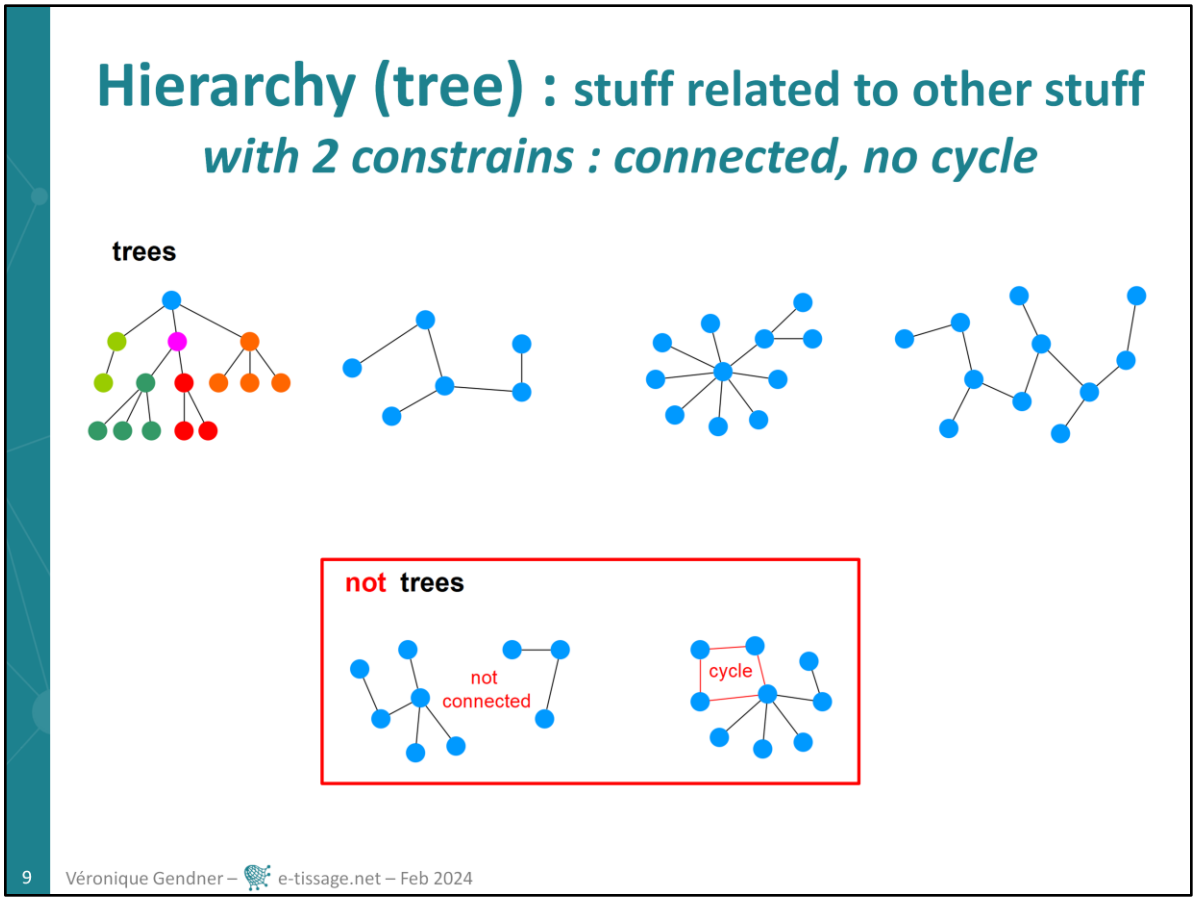
- To make sure we understand each other : I've used the word graph several times already, what I mean is what you have to the right

- To left is what I would call **charts**, it is sometimes called « graph » but when I'm saying **graph** I mean what is graphically represented right of the slide: stuff related to other stuffs



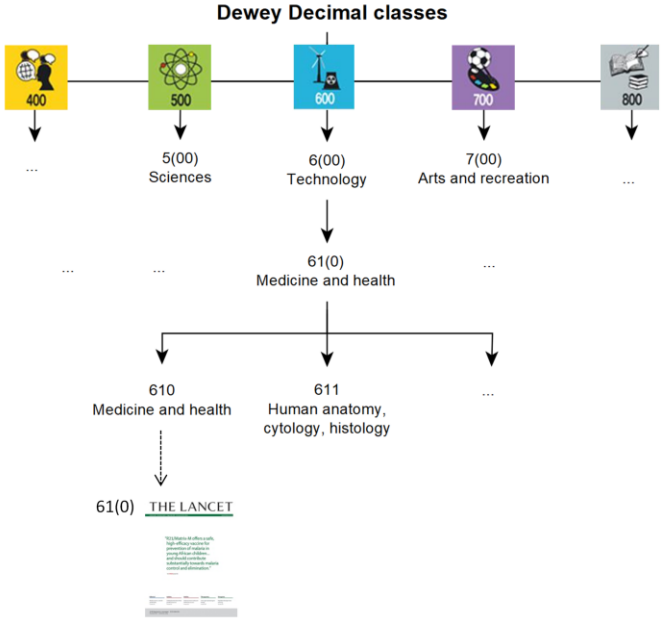
Back to the representation of classification
- you can represent the class by a node, here with the thick border
- and the you represent the classification by linking items to the class nodes

formally you are representing the same thing than with a set representation



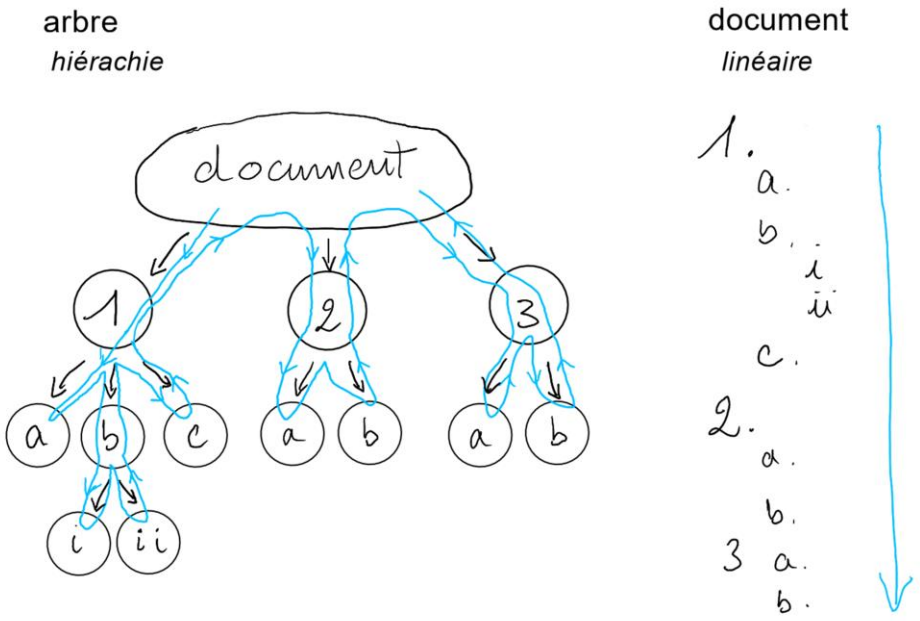
- Now if you have a lot of class, you'll probably want to **organize those classes.** This is **often done with a hierarchy**
- If you look at the top left typical representation of a tree, you could say that it is about making a dark green and a red collection, then putting them together to make the pink collection, then put the pink with the light green and orange classes to build a blue collection. that is a way of **organizing collections**.
- Let's first focus on the formal constrains of what is called a tree : it's stuff related to other stuff, but with **2 constrains**
  - Must be **connected** : if you have several pieces, it is not a tree
  - **No cycle** : what a cycle is is very easy to understand if you look at the bottom right figure : in a tree, they can not be relations that form a loop

Hierarchy (tree)



- **Hierarchies are used everywhere**
- here for **example** is the **Dewey classification** that is the organizing principle in lots of libraries
- To the left, the Star representation of the first level of the tree, with 9 main categories
- To the right some detail of category 600 – technology – down to the medical journal « The Lancet » coded 610 because it is classified in generality of the « Medicine and Health » subclass of technology

Hierarchy (tree) : can easily be linearized



- You use a tree structure every time you make an **outline for a document** :
- To the right, the **typical outline** of a document and to the left, its **graphical representation**
- What can be seen here is the big advantage of a tree structure : it can **easily be linearized** : that is if you start from the root, if you go always to the left when you can and back up when you can't, you are sure to go through all node and only once per node
- On the **contrary**, as we've seen, in **unconstrained graphs**,
  - you can have cycles, so **loop around** and therefore pass the same node several times
  - an unconstrained graph has **no obvious beginning or end**, so it is more complicated to build a strategy to traverse nodes and make sure you have them all and you do not loose yourself into cycle or pass several time on certain nodes

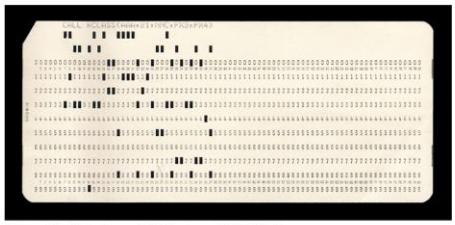
Table

Excel spread sheet ancestor !

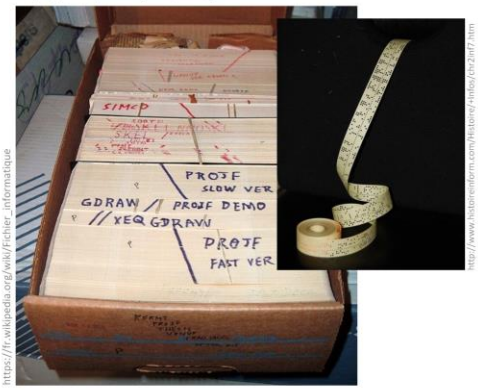
Earliest known trace of writing -2700 / -2600 AC, Administrative clay tablet used to keep track of stocks
https://collections.louvre.fr/ark:/53355/cl010120699

- the last commonly used formal structure we will be talking about is the **table**
- quick jump back in time, this is the earliest known trace of writing. It is **a clay tablet for administrative use to keep track of stocks.** So this looks like the ancestor of an Excel spread sheet to me !
- Which shows the need for structuring information is very old, because it is very much related to our cognitive processes : **we need to organize stuff,** that is to **relate them, to be able to think and understand the world we live in**. That is how we human beings are

Organizing computer data



https://fr.wikipedia.org/wiki/Carte_perfor%C3%A9e



https://fr.wikipedia.org/wiki/Fichier_informatique

- Until the 60-70s files contained information but files were **unrelated**
- 1970, Edgar Codd

  A Relational Model of Data for Large Shared Data Banks

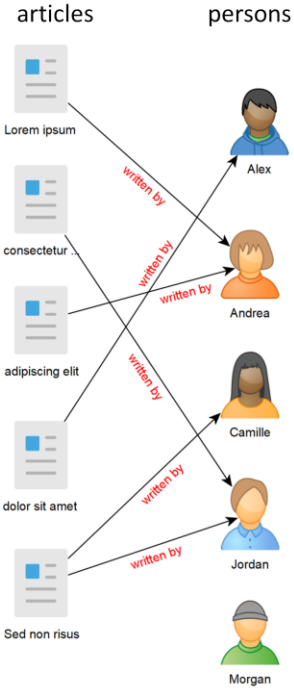
⇨Relational Databases

- Started to be used in 70s-80s (IBM SQL)
- Developed in the 90s
- MySQL v 4.0 in 2001 (David Axmark)

- fast forward to the beginning of computer science
- Until the 60-70s files contained information but files were **unrelated.**
- in 1970, Edgar Codd came up with the idea of *A Relational Model* which lead to the developing of **relational db**
- MySQL which is one very widely used kind of relational database was created by a Swedish guy.

As you know, Neo4j was also created by Swedish guys.
You Swedish people are good at databases !

Relational databases are structured in tables



articles

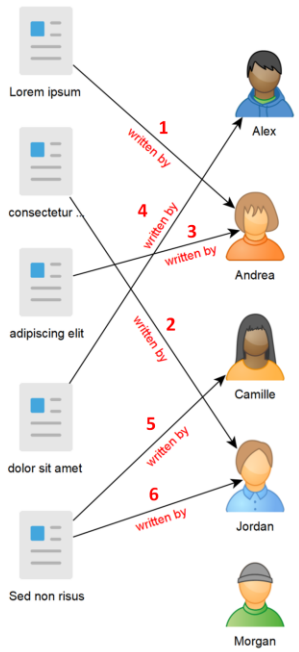
Id	titre	date
1	Lorem ipsum	23/06/16
2	consectetur ...	24/01/24
3	adipiscing elit	04/05/18
4	dolor sit amet	09/09/20
5	Sed non risus	19/04/21
6	...	

persons

Id	login	pwd	created
1	Alex	*****	23/06/16
2	Andrea	*****	24/01/24
3	Camille	*****	04/05/18
4	Jordan	*****	09/09/20
5	Morgan	*****	19/04/21
6	...	*****	

- if you have articles and people, and you want to represent the relations that shows who wrote which article(s), this is how it is done in a **relational database** – for example WordPress blogs work like this
- you will have a **table with one line per article**, an id for each article and all related information (title, date, content,…) in columns
- the same for **persons**

Relational databases are structured in tables



written by

id	idArticle	idAuthor
1	1	2
2	2	4
3	3	2
4	4	1
5	5	3
6	5	4
...	...	...

Relations computed at query time

articles

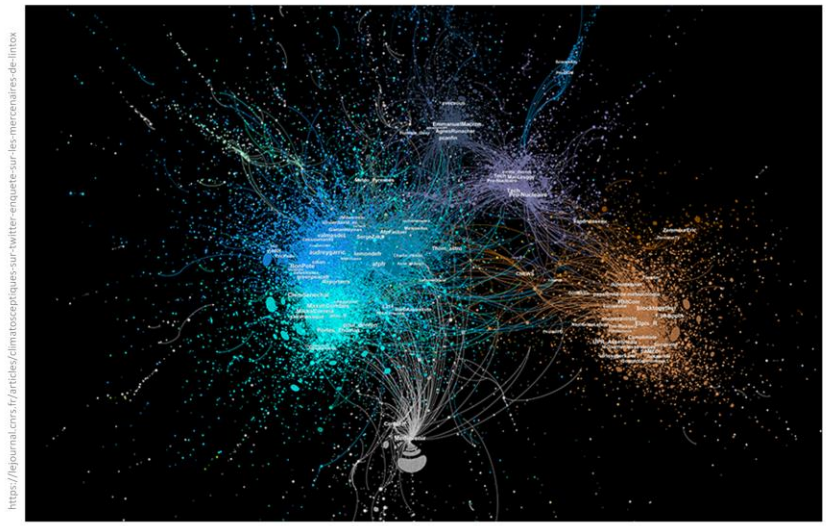
Id	titre	date
1	Lorem ipsum	23/06/16
2	consectetur ...	24/01/24
3	adipiscing elit	04/05/18
4	dolor sit amet	09/09/20
5	Sed non risus	19/04/21
6	...	

persons

Id	login	pwd	created
1	Alex	*****	23/06/16
2	Andrea	*****	24/01/24
3	Camille	*****	04/05/18
4	Jordan	*****	09/09/20
5	Morgan	*****	19/04/21
6	...	*****	

- the **relation** « written by » is **also represented by a table** : you have one line per relation, with the Id of the article and a corresponding author
- in relational databases **relations are computed at query time**

Not for tables…



52 000 twitter accounts, links are RT (who RT who)
used by researchers who analyze the structure and tactics the climate change denialist on twitter

- Long chains of relations
- Highly connected data

- If this is how your data looks like, you do not want to use tables

- This a representation of 52 000 twitter accounts, links are RT (who RT who). It is used by researchers who analyze the structure and tactics the climate change denialist on twitter

- **tables are not an appropriate representation** if **data is highly connected** and/or **contains long chains of relation**

Amounts of available data

- 710 Archbishop Boniface was a book lover. … He had 50 *
  100 KB / book
  (200 pages of 500 char)
- 1978

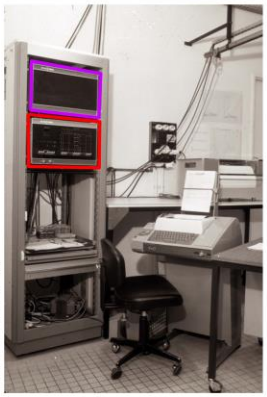


Photo Jean-Paul Gendner, 1978

How much memory ?

16 KB *in the computer*

yes, that is 160 lines of 100 characters

+ an extra 8 KB
in additional memory

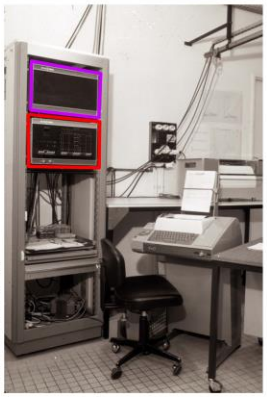
= 24 KB

* *A Place For Everything*, Judith Flander,p, 47
** https://fr.wikipedia.org/wiki/Big_data

- this slide is to give you an impression of the **crazy increase in information storage capacity**, because as we said before, the more information stock, the more you need efficient search strategies to find what you need, when you need it
- Archbishop Boniface was a book lover. … how many books do you think he had ?
- He had 50. more than the Pope and monasteries at the time
- in 1978, as I was learning my multiplication tables in primary school, my father was using this computer, in the research lab he worked for
- how much memory do you think it had ?
- 16 + 8 24 KB
- if you consider that back in 710, a book had maybe 200 pages of 500 char
- the 1978 computer my dad wrote his PhD on could not even contain ¼ of a 710 books

Amounts of available data

- 710 Archbishop Boniface was a book lover. … He had 50 * books
  100 KB / book
  (200 pages of 500 char)
- 1978



How much memory ? X 10 000 000

16 KB *in the computer*

yes, that is 160 lines of 100 characters

+ an extra 8 KB
in additional memory

= 24 KB

X 700 000

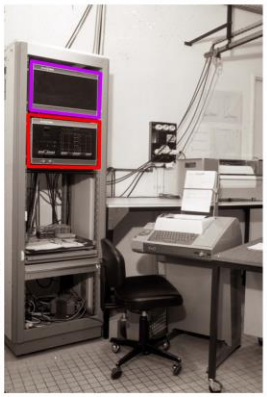
- 2020s Recent personal computers have 16 Go (16 000 000 KB) of RAM
  + disk space in Tera bytes 1 000 000 000 KB

* *A Place For Everything*, Judith Flander,p, 47
** https://fr.wikipedia.org/wiki/Big_data

- recent personal computers now have 700 000x more than this very expensive research computer in 1978
- today, 1 tera is easily accessible to many people - 10 M° more than what the riches guy in terms of books had in 710

Amounts of available data

- 710 Archbishop Boniface was a book lover. … He had 50 *
  100 KB / book
  (200 pages of 500 char)
- 1978



How much memory ? X 10 000 000

16 KB *in the computer*

yes, that is 160 lines of 100 characters

+ an extra 8 KB
in additional memory

+ compression !
(200 pages pdf only 5M)

= 24 KB

X 700 000

- 2020s Recent personal computers have 16 Go (16 000 000 KB) of RAM
  + disk space in Tera bytes 1 000 000 000 KB

2010 1 200 000 000 000 000 000 000 bytes produced worldwide **
2020 64 000 000 000 000 000 000 000 bytes

* *A Place For Everything*, Judith Flander,p, 47
** https://fr.wikipedia.org/wiki/Big_data

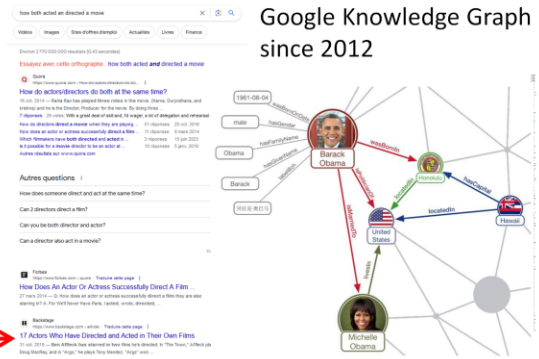
- do not remember the exact figures, they are estimations that are difficult to make in a meaningful way : I have not even taken into account compression technics which makes the factor even bigger
- just want to give you an impression of the craziness in which the **capacity to store information raised.**
- over our life times (i.e. over the 50 years of my lifetime and even more rapidly over the last 20-30 years) **it went exponential (i.e. even more crazy…)**

Indexed search / Search by relations

- Index keyword document

1998 : Google search engine official launch

Google Knowledge Graph since 2012



- **What if you do not know the keyword / name** ?
- What if you are looking for the name of someone who both acted and directed a movie ?
- What if you want to find fraudsters in a database of financial transaction ?

Relations aka a graph structure allows for more search strategies than simple indexed search

- so, it's well and good to have lots of data but you need to be able to find what you need when you need it
- automatic indexed **search was a huge leap forward** for this.

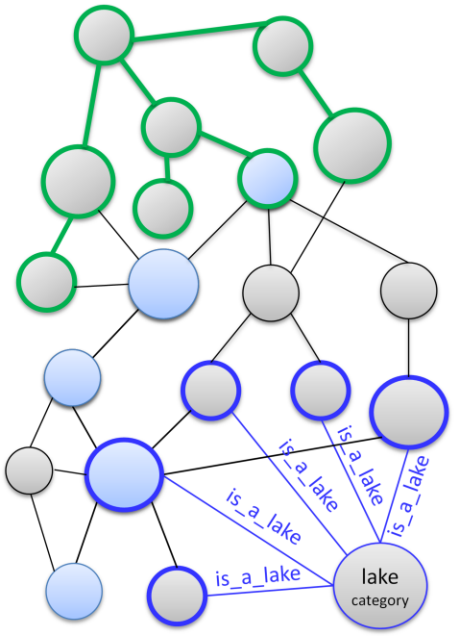
it associates keywords with corresponding relevant documents but

- **What if you do not know the keyword / name of what you are looking for** ?
- What if you are looking for the name of someone who both acted and directed a movie ? (We will do that in a moment)
- I'm guessing if google is not that bad (3rd-4th in results) it is because since 2012, they use a KG, that is knowledge represented in a graph structure, to complement the automatic indexing algorithm

- Now if you have a data base of financial transactions, indexed search can not help you find fraudsters. Graph DB can

Graph : stuff related to other stuff

no constrain

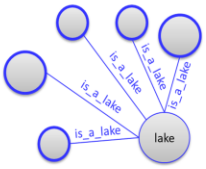


Common formal structures

List

Hierarchy

Categorization



Table

sourceLabel	versionLabel	id	name	location
BDTOPO	v2021	1	Cascade blanche	point(5.3014,45.4565)
ENEDIS	v2021	2	Pylon 456872112	point(5.3490,45.4568)
Camp2camp	v2022	3	Source des chiens	point(5.4920,45.5618)
BDTOPO	v2021	4	Canal du moulin	point(5.5420,45.6458)

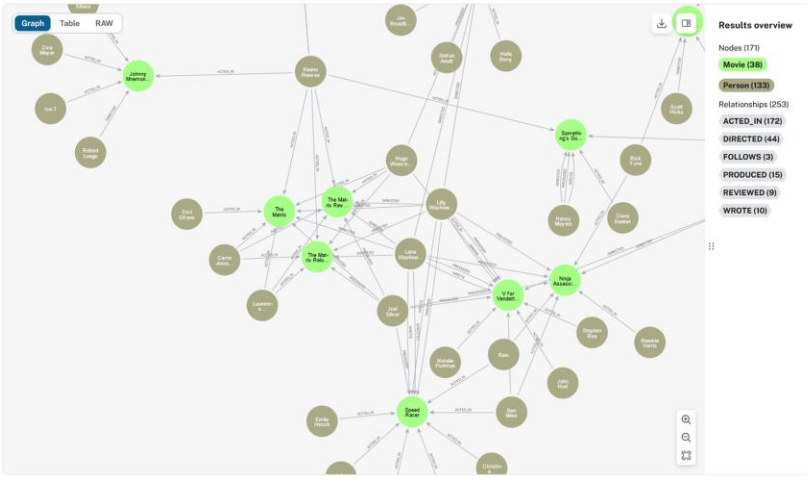
- let's wrap up on this commonly used formal structures part
- Graph = **no constrain** on **topology** : connected or bits and pieces, no constrain on **relations** (you can have cycles, as many kinds of relation you want)
- Difficulties :
  - **complexity can be difficult to grasp. V**isualization help on that aspect
  - finding a **strategy to traverse it in a finite time** is tricky, just deciding where to start and when you are done can be a challenge
- Neo4j has a magic recipe (!) for machines to do so
- but for us humans it can sometimes be useful backing up onto the common formal structure we are familiar with
- the good news is that **all common formal structure we are familiar with can be represented in a graph**

- you can get the **table** if your **nodes and relations have properties** and that's what you can do with labeled property graph database

Graph structure pros

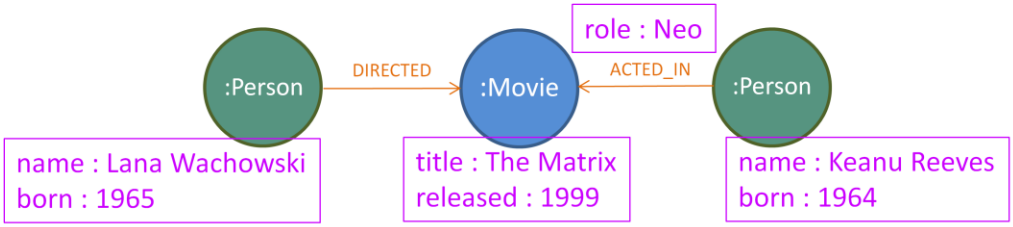
- Graph structure allows for **more search strategies** than simple indexed search
- It is the best suited formal structure to **analyze** or **structure** data without being trapped in a preconceived formal structure (table, hierarchy)
- The absence of strong structural constrain allows to use it to **centralize data structured in different ways**
- Can represent both **detailed** information and **aggregated** (more synthetic) information
- Fits for **highly connected** data or when you need to find **long chains of relations**
- Can both be **processed by machines** and easily grasped by **humans through visualizations**

Graph databases



Neo4j Label Property Graph DB vocabulary

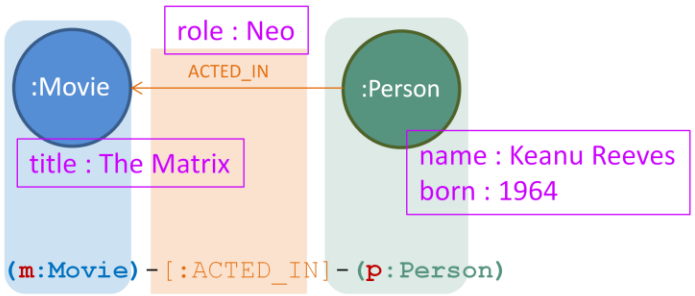
- Nodes with one or several labels
- Relations with one (and only one) type
- Properties on nodes and relations



Relations are hard encoded as first class citizen, when importing data

- Here are the Lego bricks of information you can play with
- contrary to relational databases, native **graph databases hard encode relation when importing data,** which make them **very efficient at traversing relations.**

Cypher : *visual* graph query language




```
CREATE (:Movie {title:"The Matrix"})
       -[:ACTED_IN {role:"Neo"}]-
       (:Person { name : "Keanu Reeves" , born:"1964"} )
```

How to get the movie Keanu Reves acted in ?

```
MATCH (m:Movie)
      -[:ACTED_IN]-
      (p:Person { name : "Keanu Reeves" } )
RETURN m.title
```

25 Véronique Gendner – e-tissage.net – Feb 2024

- in early versions of Neo4j graph databases, you had to use java procedures to query the databases
- until **Andres Taylor** initiated and lead the design of the **cypher language**
- inspired by what people where using on forums where they discussed graph modeling, rounded parenthesis () where chosen to represent nodes, and two dashes with the type in square brackets represent relationship, which makes **cypher a *visual* language, with which it is very easy to model and think in graphs**.

Different places to run Neo4j

- Desktop = app that contains
  - Browser : running cypher queries
  - Bloom : graph visualization
  - Neo4j ETL tool
  - etc

  https://neo4j.com/download/

  Enterprise edition
  Free for non commercial use
- Aura = cloud version

  https://console.neo4j.io

  Enterprise edition
  1 free database
  for up to 200k nodes and 400k relationships
- Sandbox

  https://neo4j.com/sandbox/

  Enterprise edition
  Temporary (3-7 days) preloaded databases
- Self-hosted

  https://neo4j.com/pricing/

  Free community edition
  Paid Enterprise edition

- The workspace (Browser + Bloom + Importer) can be connected to an instance running anywhere
  https://workspace.neo4j.io/workspace

Hands on !

https://www.v2belleville.eu/tutos/cypher-intro.html

- Do not read these 🫵 now !

  for later reference : more about MATCH in the documentation

  It is for further reference if you want to continue your journey in the world of graphs from home

- The database you will be creating today will stay online for 3 days (you can extend it or create new ones for free).

  So you can get back to this curriculum at home

Introduction to Cypher
Hands on take away

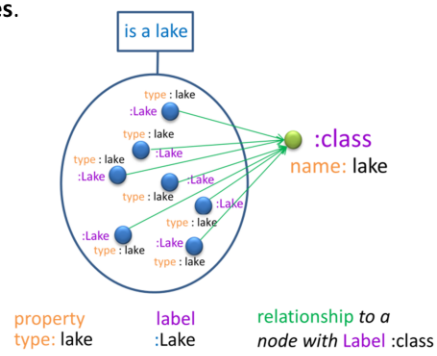
- Graph pattern matching is like writing a filter “shape” : you specify a configuration of **nodes**, **properties** values or existence (on nodes or relations), node **labels** can also be in a pattern and **relations** between node can also be part of that “shape”.

  The query engine will sort of pass this filter “shape” through the graph, to grab corresponding portions of the graph,

  In the return clause, you pick in the matched pattern, whatever it is you want to display.

- This way you can extract **lists**, **tables** as well as **portion of the graph**

- There are **3 ways to categories nodes**.
  With :
  - a property
  - a class node (:Genre)
  - a label

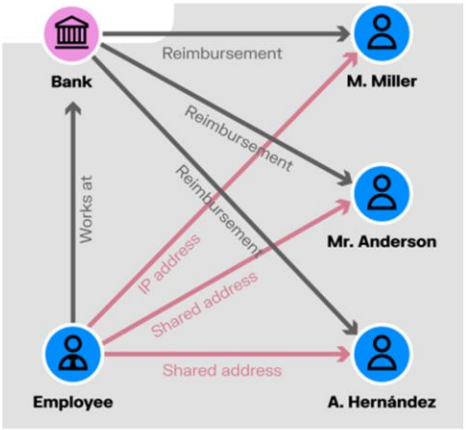


Graph pattern to detect fraud

Assisted chargeback fraud

Chargeback fraud is a common problem for banks, and one that poses a big opportunity for collusive insider agents. Someone working at a financial institution may have insider information on how to "game" the system. They might know how to obtain a fraudulent reimbursement, for example, and share that information with friends and family so they can profit.

Graph can show you the social and personal connections that indicate who the individuals taking advantage of the system may be colluding with inside your organization. Does an employee share a home address with someone who has committed fraud? What about their social media connections? Individuals colluding on a chargeback scheme may also share an IP address or devices, such as the smartphone or computer used to access the banking app. A graph will enable you to quickly visualize these kinds of relationships.



https://resources.linkurious.com/ebook/internal-fraud

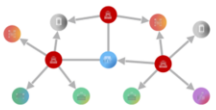
demo https://www.bigmarker.com/linkurious/Solving-the-FinCrime-efficiency-puzzle-How-Graph-Technology-Entity-Resolution-can-help

- extracted from a white book just released by Linkurious, one of Neo4j partners in France. They have a great tool for analysts to inquiry data with a graphical user interface to explore a database structured in graph

- **cycle detection**, including in relatively long chain of relations often is an **alert** that something dodgy like **money laundering or corruption** is going on

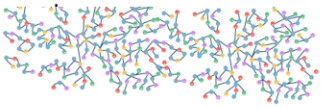
Different technical ways in which graphs are useful

- Graph pattern matching (*local pattern*)
  - Relation transitivity
  - Pathfinding
  - Cycle detection



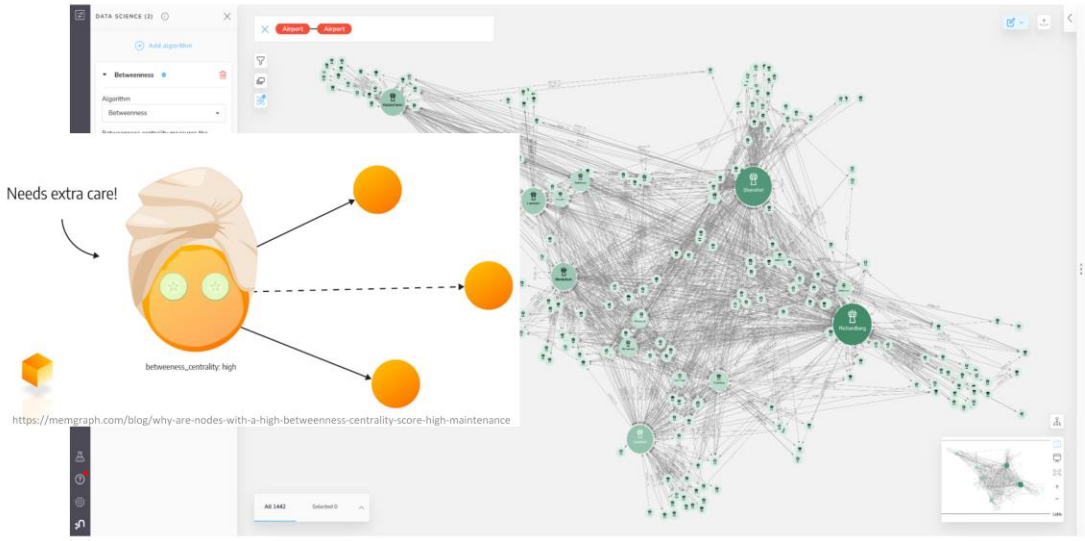
- Visualizations, Graph GUI & Graph Layout Algorithms
- Graph Data Science (GDS) algorithm (*large scale patterns*)
  - Node centrality
  - Community detection, etc …

To characterize the general shape of the



- let's wrap up by zooming out from what you've done today and all that can be done with graphs
- **local graph pattern matching** like we have practiced today is just one way to use graph structure.
- as we have briefly seen with the fraud detection example, graph **Graphical User Interface, with interaction functionalities and layout algorithms** help **analysts explore graphs databases**
- they are tens of Graph Data Science algorithms, that can calculate for example node centrality (= importance), nodes communities or similarities. **GDS algorithms detect large scale patterns**

Betweenness centrality detection for supply chain



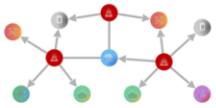
https://neo4j.com/developer-blog/supply-chain-neo4j-gds-bloom/

- there are **different kinds of centrality** (= **importance**) detection algorithms for nodes in a graph.
- **Betweenness centrality** for example represents how important a node is for transiting traffic. It is used in transportation and supply chain industry, to visualize important nodes because of the traffic that goes through them
- The figure captures this very well. It also made me laugh and I believe it is because it **relates two imaginary worlds that I've always seen separate** : the tech world with it's very male base imaginary (lot's of pizza bier references…) and skin care, that is more commonly associated to women world's …

For the gender equal (tech) world to happen, we need more representation like this one !

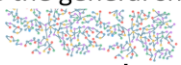
Different technical ways in which graphs are useful

- Graph pattern matching (*local pattern*)
  - Relation transitivity
  - Pathfinding
  - Cycle detection



- Visualizations, Graph GUI & Graph Layout Algorithms
- Graph Data Science (GDS) algorithm (*large scale patterns*)
  - Node centrality
  - Community detection, etc …

To characterize the general shape of the graph



- Knowledge Graph (KG), meta relational layer over data

To make logical inferences

- etc..
  - Retrieval Augmented Generation (RAG) for LLM
  - Nodes embedding for machine learning
  - State machine
  - …

- the term *Knowledge Graph* is sometimes used as a synonym of graph database, but **most of times, *Knowledge Graph* means there are some constrains on the graph, in term of relations types and or node topology**.
(i.e. (instance)-[:isA]->(subClass)-[:isSubClassOf]->(class)).
Such constrains allow to compute new information with logical inferences
(i.e. if (a)-[:isSubClassOf]->(b)-[:isSubClassOf]->(c) then (a)-[:isSubClassOf]->(c))
- a currently very hot topic is the use of graphs for Retrieval Augmented Generation (RAG), in order to frame LLM hallucinations with knowledge represented in a graph structure
- nodes embedding (= node vectorization) is used for machine learning
- state machine are a way to represent the **state of an automaton with operation that will make states evolve**. This is used to monitor tickets in a IT system, to deal with issues or updates

Graphs databases are beyond databases

- Not just about stocking information, analysis and processing can also happen in here,
  taking advantage of the graph structure
- Graph are another way of thinking / writing
  after procedural programming, Object Oriented Programming
  also beyond hierarchical thinking

In Graph database, the keyword is not database, it is GRAPH

- Computer scientists who have been trained in computer science schools tend to think that a database is about stocking information. Analysis and processing is supposed to happen before or after, but not in the db.
- I've experienced it can be difficult to have them think outside this box and realize that putting all available data in a graph database is not just about stocking but above all about **taking advantage of a graph structure**
- **In graph database, the keyword is not database, it is graph**

Learn more

Hands-on

- https://neo4j.com/sandbox/ : cloud environnement with preloaded datasets
- https://neo4j.com/cloud/platform/aura-graph-database/ : cloud service 1 DB free (*for up to 200k nodes and 400k relationships*)
- https://neo4j.com/download/ Desktop

  :play or :guide

Courses

- https://graphacademy.neo4j.com/

Documentation

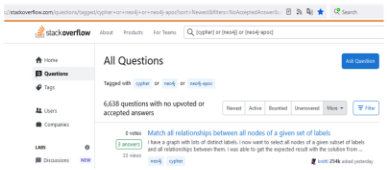
- https://neo4j.com/docs/

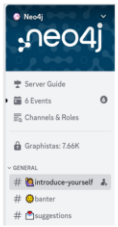
Community

- Discord https://discord.gg/neo4j
- https://community.neo4j.com/

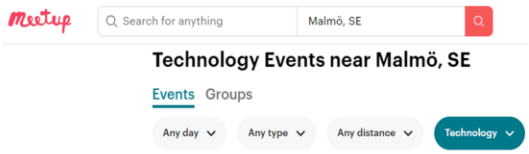
Make the gender equal (tech) world happen !

- Follow your interests in tech :
  attend meetups*, conferences, online tech communities
  (discord servers, https://stackoverflow.com/) , etc





https://www.meetup.com



- For **the future to be a mixted society**

***You* need to take *your* place in it**

Take the floor !

- attend general public tech gigs
- https://www.pinkprogramming.se/ is great (thanks for the invite !) but at some point you need to leave it's comfortable cocoon and jump into the world !
- For the future to be a mixed society

***You* need to take *your* place in it**

Take the floor !