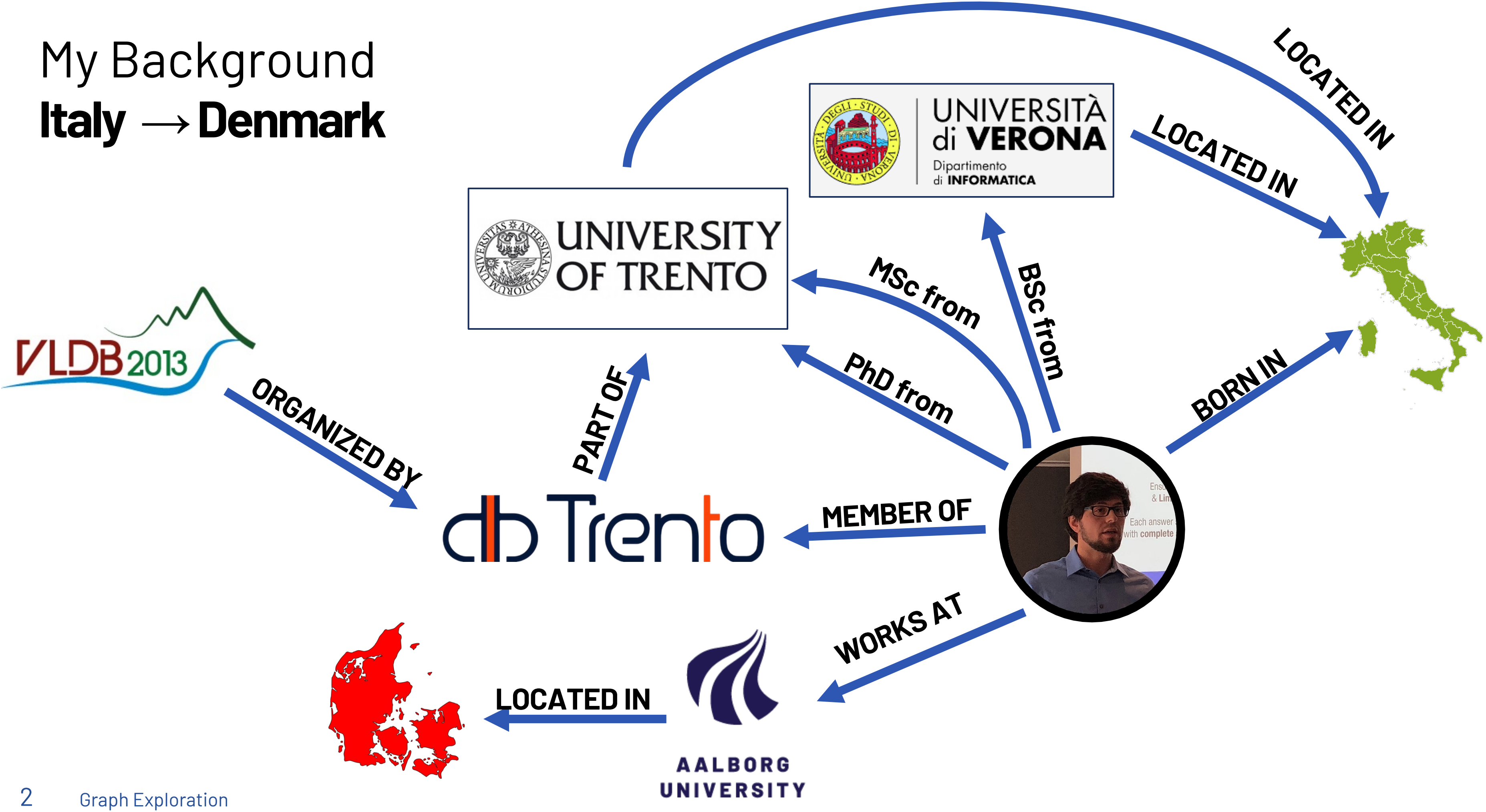
Hi! My name is **Matteo Lissandrini**

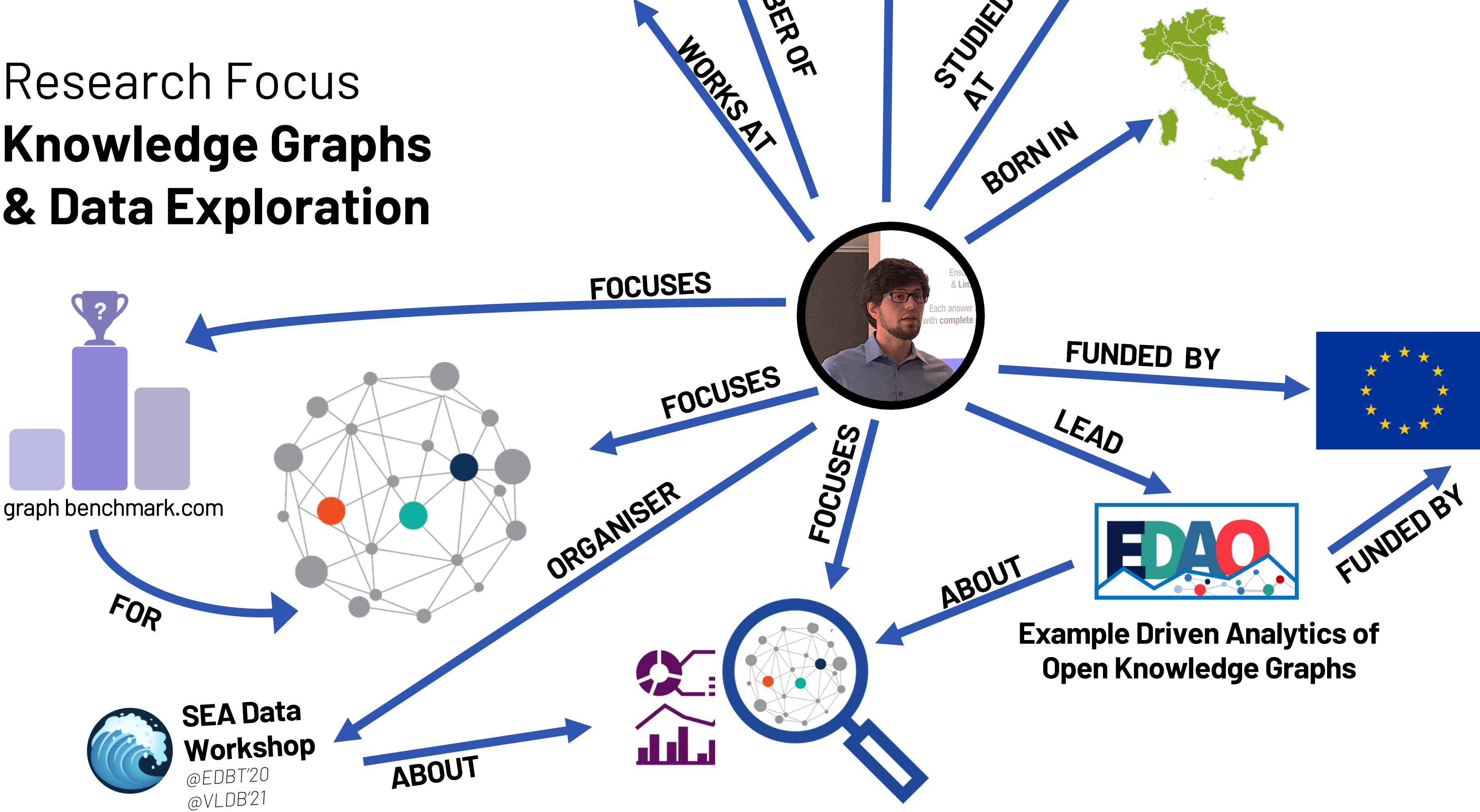
My Background

Italy → Denmark



Research Focus

Knowledge Graphs & Data Exploration

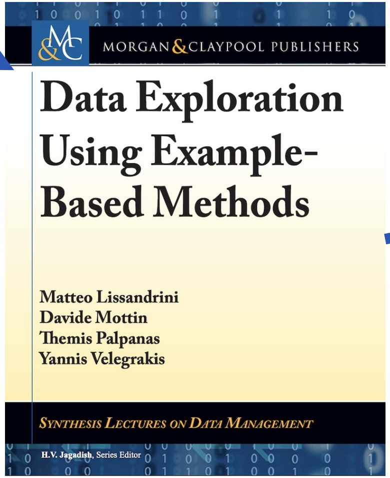


Research Direction
Example-Based Exploration

AUTHOR

AUTHOR

AUTHOR

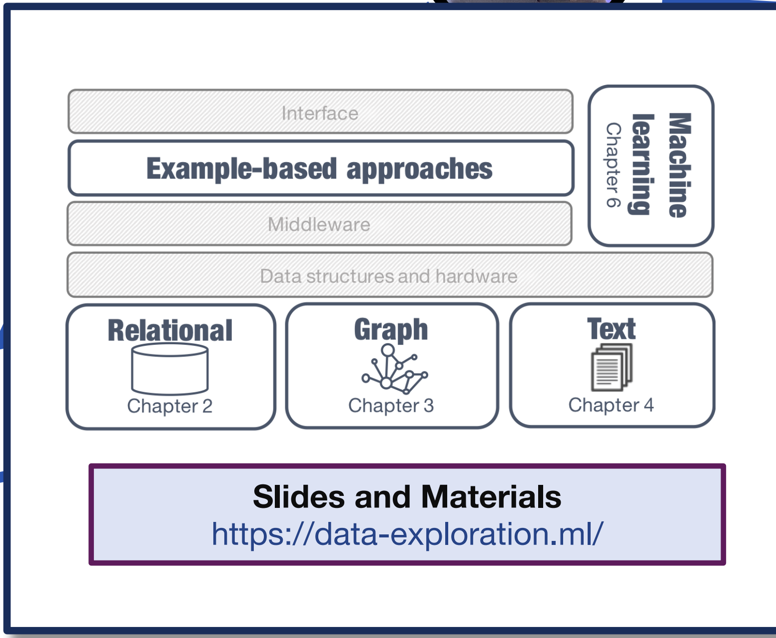

MORGAN & CLAYPOOL PUBLISHERS

Data Exploration Using Example-Based Methods

Matteo Lissandrini
Davide Mottin
Themis Palpanas
Yannis Velegrakis

SYNTHESIS LECTURES ON DATA MANAGEMENT

H.V. Jagadish, Series Editor

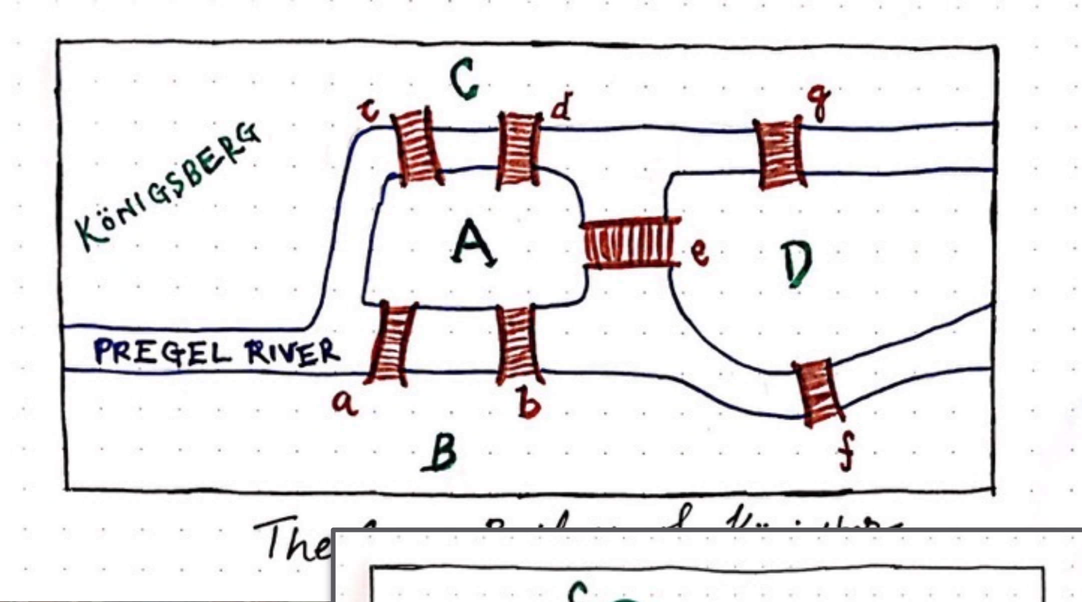


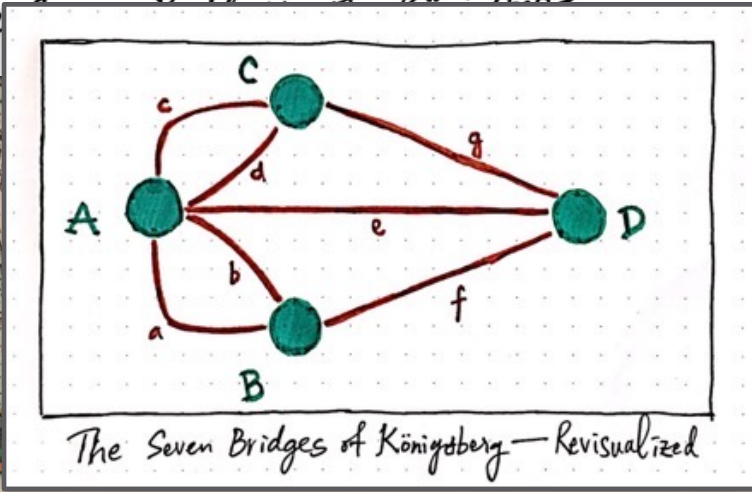
Graph Data Analysis & Exploration

— Modelling & Querying Graphs —

Matteo Lissandrini – Aalborg University

AALBORG UNIVERSITY





Welcome to **Königsberg**

– Leonhard Euler, 1735

https://medium.com/basecs/k%C3%B6nigsberg-seven-small-bridges-one-giant-graph-problem-2275d1670a12

Course Objectives:
at the end of the course

1. You understand the different ways in which the **graph model can be adopted** in different domains

2. You are familiar with **graph terminology** in relation to **challenges, methods, and solutions** for graph analysis

3. You can **identify core methods and challenges** to study the content and structure of a large graph

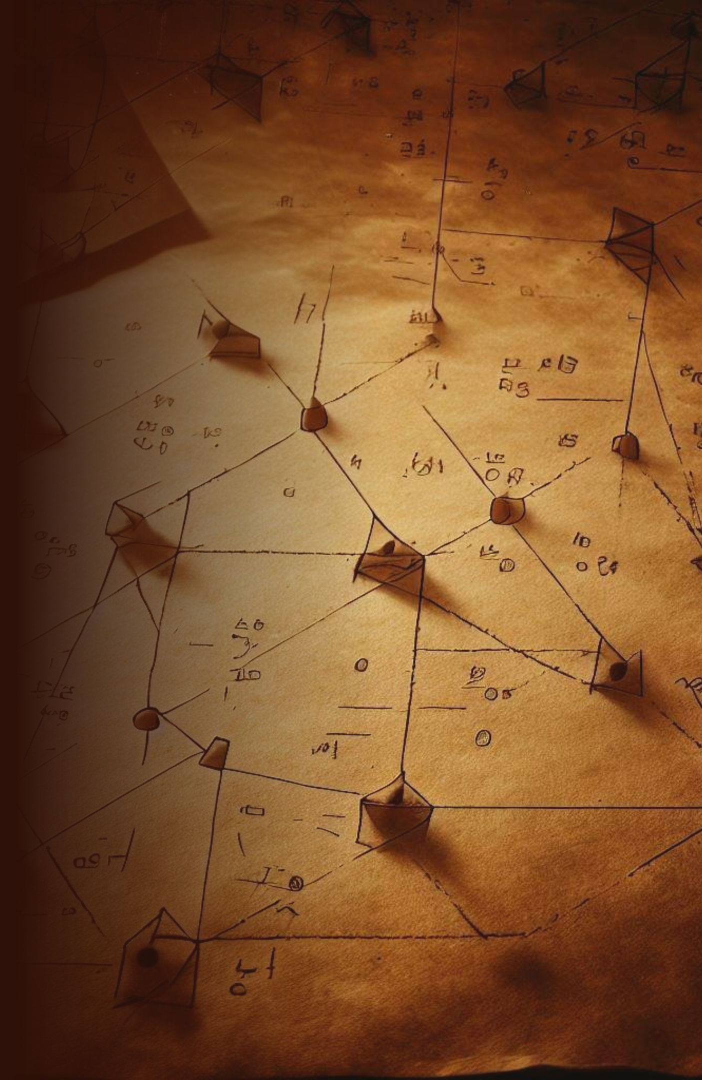
4. You have **concrete pointers and references** of advanced methods of graph analysis and exploration

Agenda

- Part 1: Core Concepts
  - Modelling & Querying Graphs
  - Network Analysis

- Part 2: Advanced Methods
  - Graph Structure Analysis
  - Graph Exploration

Extra Materials:
slides contain extra materials that we will not be able to cover today. Feel free to ask questions about those.



Optional Hands-On Exercises

github.com : AAU-WebDataScience/F23-PhD-GraphAnalysis

On References

Slides contain pointers to relevant materials

1. Many slides have been adapted from existing courses and presentations; they are referenced whenever possible

2. Some slides point to other online documentation, relevant Wikipedia pages (when sufficient), published papers, to expand when/if needed

Further Based on chapter & online slides:

from Mining of Massive Datasets; Leskovec, Rajaraman, Ullman (3rd edition)

from Web Data Mining; Bing Liu, Second Edition (July 2011)

More references also at the end of the slides

Outline

1. **Graphs are Everywhere**
   - The Web-Link structure
   - The Query-Log graph
   - The Social network
   - The Knowledge graph

2. **The Graph Model**
   - Undirected/Directed graphs
   - Labelled/Unlabelled graphs
   - N-partite graphs
   - RDF graphs
   - Property Graph
   - Graph Database vs. Database of Graphs

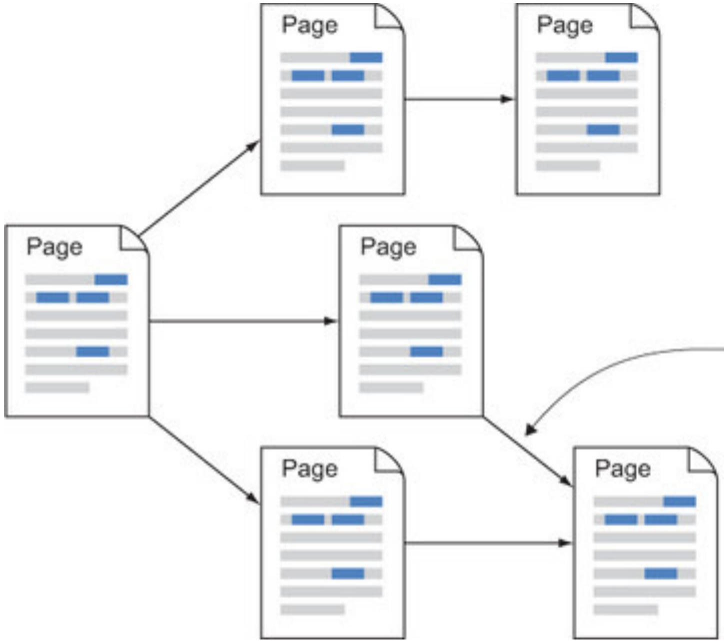
3. **Representing Graphs**
   - Adjacency matrix
   - Adjacency List
   - Triples & Storage for Triplestore
   - Property graph storage models

4. **Graph Navigation**
   - Breadth-First Search / Depth-First Search
   - Connected Components
   - Paths & Shortest path
   - CYPHER
   - SPARQL
   - Gremlin

Webpages and Links

WorldWideWeb is a hypertext browser/editor which allows one to read information from local files and remote servers. It allows **hypertext links to be made and traversed**, and also remote indexes to be interrogated for lists of useful documents. Local files may be edited, and **links made from areas of text to other files**, remote files, remote indexes, remote index searches, internet news groups and articles. All these sources of information are presented in a consistent way to the reader. For example, an index search returns a **hypertext document with pointers to documents** matching the query. Internet news articles are displayed with **hypertext links to other referenced articles** and groups.

– Tim Berners-Lee, 20 Aug 1991



Hyper-Links:
Text in pages links to other pages containing relevant information

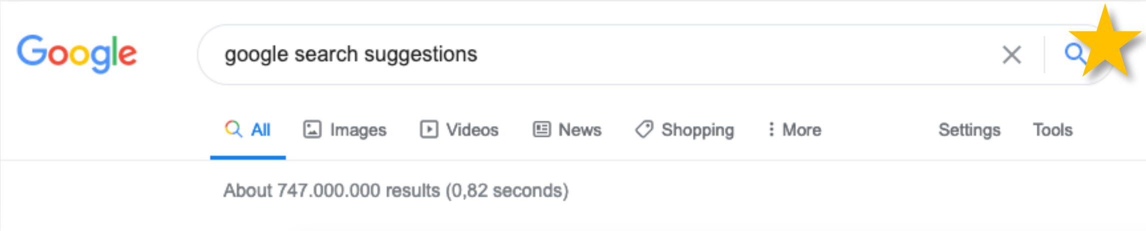
Traversing a Link:

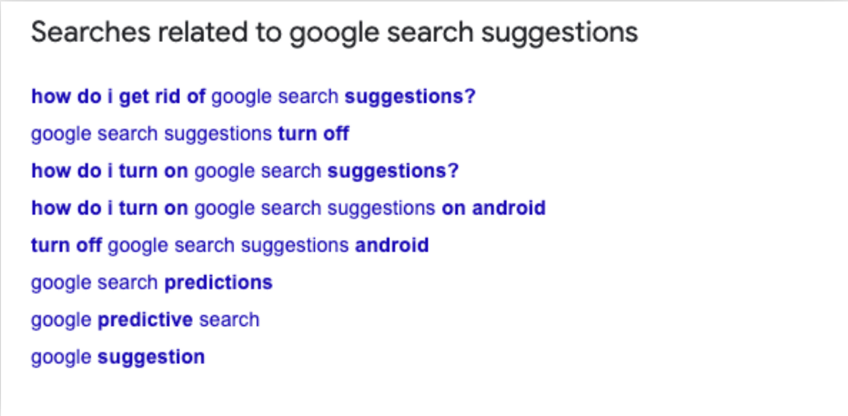
A link from Page A to Page B tells us that there is a relationship between the two documents.
Page A mentions something for which Page B contains additional relevant information

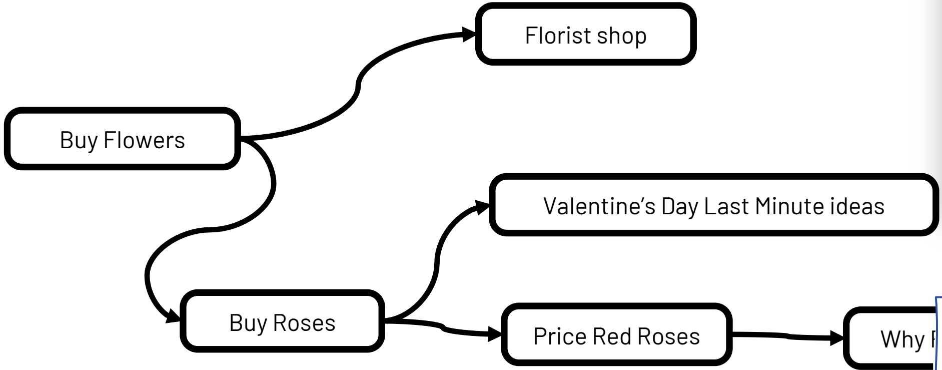
Query Logs

Monitoring Search Activites

1. Record Search query + User Click: Click Log
2. Record for the same user keyword search happening one after the other: Search Session







Session Links:
The user search for B
after searching for A

Implicit Links:

We infer a link from the behaviour of the user

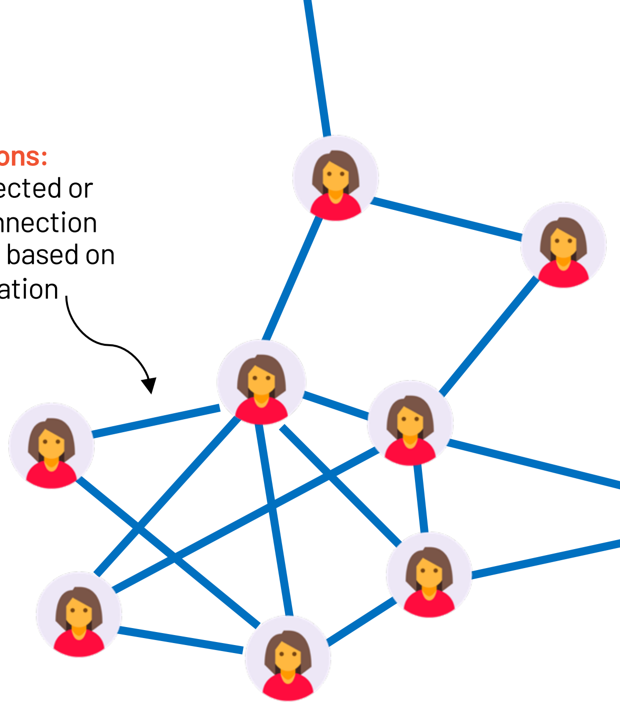
http://www.cim.mcgill.ca/~dudek/206/Logs/AOL-user-ct-collection/

Social Networks

User connections

- On a social media platform users can express explicitly their connection with other users
- Connection can be:
  1. Undirected: friendship, colleague
  2. Directed: Follow

User Connections:
Establish a directed or undirected connection between users based on explicit information



Objects and links are all of the same type:

Pages, Search queries, Users...

Product & Customer Networks

Heterogenous Networks:

Nodes are of different types

User-product & product-product connections

- Main nodes are customers & products, edges are transactions or interactions
- Other nodes can describe products

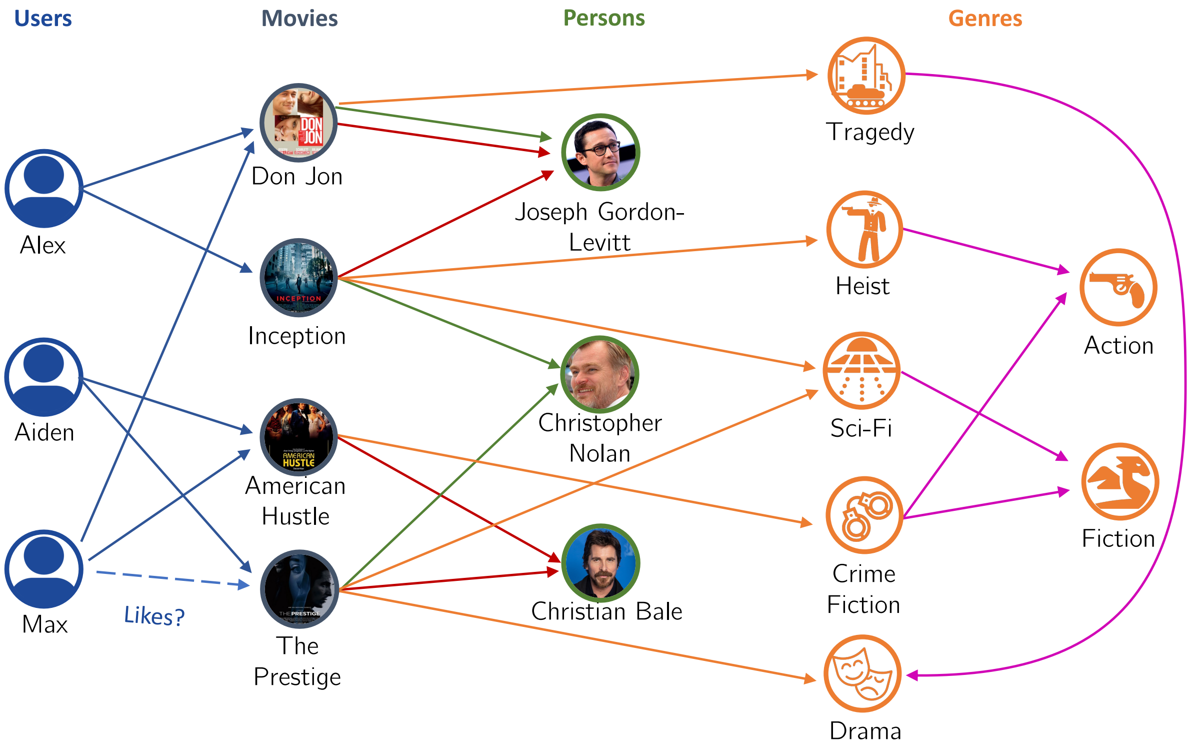
⟶ Likes

⟶ Starring

⟶ Directed by

⟶ Has genre

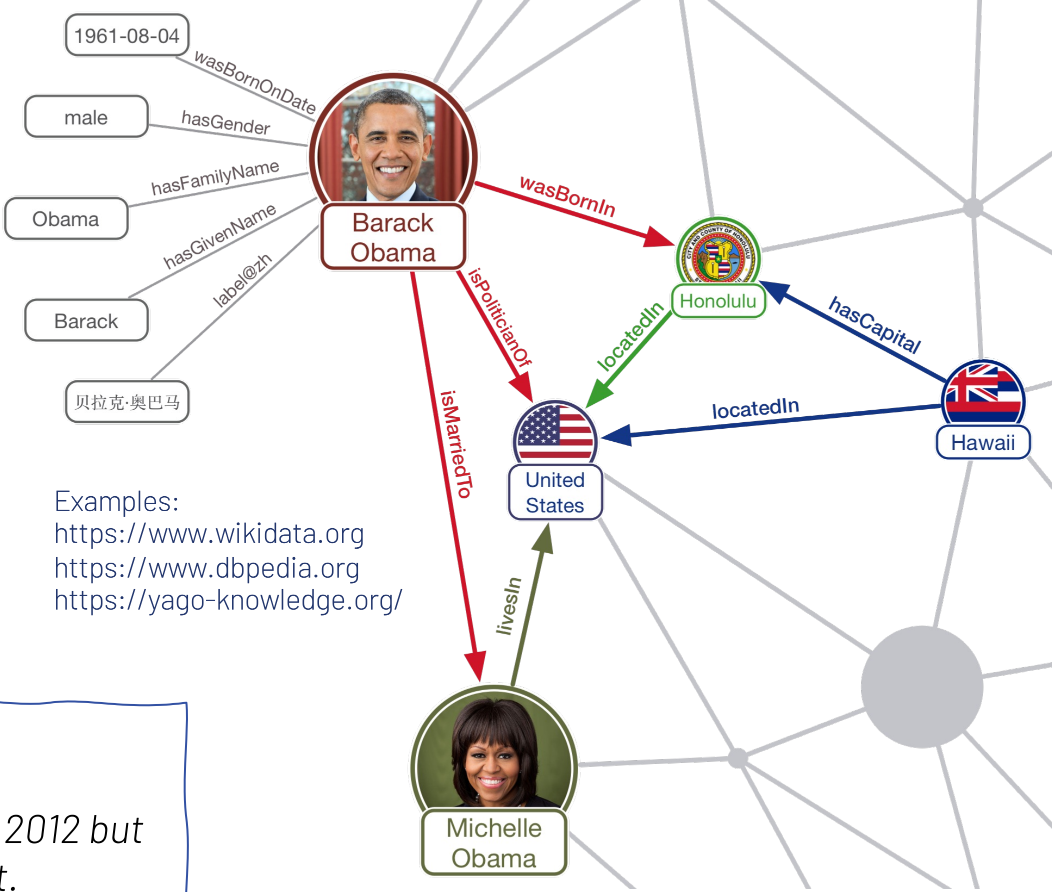
⟶ Subgenre of



“Concept Networks”

Abstract model of Knowledge

- Links represent «facts»
- Facts connect different objects
  1. Real Entities
  2. Abstract Concepts
  3. Pieces of Data
- Facts are of different type
  they have different meaning



Examples:
https://www.wikidata.org
https://www.dbpedia.org
https://yago-knowledge.org/

This model is called “Knowledge Graph”:

The term has been popularized by Google in 2012 but it existed in different forms earlier than that.
https://blog.google/products/search/introducing-knowledge-graph-things-not/

Knowledge Graph Adoption

Existing Open Knowledge Graphs

WIKIDATA
1.9B Facts

yago select knowledge
210M Facts

DBpedia
52M Facts

linked life data
http://linkedlifedata.com/sources.html

6.7B Facts

PubChem
https://pubchem.ncbi.nlm.nih.gov/docs/rdf

132B Facts

The Growing Role of Graphs & Knowledge Graphs

COMMUNICATIONS OF THE ACM

Home / Magazine Archive / August 2019 (Vol. 62, No. 8) / Industry-Scale Knowledge Graphs: Lessons and

PRACTICE

Industry-Scale Knowledge Graphs: Lessons and Challenges

By Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, Jamie Taylor

Communications of the ACM, August 2019, Vol. 62 No. 8, Pages 36-43
10.1145/3331166

Comments

Credit: Adempercem / Stutterstock

Knowledge graphs are critical to many enterprises today provide the structured data and factual knowledge that many products and make them more intelligent and "ma

In general, a knowledge graph describes objects of intere connections between them. For example, a knowledge g have nodes for a movie, the actors in this movie, the dire so on. Each node may have properties such as an actor's and age. There may be nodes for multiple movies involv particular actor. The user can then traverse the knowled to collect information on all the movies in which the act appeared or, if applicable, directed.

Many practical implementations impose constraints on in knowledge graphs by defining a *schema* or *ontology*. For example, a link from a movie to its direct connect an object of type Movie to an object of type Person. In some cases the links themselves might have their own properties: a link connecting an actor and a movie might have the name of the specific role the actor played. Similarly, a link connecting a politician with a specific role in government might have the time period

COMMUNICATIONS OF THE ACM

Home / Magazine Archive / September 2021 (Vol. 64, No. 9) / The Future Is Big Graphs: A Community View on Graph... /

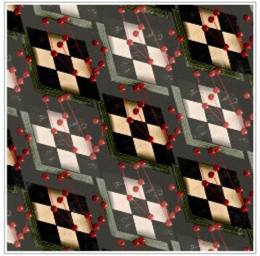
CONTRIBUTED ARTICLES

The Future Is Big Graphs: A Community View on Graph Processing Systems

By Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Ka Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei R. Ripeanu, Semih Juan F. Sequeda, Joshua Shinavier

Communications of the ACM, September 2021, Vol. 64 No. 9, Pages 62-71
10.1145/3434642

Comments



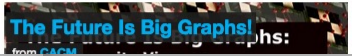
Credit: Alli Torban

Graphs are, by nature, 'unifyi interconnectedness to repres real- and digital-world pheno consumers of graph instances these abstractions, future pro and systems. What needs to h graph processing to continue

Back to Top

Key Insights

- Graphs are enabling r graph proc every dom
- Diverse w languages suitable a metrics w processing decade.



COMMUNICATIONS OF THE ACM

Home / Magazine Archive / March 2021 (Vol. 64, No. 3) / Knowledge Graphs / Full Text

REVIEW ARTICLES

Knowledge Graphs

By Claudio Gutierrez, Juan F. Sequeda

Communications of the ACM, March 2021, Vol. 64 No. 3, Pages 96-104
10.1145/3418294

Comments

"Those who cannot remember the past are condemned to repeat it."
—George Santayana

Back to Top

Key Insights

- Data was traditionally considered a material object, tied to bits, with no semantics per se. Knowledge was traditionally conceived as the immaterial object, living only in people's minds and language. The destinies of data and knowledge became bound together, becoming almost inseparable, by the emergence of digital computing in

Outline

1. **Graphs are Everywhere**
   - The Web-Link structure
   - The Query-Log graph
   - The Social network
   - The Knowledge graph

2. **The Graph Model**
   - Undirected/Directed graphs
   - Labelled/Unlabelled graphs
   - N-partite graphs
   - RDF graphs
   - Property Graph
   - Graph Database vs. Database of Graphs

3. **Representing Graphs**
   - Adjacency matrix
   - Adjacency List
   - Triples & Storage for Triplestore
   - Property graph storage models

4. **Graph Navigation**
   - Breadth-First Search / Depth-First Search
   - Connected Components
   - Paths & Shortest path
   - CYPHER
   - SPARQL
   - Gremlin

Outline

1. **Graphs are Everywhere**
   - The Web-Link structure
   - The Query-Log graph
   - The Social network
   - The Knowledge graph

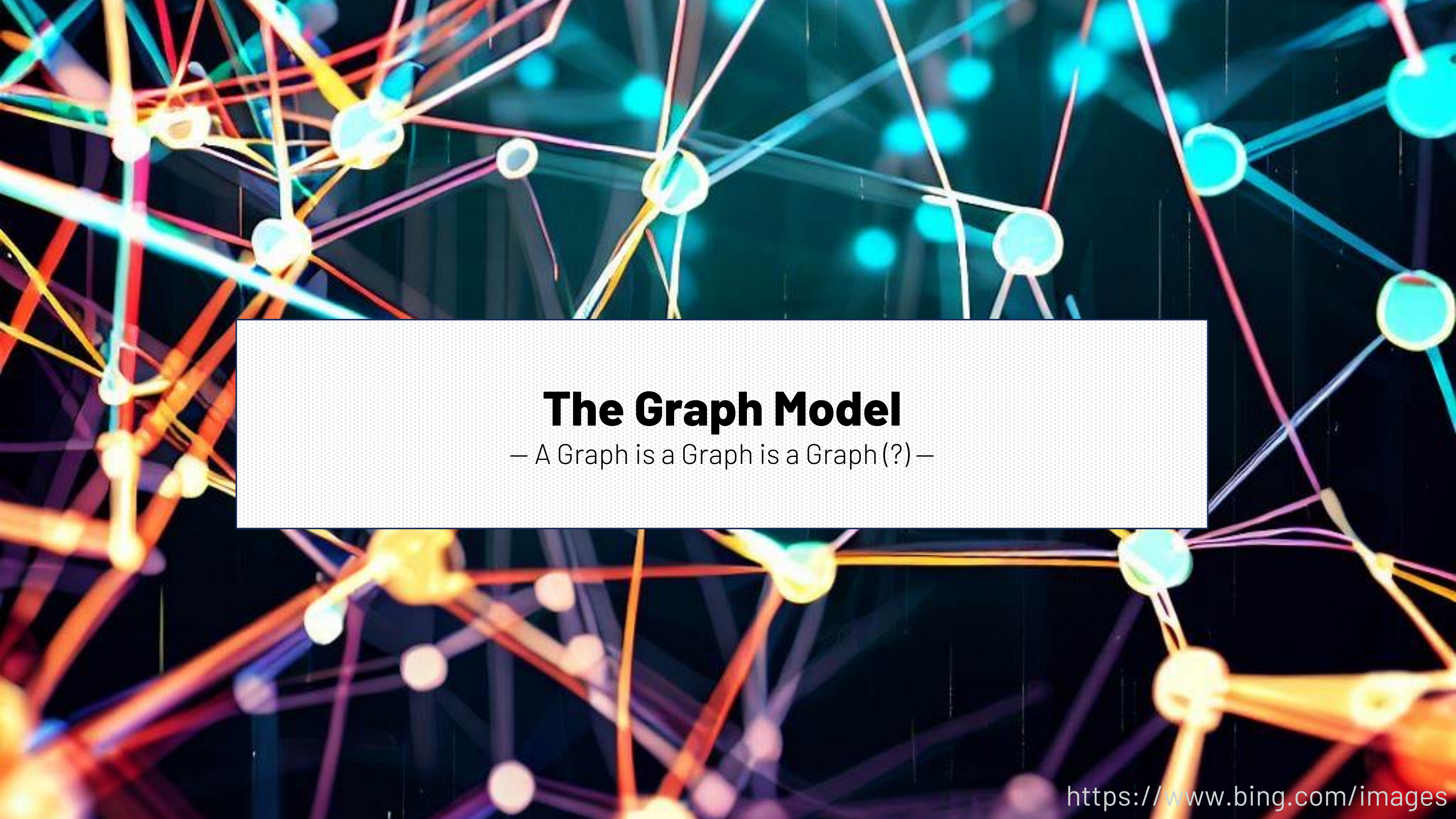
2. **The Graph Model**
   - Undirected/Directed graphs
   - Labelled/Unlabelled graphs
   - N-partite graphs
   - RDF graphs
   - Property Graph
   - Graph Database vs. Database of Graphs

3. **Representing Graphs**
   - Adjacency matrix
   - Adjacency List
   - Triples & Storage for Triplestore
   - Property graph storage models

4. **Graph Navigation**
   - Breadth-First Search / Depth-First Search
   - Connected Components
   - Paths & Shortest path
   - CYPHER
   - SPARQL
   - Gremlin

The Graph Model

— A Graph is a Graph is a Graph (?) —



https://www.bing.com/images

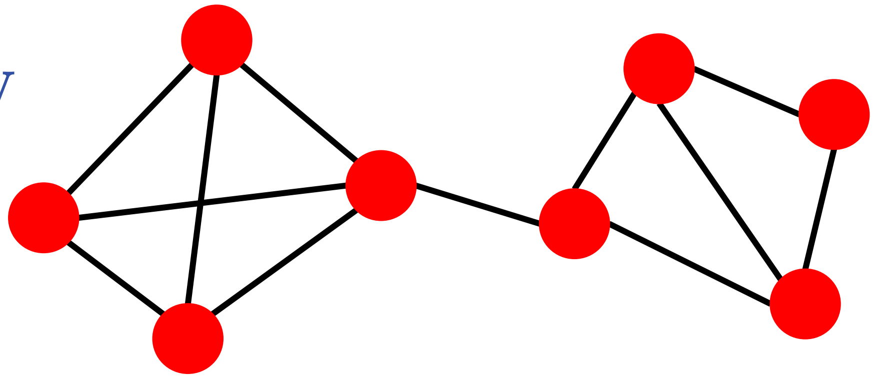
The Graph Model: Formalized

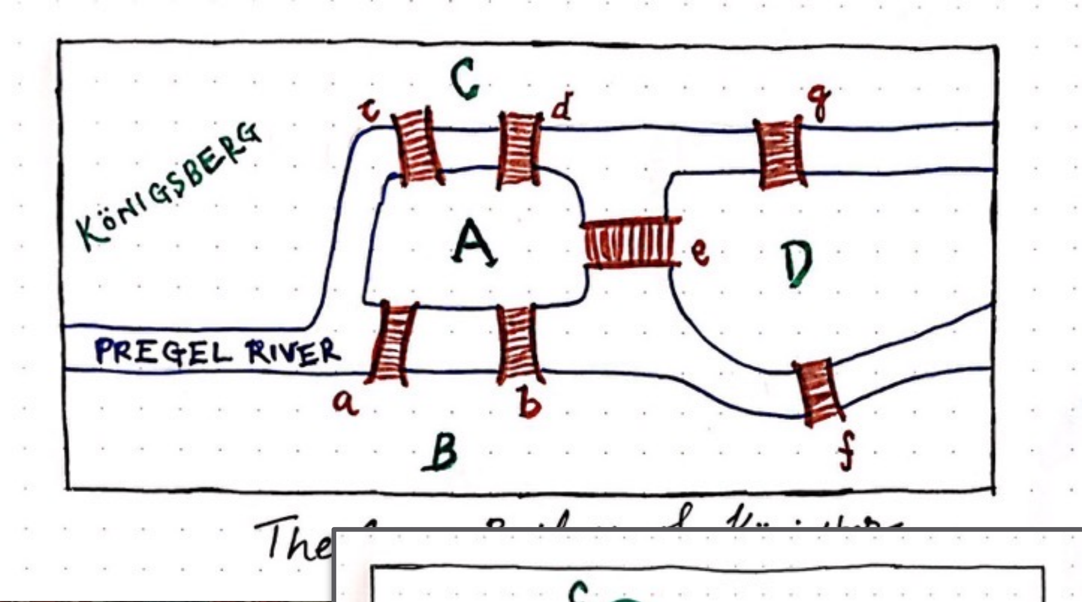
These are "simplistic" formalization, we will see better versions

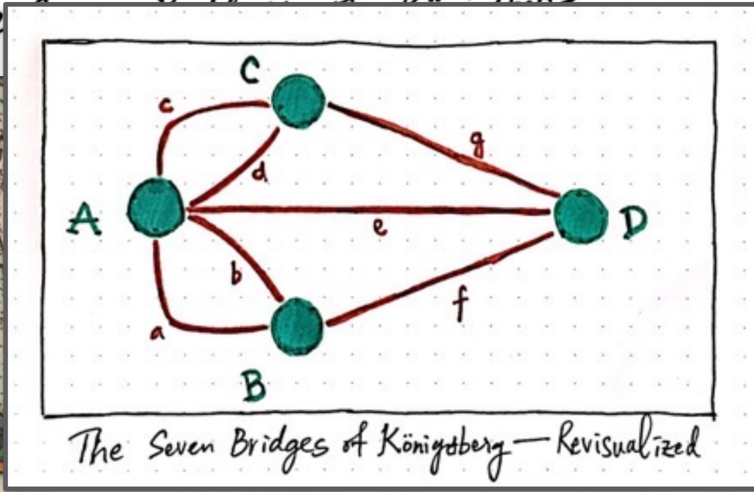
$\mathrm{G}\colon \langle\ \mathrm{Nodes}\ ;\ \mathrm{Edges}\ \rangle$

- **Nodes** N : identified by some ID
- **Edges** E : $\mathrm{E} \subseteq \mathrm{N} \times \mathrm{N}$ $\rightarrow$ identified by pair of nodes

Also called Vertices $\mathbf{V}$







Welcome to **Königsberg**

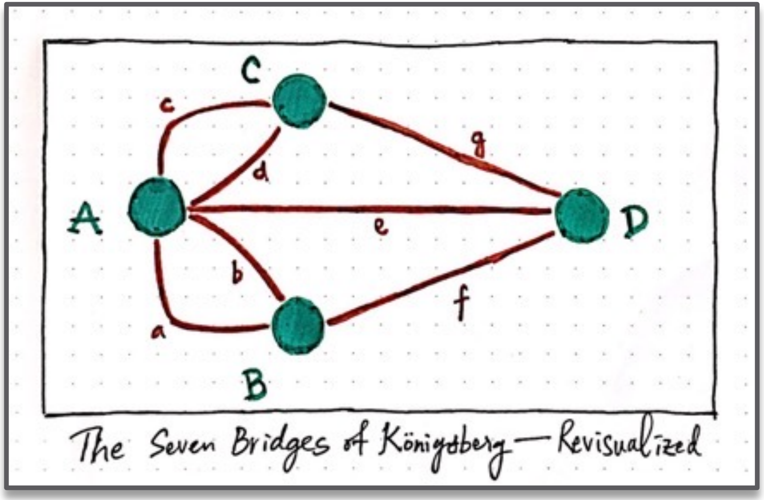
– Leonhard Euler, 1735

https://medium.com/basecs/k%C3%B6nigsberg-seven-small-bridges-one-giant-graph-problem-2275d1670a12

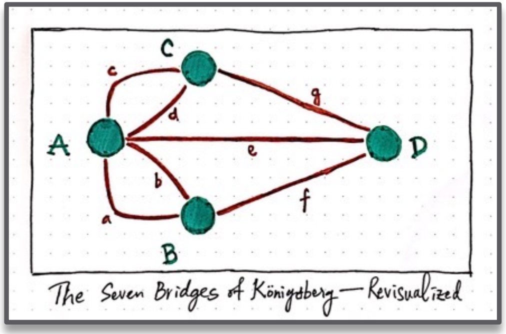
The Graph Model: Formalized

$G: \langle\ N\ ;\ E\ \rangle$

- **Nodes** N : identified by some ID
- **Edges** E **:** $E \subseteq N \times N$ $\rightarrow$ identified by pair of nodes



The Graph Model: Formalized



$$G\colon \langle\ N\ ;\ E\ \rangle$$

- **Nodes** N : identified by some ID
- **Edges** E **:** $E \subseteq N \times N$ $\rightarrow$ identified by pair of nodes

Options: 1) Undirected $e_{ij}\colon \langle ID_i, ID_j\rangle \equiv e_{ji}\colon \langle ID_j, ID_i\rangle$

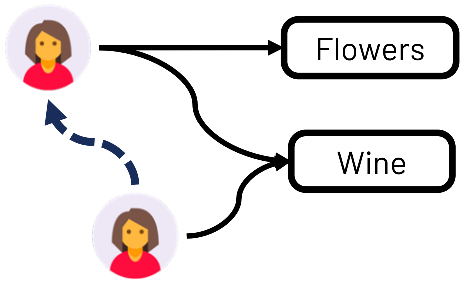
2) Directed $e_{ij}\colon \langle ID_i, ID_j\ \rangle \neq e_{ji}\colon \langle ID_j, ID_i\rangle$

Source Destination

Multigraph: If E can contain duplicates

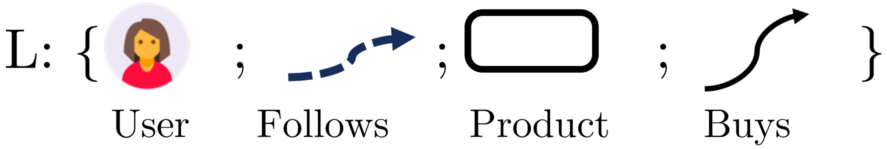
$E : E \subseteq \mathbb{N} \times N \times N$ $\rightarrow$ we assign IDs to edges

The Graph Model: Formalized



$G: \langle\, N \,;\, E \,;\, L \,;\, f_{(L)} \,\rangle$

- **Nodes** N : identified by some ID
- **Edges** E : $E \subseteq N \times N$ $\rightarrow$ identified by pair of nodes
- **Labels** L : special values that describe the type of a node or edge
- **Labeling Function** $f_{(L)}$: $N \cup E \rightarrow L$



The Graph Model: N-partite graphs

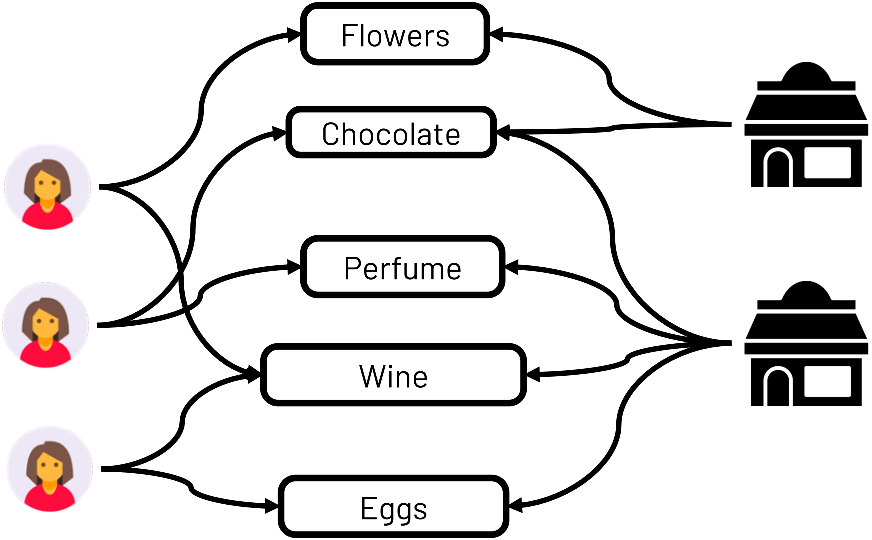
Assume the following Graph

- **Nodes** N : *Users + Products + Stores*
- **Edges** E : *User Buys Product*
  + Store sells product

Bi-partite graph
Tri-partite graph

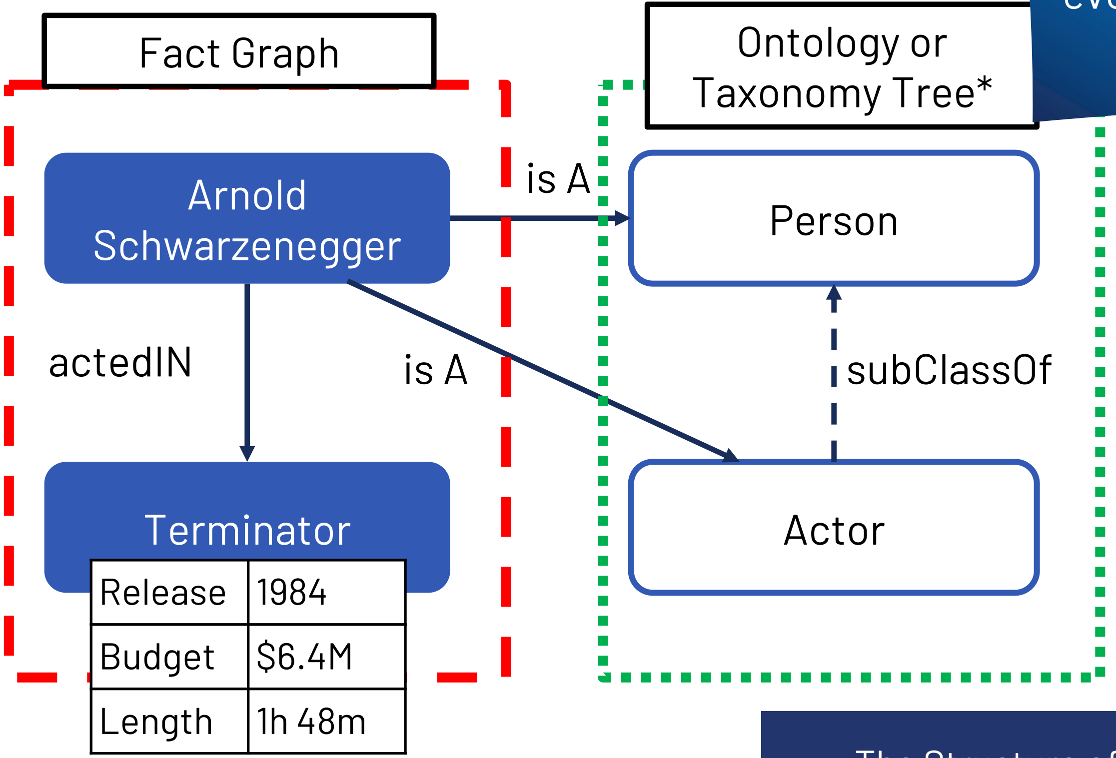
N-partite graphs:

(a) Nodes are divided in subsets.
(b) Connections exists only from one subset to another, and never within the same subset



Knowledge Graphs

The Structure evolves & adapts to the Data



The Structure of the Graph
Is as important as the Data values

RDF & Triples

Representing a KG as a Collection of Facts

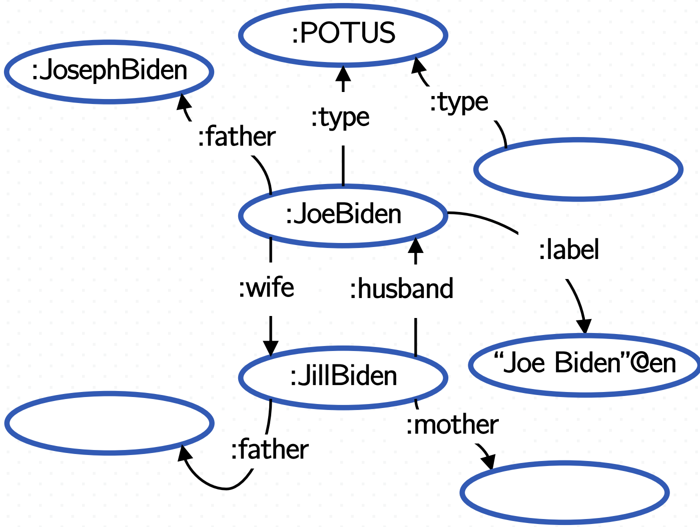
the RDF Model

- **Nodes are either:**
  - Entities (resources identified by IRI)
  - Literals (values as strings, integers, dates)
  - Blank Nodes (special kind of nodes without IRI)

- **Edges are statements**
  (<u>S</u>ubject, <u>P</u>redicate, <u>O</u>bject)

- Edge types (predicates) are resources

That's a triple!




```
@prefix : <http://www.example.kg/> .
:JoeBiden :label "Joe Biden"@en .
:JoeBiden :type  :POTUS .
:JoeBiden :wife :JillBiden .
:JillBiden :husband :JoeBiden .
```

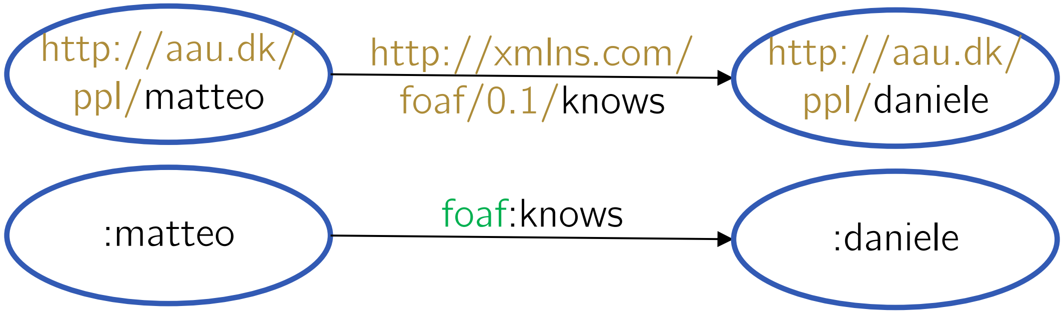
Statements – example

- We want to express the fact that “Matteo knows Daniele”
- “Matteo”, “knows”, and “Daniele” are resources and should be identified by URIs

  Matteo - http://aau.dk/ppl/matteo

  Daniele - http://aau.dk/ppl/daniele

  knows - http://xmlns.com/foaf/0.1/knows



foaf: -> http://xmlns.com/foaf/0.1/

: -> http://aau.dk/ppl/

Thanks Daniele Dell’Aglio, AAU

RDF Graph Formal Definition

$\mathcal{I}$: Internationalized Resource Identifiers (IRIs),
$\mathcal{L}$: typed or un-typed literals (constants),
$\mathcal{B}$: blank nodes (placeholders for IRIs or literals).

```
@prefix : <http://www.example.kg/> .
:JoeBiden :label "Joe Biden"@en .
:JoeBiden :type :POTUS .
:JoeBiden :wife :JillBiden .
:JillBiden :husband :JoeBiden .
```

An RDF graph is a labeled directed graph $G = \langle \mathcal{N}, \mathcal{E} \rangle$ with:

- $\mathcal{N} \subseteq \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ is the set of nodes
  $\mathcal{N}^{>0} = \mathcal{N} \setminus \mathcal{L}$ nodes in $\mathcal{N}$ allowed to have outgoing edges
  (*literals are never subjects!*)
- $\mathcal{E} \subseteq \mathcal{N}^{>0} \times \mathcal{I} \times \mathcal{N}$ is the set of directed edges;
- $\mathcal{P}: \{p \in \mathcal{I} \mid \exists (s, p, o) \in \mathcal{E}\}$ is the set of predicates for G.

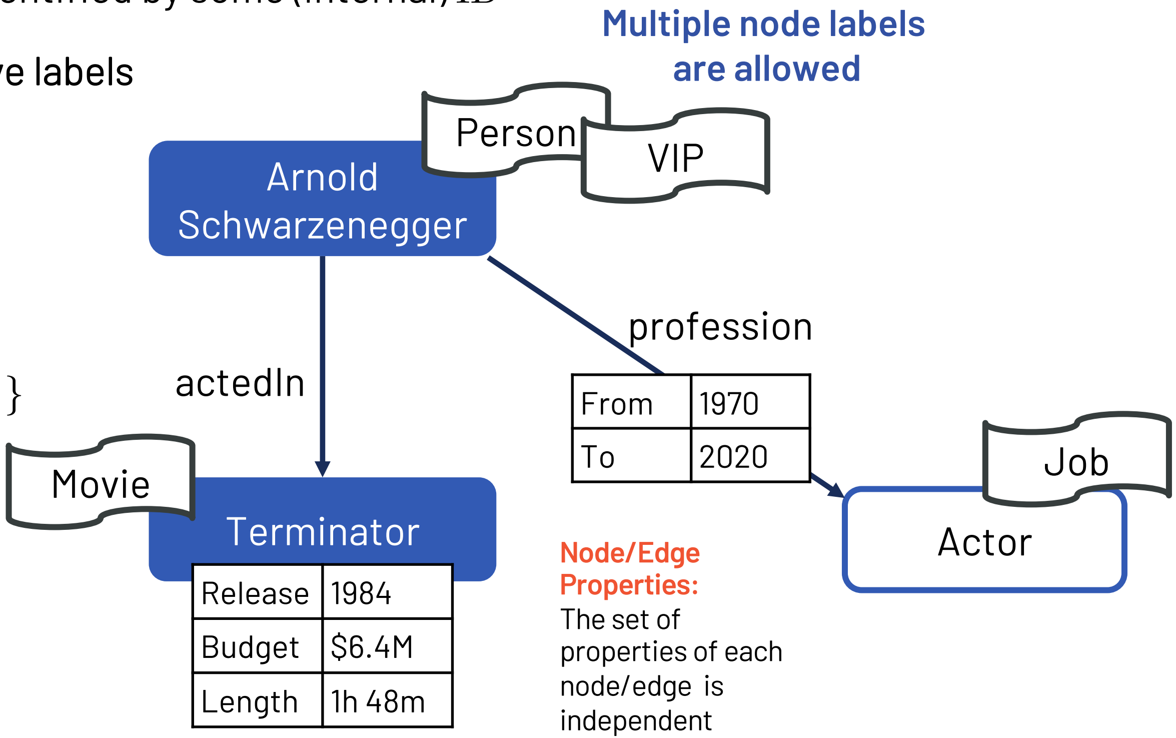
It is not properly a multigraph!

W3C RDF Working Group. 2014. Resource description framework. http://www.w3.org/RDF/.

The Graph Model Extended: Property Graph

- **Both Nodes** N **& Edges** E : identified by some (internal) ID
- **Both Nodes** N **& Edges** E have labels
- Assign to each node & edge

  a dictionary of attributes
  - ID $\rightarrow \{ <\mathrm{key}_1,\mathrm{value}_1>, \ldots \}$



Formalization of Property Graph

Countable sets $\mathcal{L}$: Labels $\mathcal{K}$: Keys (property names) and $\mathcal{V}$: property values.

A record is a partial function $o: \mathcal{K} \to \mathcal{V}$ mapping keys to values.

($\mathcal{R}$ for the set of all records)

Property Graph: $G = (N, E, \rho, \lambda, \pi)$ where:

It is properly a multigraph!

- N is a finite set of nodes (identified by an ID);
- E is a finite set of edges (identified by an ID) such that $N \cap E = \emptyset$;
- $\rho: E \to (N \times N)$ maps edges to pairs of nodes
- $\lambda: (N \cup E) \to 2^{\mathcal{L}}$ labelling function maps nodes and edges to finite sets of labels (including the empty set)
- $\pi: (N \cup E) \to \mathcal{R}$ property mapping is a function mapping nodes and edges to records.

A Standard for Property Graphs?

```
CREATE GRAPH TYPE  fraudGraphType STRICT {
  (personType: Person {name STRING}),
  (customerType: personType & Customer {id INT32}),
  (creditCardType: CreditCard {num STRING}),
  (transactionType: Transaction {num STRING}),
  (accountType: Account {id INT32}),
  (:customerType)
    -[ownsType: owns]->
  (:accountType),
  (:customerType)
    -[usesType: uses]->
  (:creditCardType),
  (:transactionType)
    -[chargesType: charges {amount DOUBLE}]->
  (:creditCardType),
  (:transactionType)
    -[activityType: deposits|withdraws]->
  (:accountType)
}
```

Fig. 2. PG-Schema of a fraud graph schema.

PG-Schema: Schemas for Property Graphs

RENZO ANGLES, Faculty of Engineering, Universidad de Talca, Chile
IIFATI, Lyon 1 University & Liris CNRS, France
MBRAVA, ENSIIE & SAMOVAR - Institut Polytechnique de Paris, France
TCHER, Eindhoven University of Technology, Netherlands
EEN, LDBC, UK
, Birkbeck, University of London, UK
JSA
N, University of Edinburgh, UK and RelationalAI & ENS, PSL University, France
AULT, LIGM, Université Gustave Eiffel, CNRS, France
S, University of Bayreuth, Germany
, University of Warsaw, Poland
TIKOW, Neo4j, Germany
OVIĆ, Free University of Bozen-Bolzano, Italy
MIDT, Amazon Web Services, USA
A, data.world, USA
ORKO, RelationalAI, USA and Univ. Lille, CNRS, UMR 9189 CRIStAL, France
ASZUK, University of Bialystok, Poland
T, Neo4j, Germany
GOČ, University of Zagreb, Croatia and PUC Chile, Chile
gerGraph, USA
VIĆ, Integral Data Solutions, UK

e reached a high level of maturity, witnessed by multiple robust graph datab

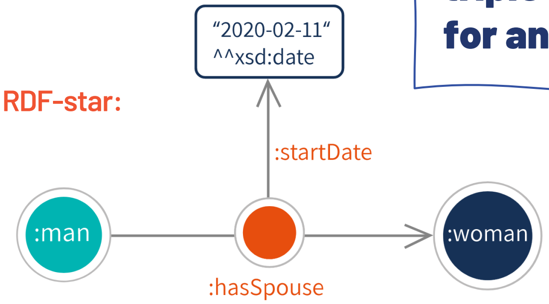
Edge Property in RDF: RDF-Star

Classical RDF:
Only nodes can be subjects of triples

How can we represent edge property in RDF?

Use an entire triple as subject for another triple

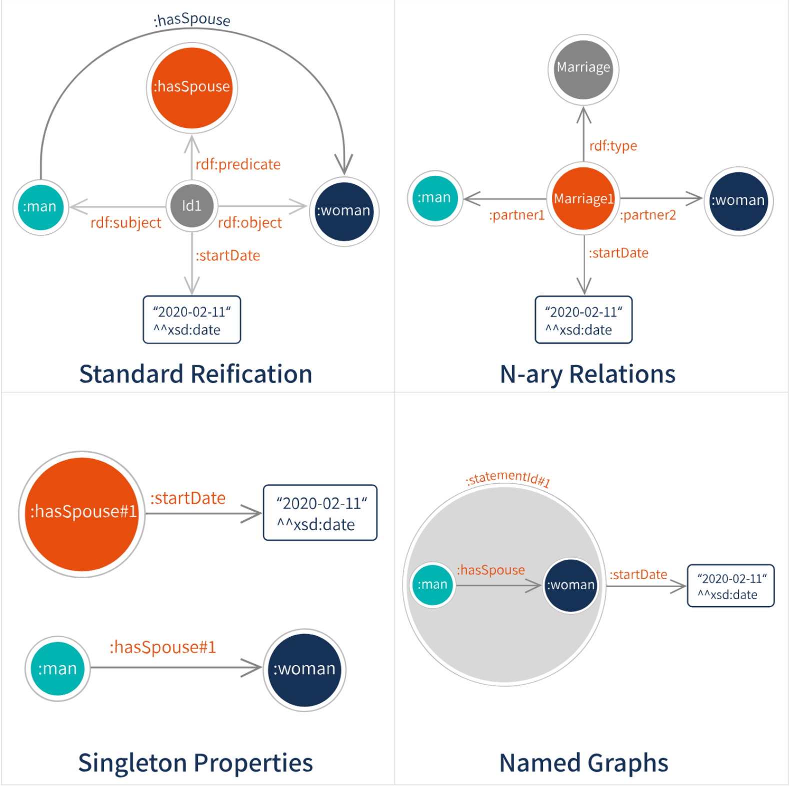
RDF-star:



(S,   P,   O)
:man :**hasSpouse** :woman .

<<:man :hasSpouse :woman>> **:startDate** "2020-02-11"^^xsd:date .

(S,   P,   O)



https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-star/
https://w3c.github.io/rdf-star/cg-spec/editors_draft.html
https://www.bobdc.com/blog/rdf-and-sparql/

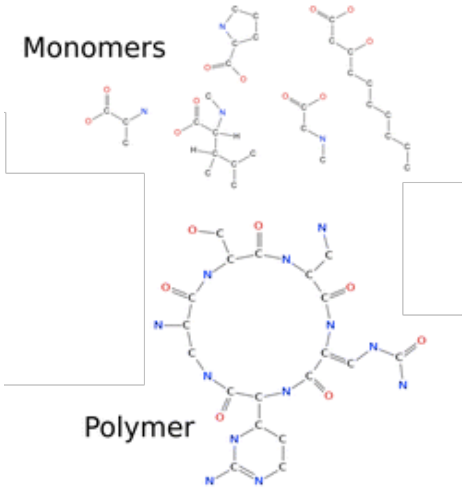
Types of Graph Databases

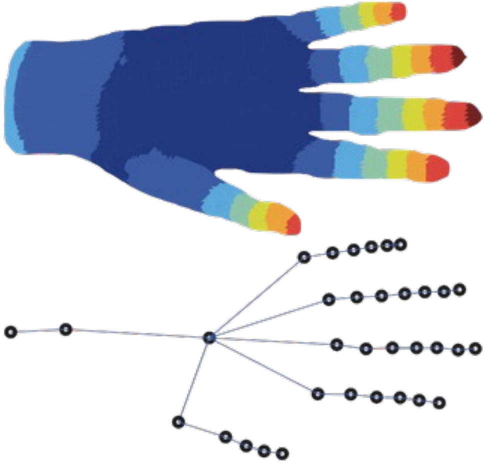
Single large graphs

- The web
- Social network
- Knowledge Graph

Distinct Graphs (a.k.a. database of graphs)

- Protein-Protein interactions
- Molecules
- 3D Objects





Graph-Databases / Databases of Graphs

There is often confusion in how this terminology is used. It depends on the context, pay attention!

Databases vs. DBMS

A collection of related pieces of data vs. Database Management Systems (software)

Outline

1. **Graphs are Everywhere**
   - The Web-Link structure
   - The Query-Log graph
   - The Social network
   - The Knowledge graph

2. **The Graph Model**
   - Undirected/Directed graphs
   - Labelled/Unlabelled graphs
   - N-partite graphs
   - RDF graphs
   - Property Graph
   - Graph Database vs. Database of Graphs

3. **Representing Graphs**
   - Adjacency matrix
   - Adjacency List
   - Triples & Storage for Triplestore
   - Property graph storage models

4. **Graph Navigation**
   - Breadth-First Search / Depth-First Search
   - Connected Components
   - Paths & Shortest path
   - CYPHER
   - SPARQL
   - Gremlin

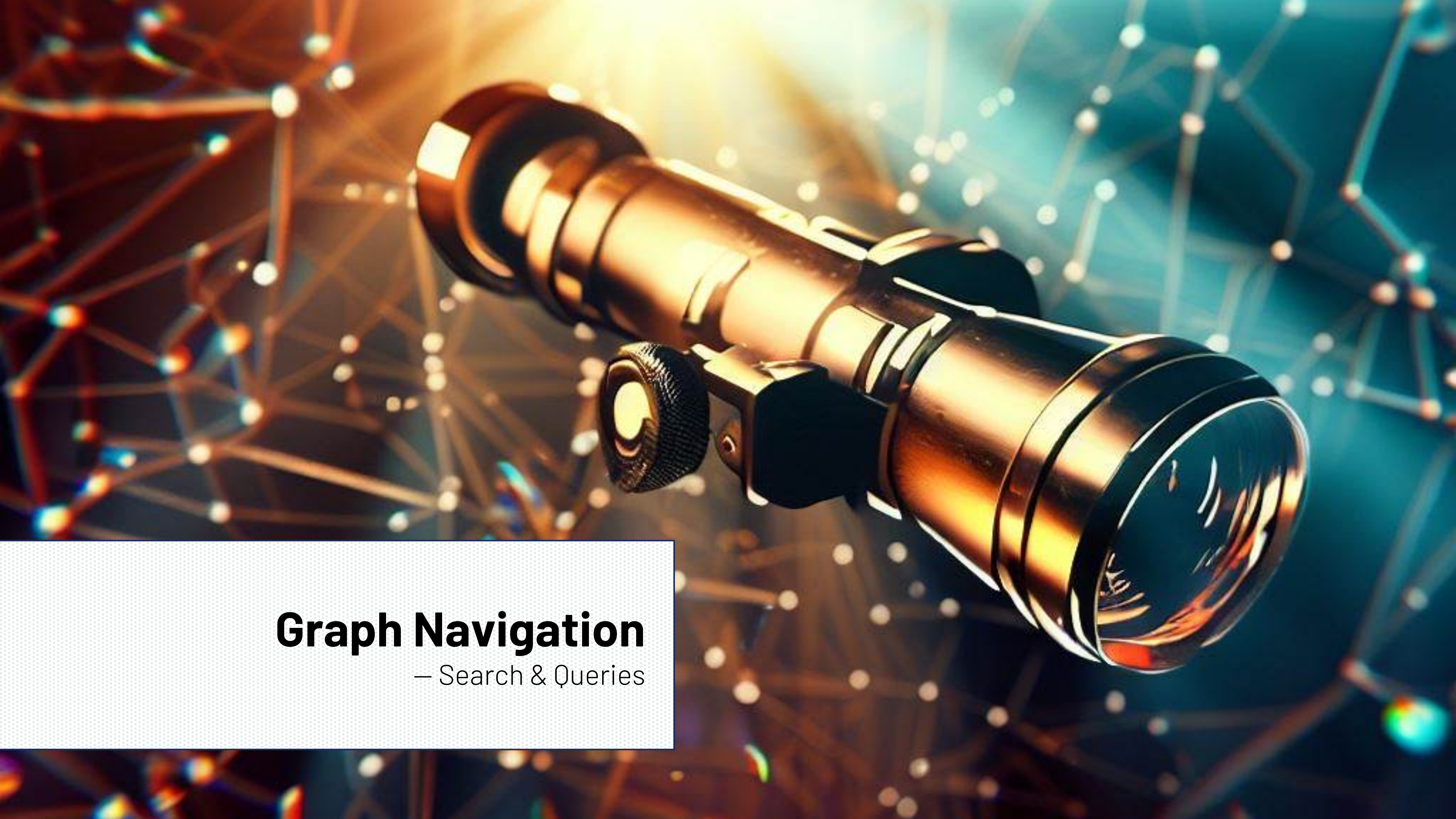
Outline

1. **Graphs are Everywhere**
   - The Web-Link structure
   - The Query-Log graph
   - The Social network
   - The Knowledge graph

2. **The Graph Model**
   - Undirected/Directed graphs
   - Labelled/Unlabelled graphs
   - N-partite graphs
   - RDF graphs
   - Property Graph
   - Graph Database vs. Database of Graphs

3. **Representing Graphs**
   - Adjacency matrix
   - Adjacency List
   - Triples & Storage for Triplestore
   - Property graph storage models

4. **Graph Navigation**
   - Breadth-First Search / Depth-First Search
   - Connected Components
   - Paths & Shortest path
   - CYPHER
   - SPARQL
   - Gremlin



Graph Navigation

— Search & Queries

Graph Navigation

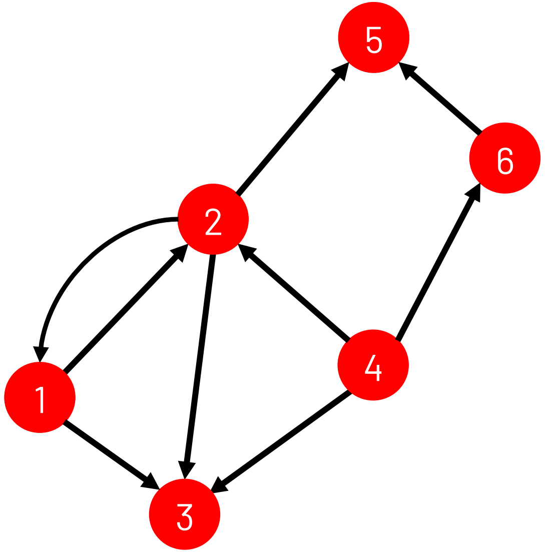
What operations we do on a graph? Given a node obtain:

- **Neighbors:** obtain the list of all nodes connected to it
- **Degree:** number of nodes connected (when undirected)
  - When directed: In-degree / out-degree
- **Graph Traversal**
  - Start from a node, obtain the list of all reachable nodes

$$\text{Neighbors}(1)\text{: } \{2, 3\}$$

$$\text{InDegree}(2) = 2 \quad \text{OutDegree}(2) = 3$$

$$\text{Reachable}(4) = \{2, 3, 1, 6, 5\} \quad \text{Reachable}(1) = \{2, 3, 5\}$$



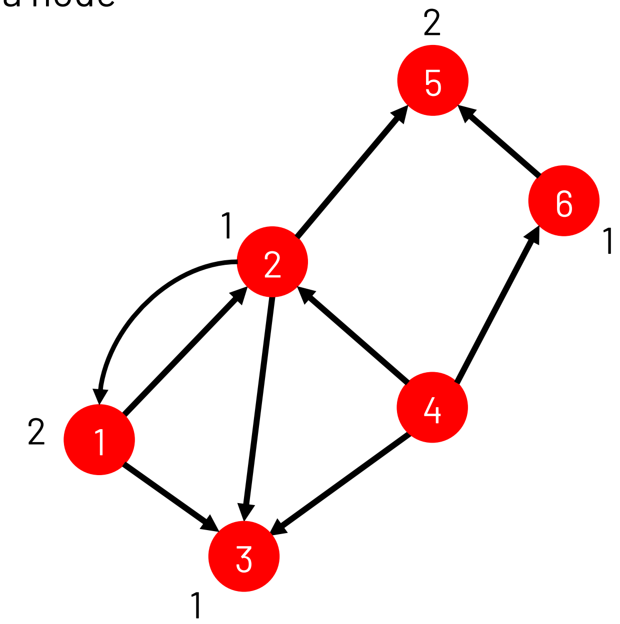
Graph Traversal: BFS vs. DFS

- **Graph Traversal:** start from a node, obtain the list of all reachable nodes, in which order?

- **Breadth-First Search (BFS):** visit first all the neighbors of a node before visiting the other

$$\mathrm{BFS}(4) = [2,\, 3,\, 6,\, 1,\, 5]$$

```
Q = queue()
Q.enqueue(startNode)
mark startNode as visited
while Q is not empty do
    v := Q.dequeue()
    // do something with v here //
    for all w in neighbors(v) do
       if w is not visited then
             mark w as visited
         Q.enqueue(w)
```



Graph Traversal: BFS vs. DFS

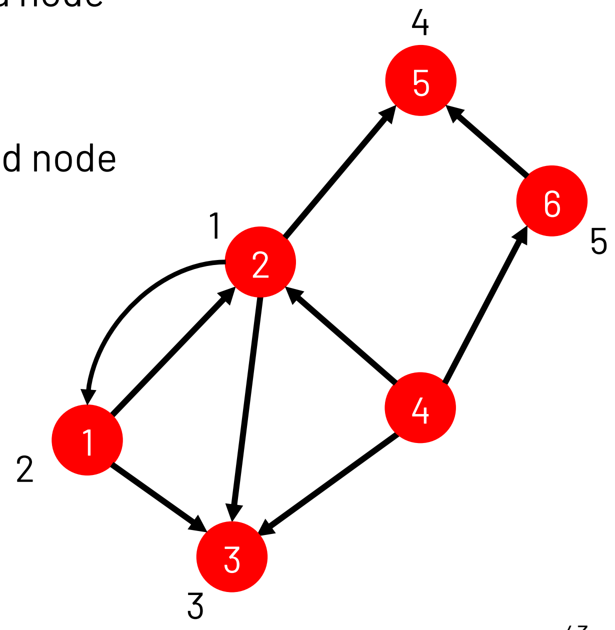
Graph Traversal from a single node is used to find all reachable nodes

- **Graph Traversal:** start from a node, obtain the list of all reachable nodes, in which order?

- **Breadth-First Search (BFS):** visit first all the neighbors of a node before visiting the other
  $$\mathrm{BFS}(4) = [2,\ 3,\ 6,\ 1,\ 5]$$
- **Depth-First Search (DFS):** visit a neighbor of the last visited node
  $$\mathrm{DFS}(4) = [2,\ 1,\ 3,\ 5,\ 6]$$

```
Q = queue()
Q.enqueue(startNode)
mark startNode as visited
while Q is not empty do
    v := Q.dequeue()
    // do something with v here //
    for all w in neighbors(v) do
       if w is not visited then
             mark w as visited
          Q.enqueue(w)
```

```
Q = stack()
Q.push(startNode)
mark startNode as visited
while Q is not empty do
    v := Q.pop()
    // do something with v here //
    for all w in neighbors(v) do
       if w is not visited then
             mark w as visited
          Q.push(w)
```



Reachability: Connected components

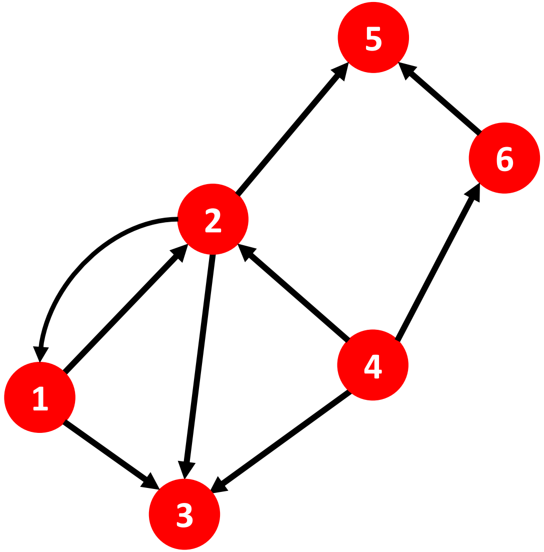
- **A connected component** is a portion of the graph where each node can reach all other nodes: pairwise reachable.

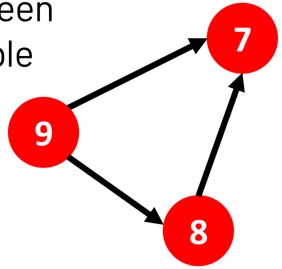
In a directed graph we can have connected components, but if we follow directions, then it may happen that we cannot reach all nodes.

- **A strongly connected component** is a portion of a directed graph where there is a directed path between any two nodes. All nodes are pairwise reachable when following directions.

- **A weakly connected component** is a portion of a directed graph where there is an **undirected** path between any two nodes. All nodes are pairwise reachable when **ignoring** directions.

UNDIRECTED GRAPH, USE BFS :
1) Start from a node;
2) Obtain all reachable nodes and mark them;
3) Increment CC counter
4) Take next node not already marked, and start again



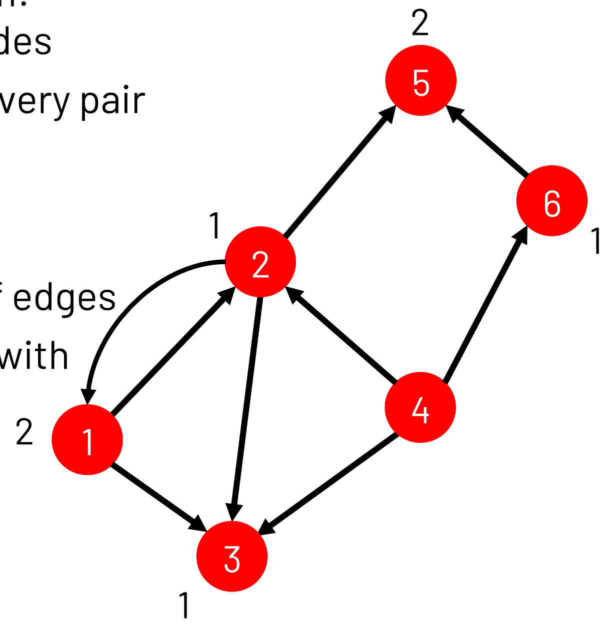


Graph Traversal: Shortest Path

- **Find the "quickest" way to reach nodes** *(Dijkstra's algorithm)*:
  - **Single source:** Given 1 source node find the "quickest" way to reach all other nodes
  - **Single Source-Destination** (pair of nodes) shortest path: find the "quickest" way – if exists – between the two nodes
  - **All-pairs shortest path:** find shortest paths between every pair of vertices in the entire graph
- **Definition of "quickest":**
  - All **edges cost the same** → find the smallest number of edges
  - Edges have **different cost** → weighted path, find path with minimum sum of edge weights

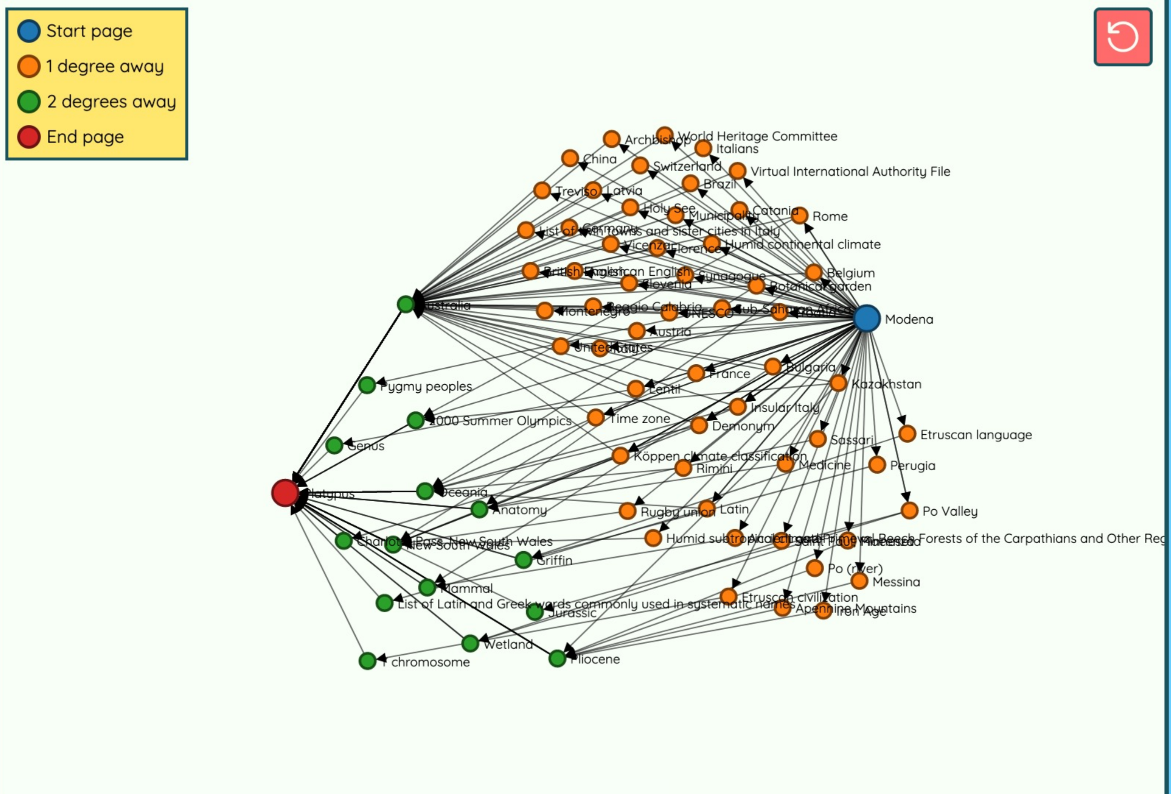
BFS & SHORTEST PATH

If all edges cost equal, BFS can compute the shortest path from one node to all other nodes



https://en.wikipedia.org/wiki/Shortest_path_problem#Single-source_shortest_paths

Found **76 paths** with **3 degrees** of separation from **Modena** to **Platypus** in **1.81 seconds!**



https://www.sixdegreesofwikipedia.com

Graph Queries: PGs and Triples

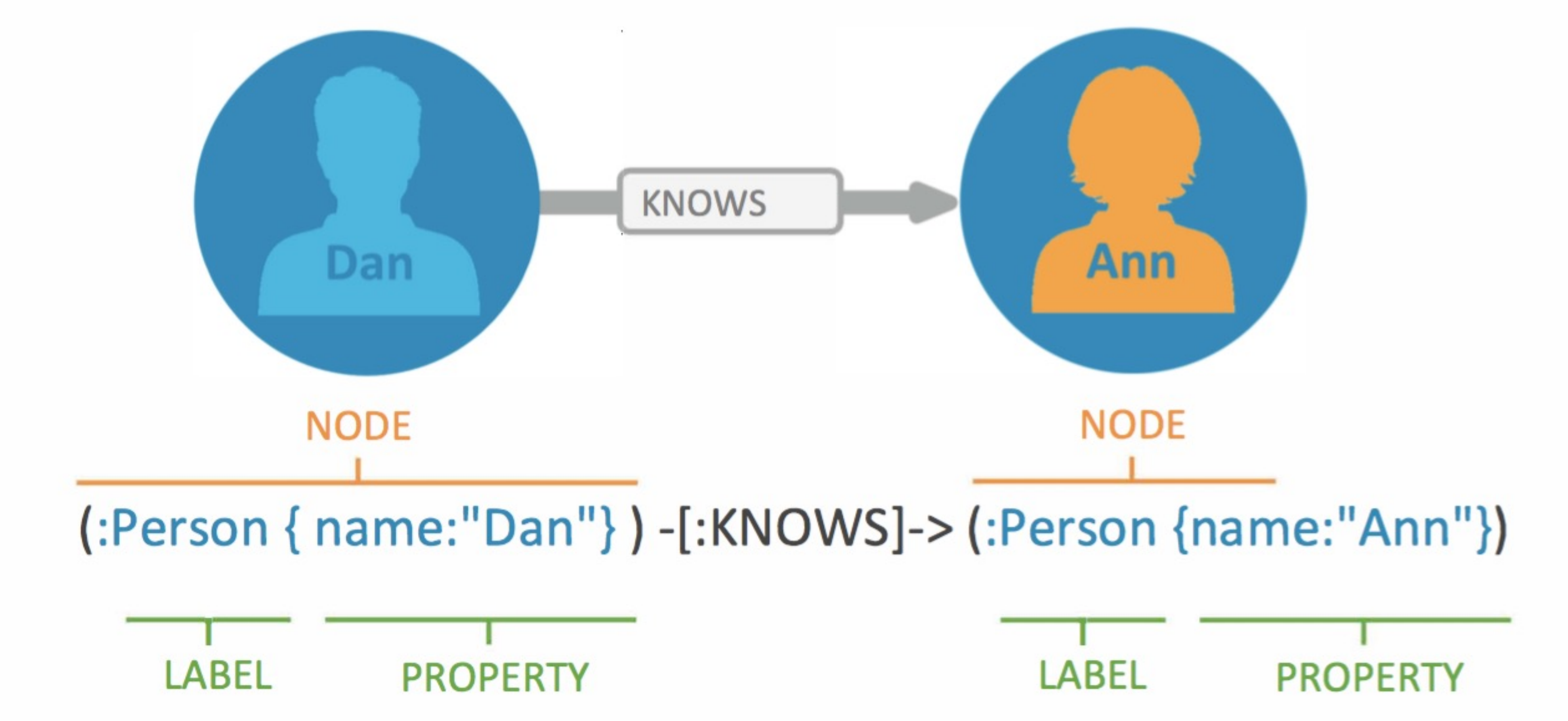
Different Data Models have different Query Paradigms

- **Property Graphs (PGs): everything is an “object” that can contain data**
  - Queries can retrieve: (a) nodes, (b) edges, (c) paths
  - Query language: CYPHER or GQL (gqlstandards.org) or GREMLIN

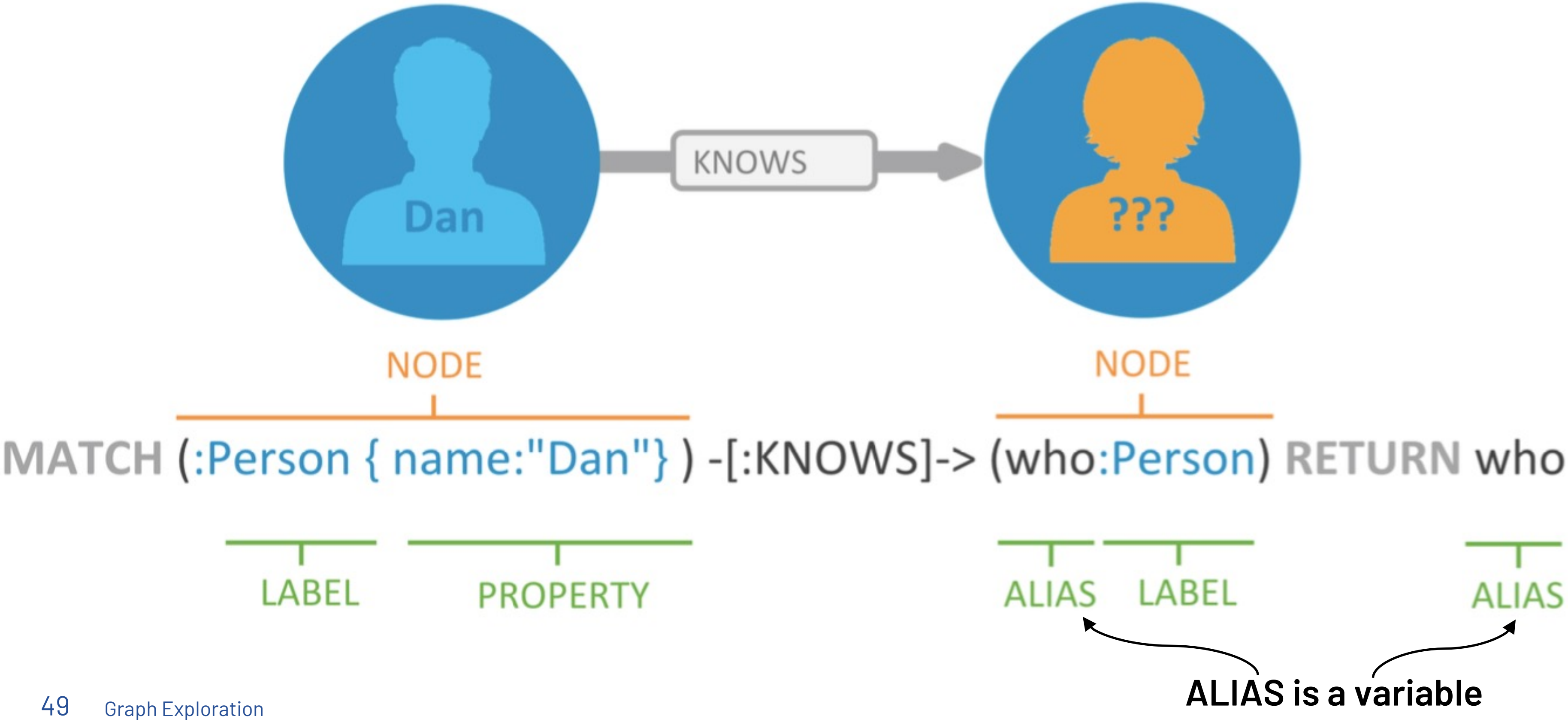
- **RDF (KGs): everything is a “triple” (a statement)**
  - Queries can only retrieve triples (matching paths / patterns)
  - Query language: SPARQL

OUTPUT:
A Graph query (PG or RDF) does not always (almost never) return a graph, usually they return tuples of variable assignments

Graph Example: PG



Graph Query Example



Graph Query Example (II)

```
//Find all the movies Tom Hanks directed and order by latest movie
MATCH (:Person {name:"Tom Hanks"})-[:DIRECTED]->(m:Movie)
RETURN m.title, m.released ORDER BY m.released DESC;
```

```
//Find all of the co-actors Tom Hanks have worked with
MATCH (th:Person{name:"Tom Hanks"})-->(:Movie)<-[:ACTED_IN]-(oth:Person)
WHERE th <> oth
RETURN oth.name;
```

```
MATCH pattern
WHERE predicate
ORDER BY expression
SKIP ... LIMIT ...
RETURN expression AS alias
```

Path pattern variations:

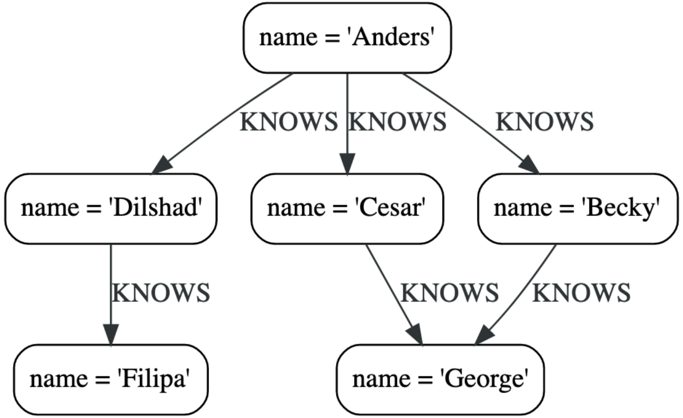
```
(n1)-[r1]->(n2)<-[r2]-(n3)
(n1)-[:KNOWS*]->(n2)
```

https://neo4j.com/developer/download-materials/

Graph Query Example (III)

```
 MATCH (node1:Person)-[:KNOWS]->(node2:Person)
       (node1)-[:LIVES_IN]->(node3:City)
       (node2)-[:LIVES_IN]->(node3)
 WHERE node1.age = 30
RETURN node1.name, node2.name, node2.age
```

Graph Query Example (IV) : Paths




```
MATCH (me)-[:KNOWS*1..2]-(other)
WHERE me.name = 'Filipa'
RETURN other.name
```

Results:
"Dilshad"
"Anders"

```
MATCH (me)-[:KNOWS*1..2]->(other)
WHERE me.name = 'Anders'
      AND other.name > ''F''
RETURN other.name
```

Results:
"George"

In Neo4j, all relationships have a direction. However, you can have the notion of undirected relationships at query time.

```
MATCH (me)-[:KNOWS*0..2]->(other)
RETURN COUNT(other.name)
```

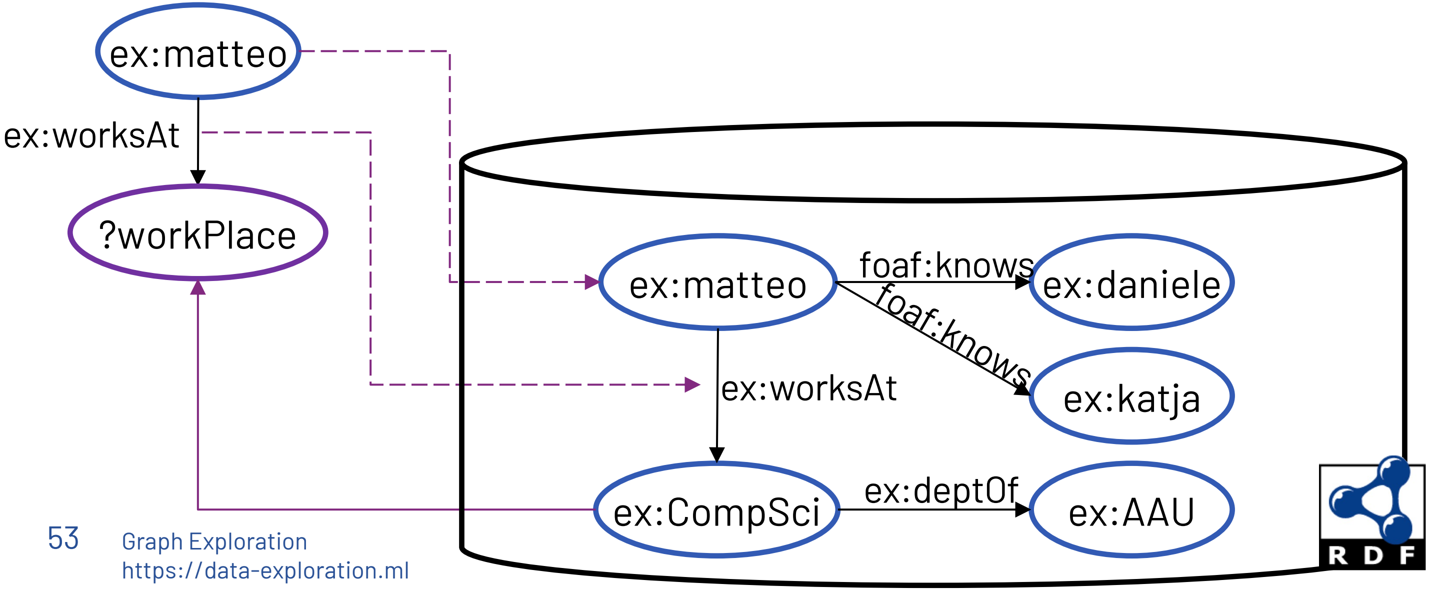
The SPARQL query language

ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/

The idea behind SPARQL as a query language is simple

- Define **patterns** & Patterns have **variables**
  - Everything in RDF is a triple ⟼ so we define **triple patterns**
- Identify portions of the RDF graphs that **match the pattern** ⟼ exists a valid **assignment**

Query:



Evaluation result:

{?workplace -> ex:CompSci}

or

?workplace
ex:CompSci

Thanks Daniele Dell'Aglio, AAU

The SPARQL query language (II)

```
ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/
```

- **match the pattern** $\longmapsto$ exists a valid **assignment?**

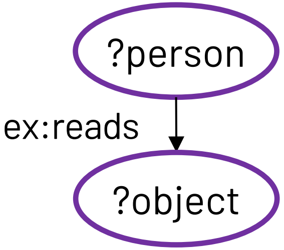
Evaluation result:

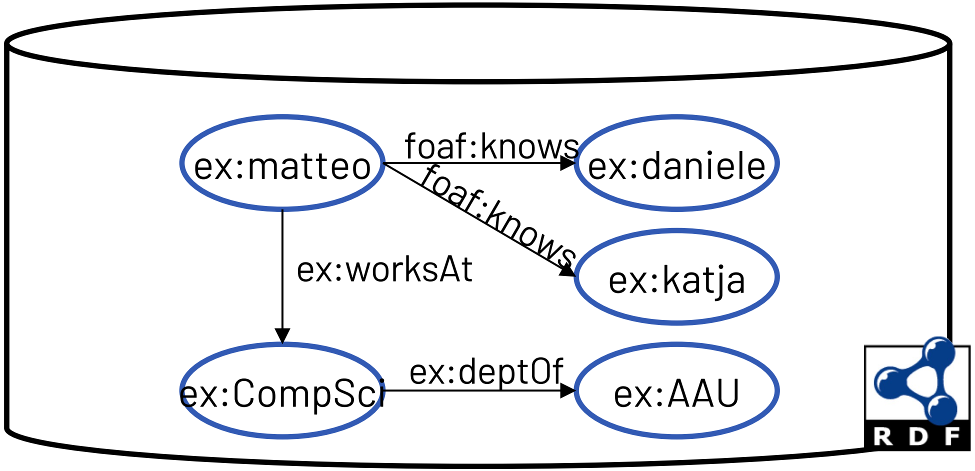
{},

or

?person	?object

Query:



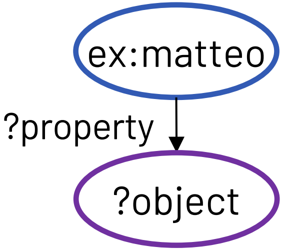


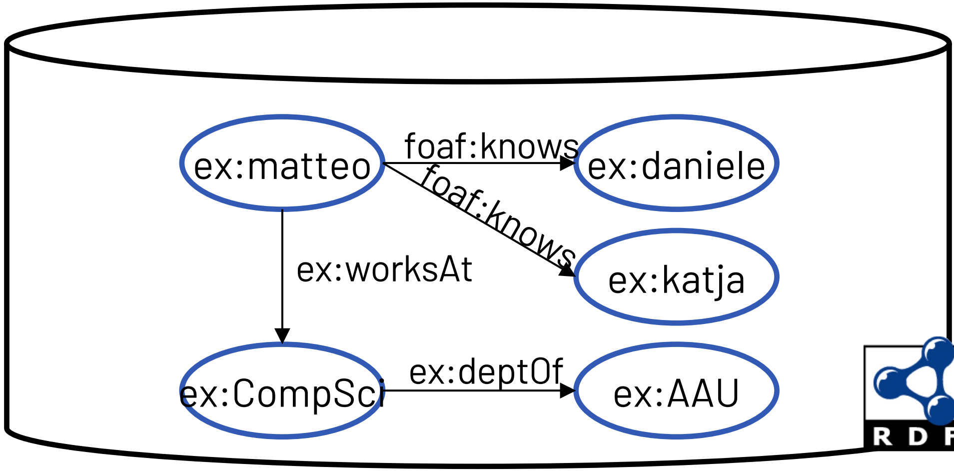
Thanks Daniele Dell'Aglio, AAU

The SPARQL query language (III)

`ex -> http://example.org#`
`foaf: -> http://xmlns.com/foaf/0.1/`

Query:





Evaluation result:

{?property -> ex:worksAt ; ?object -> ex:CompSci},
{?property -> foaf:knows ; ?object -> ex:daniele},
{?property -> foaf:knows ; ?object -> ex:katja}
or

?property	?object
ex:worksAt	ex:CompSci
foaf:knows	ex:daniele
foaf:knows	ex:katja

Thanks Daniele Dell'Aglio, AAU

Triple patterns and solution mappings

Triple pattern: an RDF triple where one or more nodes are variables

Variables are denoted by ? (or $) at their beginning

1. ex:matteo ex:worksAt **?workPlace**
2. ex:matteo **?property ?object**
3. **?person** ex:reads **?book**

Evaluating a triple pattern over an RDF graph produces a **multiset (bag) of solution mappings**

1. {?workplace -> ex:IFI}
2. {?property -> ex:worksAt ; ?object -> ex:CompSci},
   {?property -> foaf:knows ; ?object -> ex:daniele},
   {?property -> foaf:knows ; ?object -> ex:katja}
3. { }

ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/

Thanks Daniele Dell'Aglio, AAU

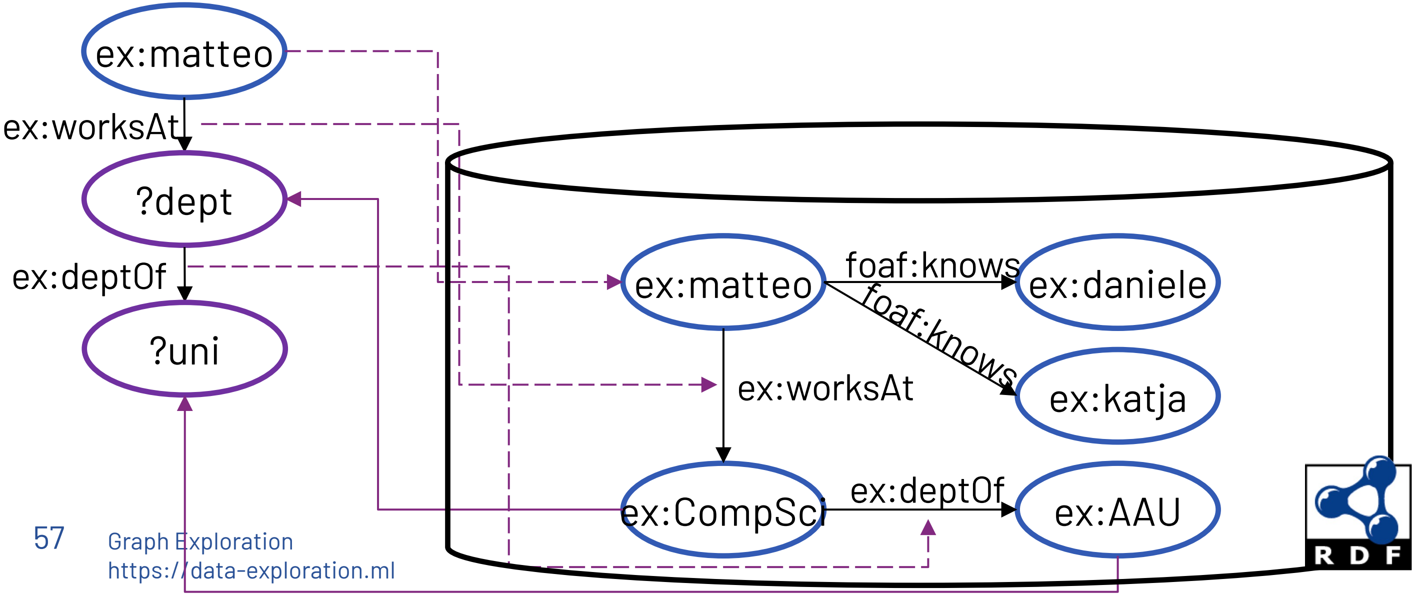
Basic graph patterns

ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/

Basic graph pattern (BGP): a set of one or more triple patterns
(with optional FILTER clauses)

Evaluation result:
{?dept -> ex:CompSci ; ?uni -> ex:AAU}

Query:



Thanks Daniele Dell'Aglio, AAU

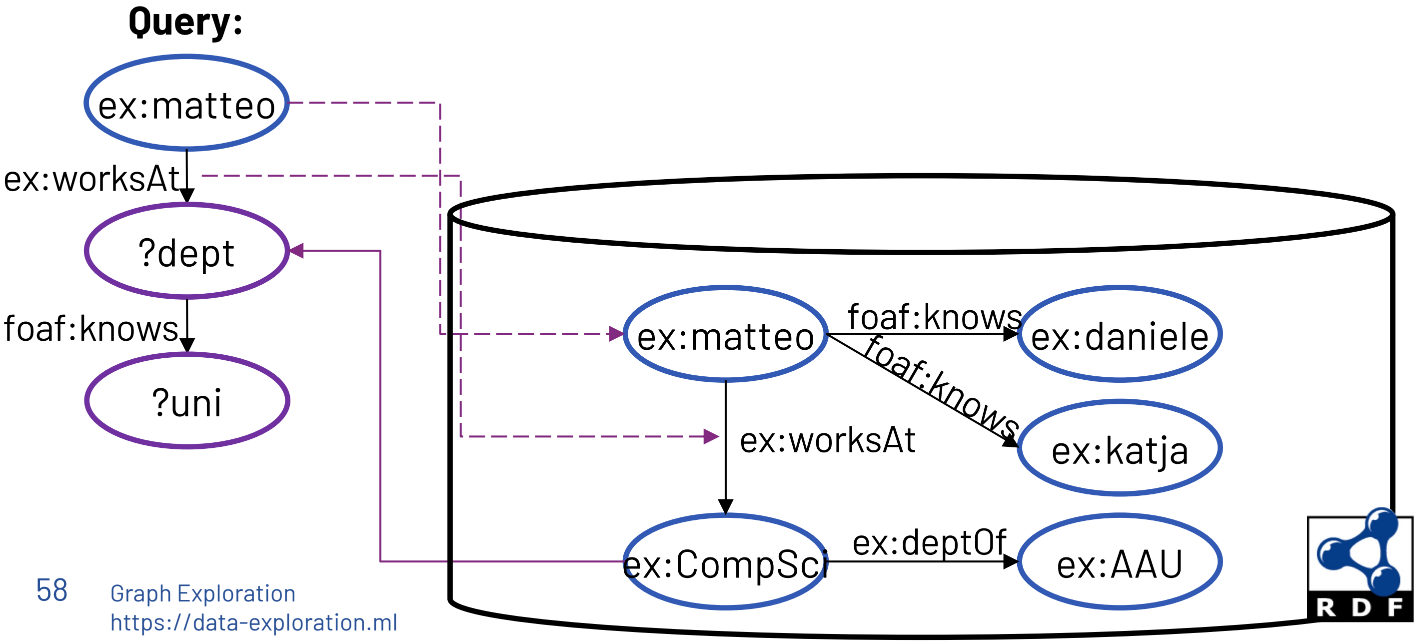
Basic graph patterns (II)

ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/

All the triple patterns in the BGP should match to create a result!

Evaluation result:

{}



Thanks Daniele Dell'Aglio, AAU

Basic graph pattern – syntax

```
ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/
```

Basic graph patterns are written in a Turtle-like style:

A set of triple patterns separated by dots (i.e., AND)

ex:matteo ex:worksAt ?dept . ?dept ex:deptOf ?uni .

Shared variables refer to the same node in the graph (like joins)

Turtle abbreviations (using **;** and **,**) can be used

ex:matteo ex:worksAt ?dept **;**

foaf:knows ?person1 **,** ?person2 **.** FILTER (?person1 != ?person2)

(evaluation result:
{?dept -> ex:CompSci ; ?person1 -> ex:daniele ; ?person2 -> ex:katja},
{?dept -> ex:CompSci ; ?person1 -> ex:katja ; ?person2 -> ex:daniele }
)

Thanks Daniele Dell'Aglio, AAU

FILTER

- FILTER denotes selection in relational algebra

```
FILTER (?person1 != ?person2)
```

- FILTER allows to specify common **unary/binary operators**:
  - *Less than, greater than, equalities for integer, decimals and date/time*
  - **Conditions over strings**: *Regular expressions*
  - A list of **functions** for specific situations:
    isURI, isIRI, isBlank, isLiteral, isNumeric
  - Selection for **lang & datatype**

```
FILTER ( isLiteral(?age)
         && datatype(?age) = xsd:integer )
         && ?age > 30)
)
```

More at: https://www.w3.org/TR/sparql11-query/#func-rdfTerms

SPARQL query with BGP – syntax

The query structure is similar to SQL:

SELECT [FROM] WHERE

```
PREFIX ex: <http://example.org#>      ←Prefixes
SELECT ?uni        ← Variable(s) of interest (projection)
WHERE {
      ex:matteo ex:worksAt ?dept .
      ?dept ex:deptOf ?uni .           ← BGP (join)
}
```


Thanks Daniele Dell'Aglio, AAU

Property path: Sequence and alternative paths

Property path allows to define routes between nodes

- Sequence path /

```
PREFIX ex: <http://example.org#>
SELECT ?uni
WHERE {
 ex:matteo ex:worksAt/ex:deptOf ?uni .
}
```

```
PREFIX ex: <http://example.org#>
SELECT ?uni
WHERE {
  ex:matteo ex:worksAt ?loc .
  ?loc ex:deptOf ?uni .
}
```

- Alternative path |

```
PREFIX ex:
<http://example.org#>
SELECT ?x ?addr
WHERE {
  ?x ex:zip|ex:address ?addr .
}
```

```
PREFIX ex:
<http://example.org#>
SELECT ?x ?addr
WHERE {
  { ?x ex:zip ?addr . }
  UNION
  { ?x ex:address ?addr . }
}
```

ex -> http://example.org#
foaf: -> http://xmlns.com/foaf/0.1/

Property path: Arbitrary length path

- Specify path of variable length
- One or more path +

  { ex:matteo foaf:knows+ ?person . }

  {?person -> ex:daniele}, {?person -> ex:katja}, {?person -> ex:alice}, {?person -> ex:bob}

- Zero or one path ?

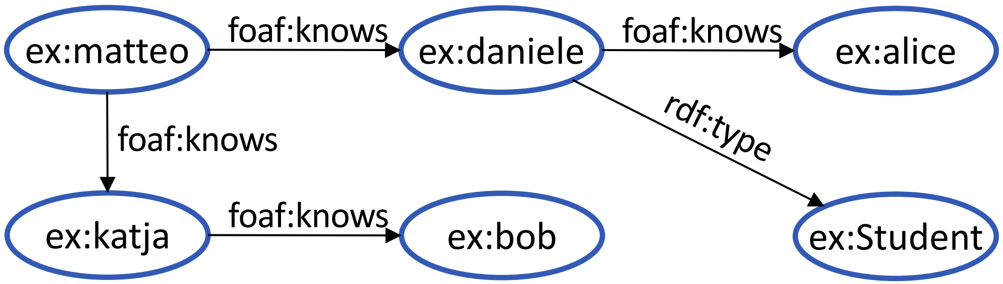
  { ex:matteo foaf:knows? ?person . }

  {?person -> ex:matteo}, {?person -> ex:daniele} {?person -> ex:katja}

- Zero or more path *

  { ex:matteo foaf:knows* ?person . }

  {?person -> ex:daniele}, {?person -> ex:katja}, {?person -> ex:alice}, {?person -> ex:bob}, {?person -> ex:matteo}



Property Graph Query Language: Gremlin

Gremlin
G = (V, E)

Imperative Graph Traversal

```
g.V().has(‘name’, ‘Tom Hanks’).out() .values("name");

g.V().has(‘name’, ‘Tom Hanks’).out().out().out().values("name");
```

Declarative Graph Traversal

```
g.V().match(
  as("a").has("name", "Tom Hanks"), as("a").out("directed").as("b"),
  as("b").in("acted_in").as("c"), where("a",neq("c"))
).values("name")
```

Try out: https://gremlify.com/

https://kelvinlawrence.net/book/Gremlin-Graph-Guide.html

Let’s model this as a graph (1)

You are helping organize a conference

and want to model the data about its participants.

You have the citation network of all the people

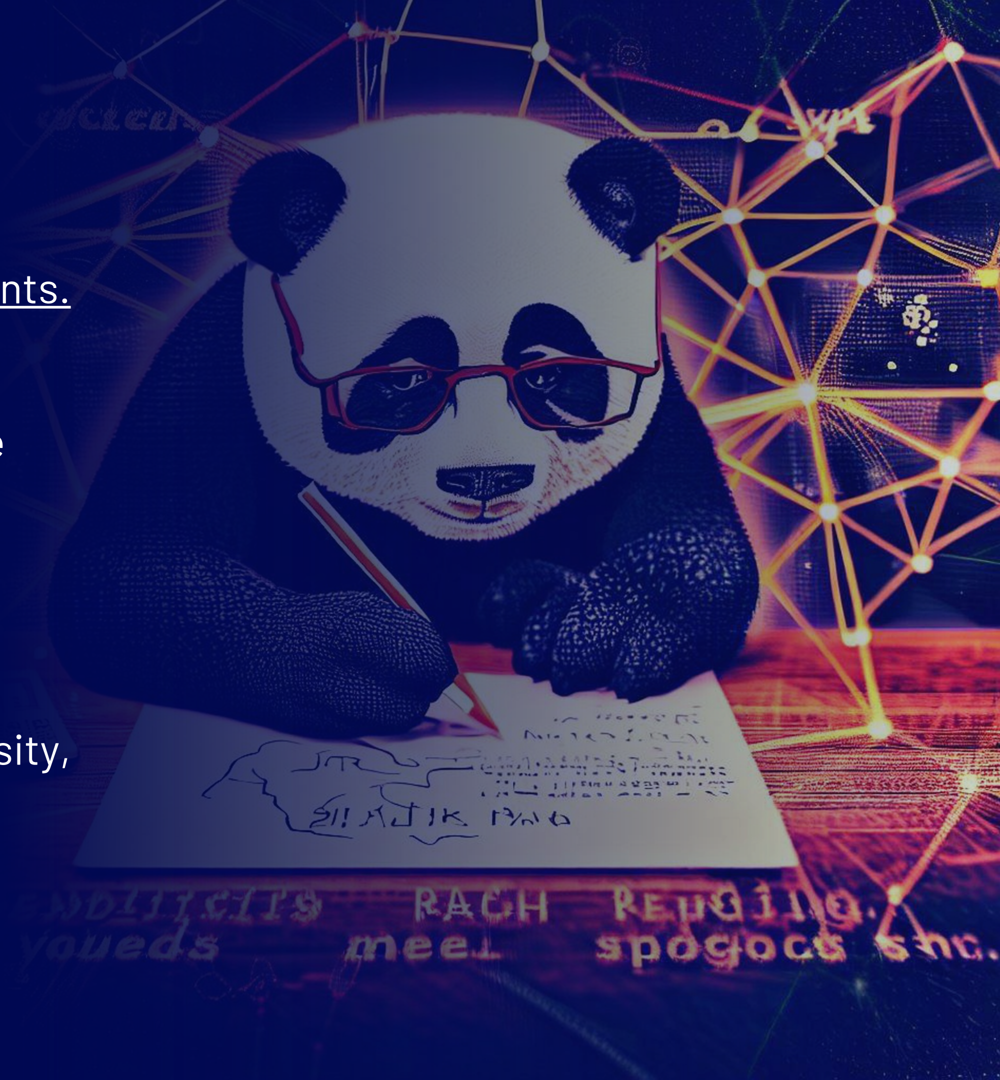
at the conference.

There are **papers**, **authors**, and **universities**.

You know which author **works in** which university,

which author **wrote** which paper, and which

paper **cited** which paper.



Let’s model this as a graph (2)

You are tracking users on a complex website.

They **visit many pages** of the website to complete their work, when done they close the website.

For each user you know on which **page they start**, what **action** they take, on which **page they end**.

Then from that page they can take another action and **go to another page.** They can never go to random pages.

They **can also go back** to the previous page and do an additional actions and thus may end up in a different page from there.

Graph Data Analysis & Exploration

— Network Analysis —

Matteo Lissandrini – Aalborg University

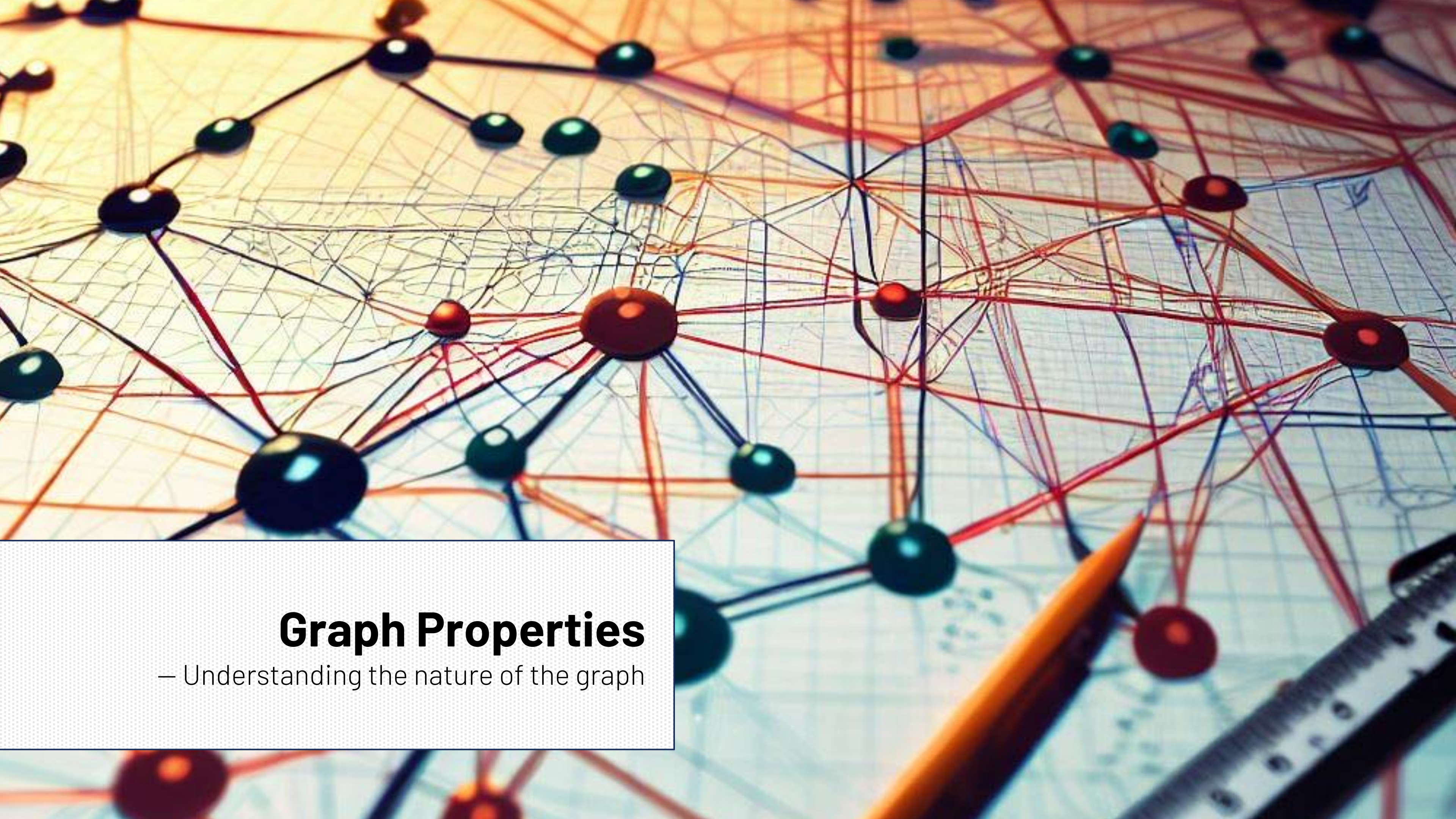
AALBORG UNIVERSITY

Outline

1. **Graph Properties**
   - Scale Free Networks
   - Preferential Attachment
   - Small world property
   - Erdös Number
   - Density/Diameter/Eccentricity
   - Clustering Coefficient/ Wiener Index

2. **Centrality Measures**
   - Degree/Closeness
   - Betweenness Centrality
   - Katz Centrality
   - Prestige / H-index

3. **Page Rank**
   - Random Walk & Transition Probability
   - Markov Model
   - Algebraic representation
   - Power Iteration
   - Personalized Page Rank
   - Particle Filtering
   - SimRank



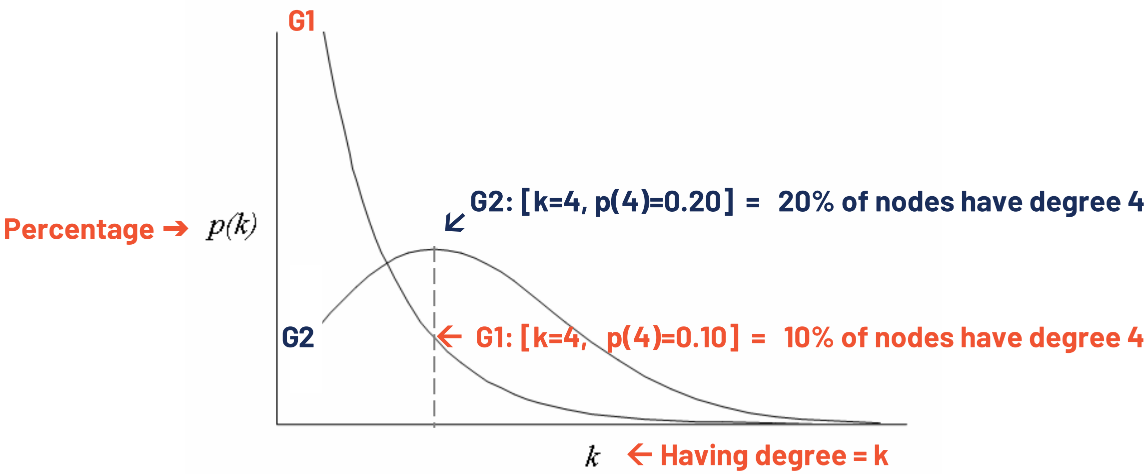
Graph Properties

— Understanding the nature of the graph

Degree Distribution

- **Node Degree:** number of nodes connected
- **What is the Degree Distribution in a Graph?**

  Plot the ratio of nodes having a specific Node-Degree

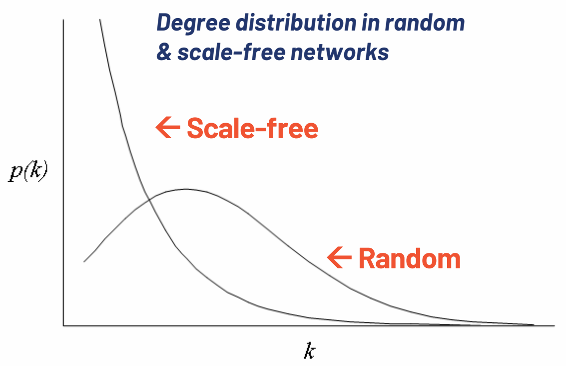


Scale Free Network/Graph

- **Node Degree:** number of nodes connected
- **What is the Degree Distribution in a Graph?**

Plot Number of nodes having a specific Node-Degree

$P(k)$ *proportion of nodes with degree =* k



A scale-free network is a network whose degree distribution follows a power law.

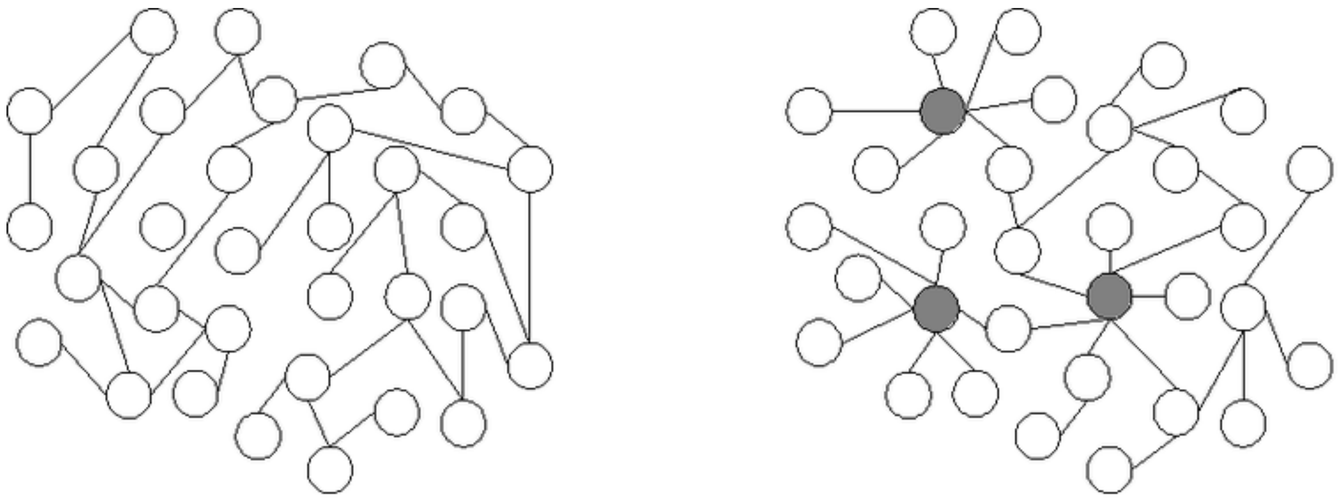
The fraction P(k) *of nodes in the network having* k *connections to other nodes follows approximately*

$$P(k) \sim k^{-\gamma}$$

Typically $2 < \gamma < 3$

https://en.wikipedia.org/wiki/Scale-free_network

Scale Free Network/Graph



(a) Random network

(b) Scale-free network

Few highly connected hubs

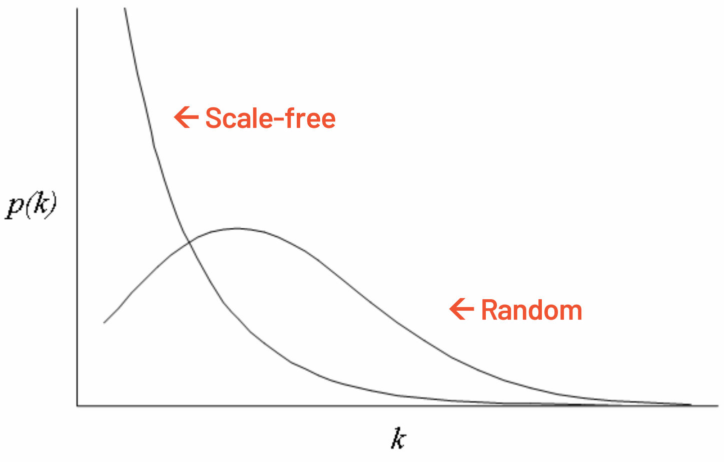
https://en.wikipedia.org/wiki/Scale-free_network

Power Law: meaning

Power law degree distribution: large events are rare, but small ones are quite common.

The probability of finding a highly connected node decreases exponentially with k
(degree of node, inversely proportional to k):

$$P(k) \sim k^{-\gamma} = \frac{1}{k^{\gamma}}$$



Degree distribution in random & scale-free networks

P(K) = $K^{2.1}$

K= 1	P(K)=1.0	K= 2	P(K)=0.233258
K= 5	P(K)=0.0340536	K= 10	P(K)=0.007943
K=20	P(K)=0.0018528	K=100	P(K)=0.000063

The Faloustos-cubed paper

"On Power-Law Relationships of the Internet Topology"
by Faloutsos, Michalis; Petros Faloutsos; and Christos Faloutsos.

https://dl.acm.org/doi/pdf/10.1145/316194.316229

https://en.wikipedia.org/wiki/Scale-free_network

Cause of Scale-free: Preferential attachment

Rich gets Richer

1. New nodes are added to the network one at a time.
2. Each new node is connected to existing ***nodes with a probability that is proportional to the number of links*** that the existing nodes already have

– **Barabási–Albert model**

the probability p_i that the new node is connected to node i

$$p_i = \frac{k_i}{\sum_j k_j}$$

← **Sum of all degrees = 2*|E|**

where k_i is the degree of node i

Random Graphs instead follow The Erdos-Renyi model

https://en.wikipedia.org/wiki/Preferential_attachment
https://en.wikipedia.org/wiki/Barab%C3%A1si%E2%80%93Albert_model

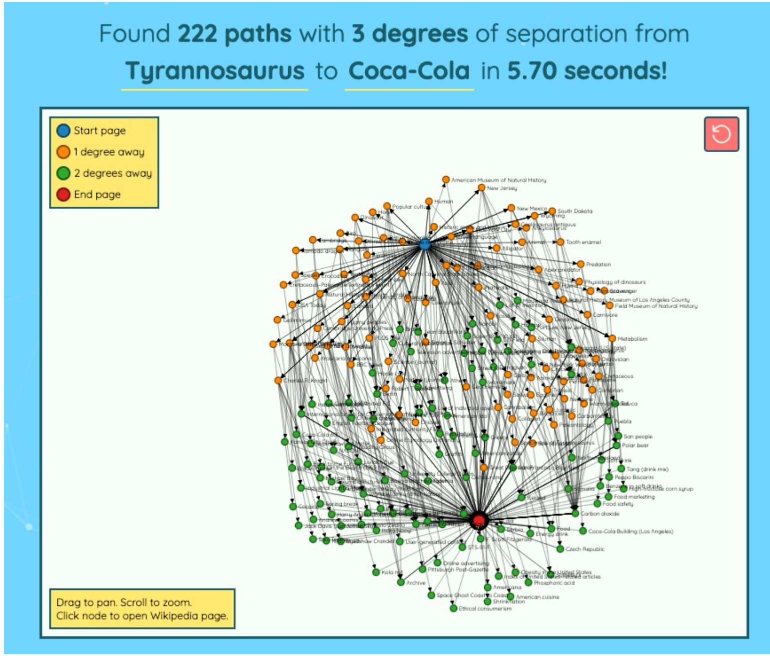
Small World Property

Shortest path: the path with the smallest number of links (edges) between 2 selected nodes.

Small world networks:

the average shortest path length between any two nodes in the network is relatively small.
Any node can be reached within a small number of edges, e.g., 4~5 hops.

7 degrees of separation: *in a social network there are at most 7 "handshakes" between you and any other person in the world*



https://www.sixdegreesofwikipedia.com

Erdős Number

the "collaborative distance" between mathematician Paul Erdős and another person

Erdős number, *the number of steps in the shortest path between a mathematician and Erdős in terms of co-authorships.*

Co-author/Collaboration Network: *An undirected graph representing authors as nodes, an edge exists between A and B if it exist a publication where A and B are co-authors*

Citation Network: a directed graph representing scientific publications as nodes, an edge goes from A to B if A has a reference to B. This is a ***directed acyclic graph*** (DAG)

Similar Concept: Bacon Number
https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon

Fun Fact: Natalie Portman has both Erdős Number and Bacon Number!

https://en.wikipedia.org/wiki/Paul_Erd%C5%91s

Paul Erdős in 1992
authored~ 1,500
mathematical papers

https://oakland.edu/enp/compute/
https://www.csauthors.net/distance/

How Compact is a Graph? (I)

- **Eccentricity of node:** the greatest distance between a node $\mathrm{N_i}$ and any other vertex

$$\text{Eccentricity}(1) = 3$$

- **Radius of a graph:** the minimum eccentricity of any node

$$\text{Radius} = 2$$

- **The diameter of a graph:** the maximum eccentricity of any vertex in the graph. (the maximum distance between any 2 nodes)

$$\text{Diameter} = 4$$

- **Density of a graph:** fraction between number of edges and maximal number of edges

$$\text{Density} = 2*11/56 = 0.39$$

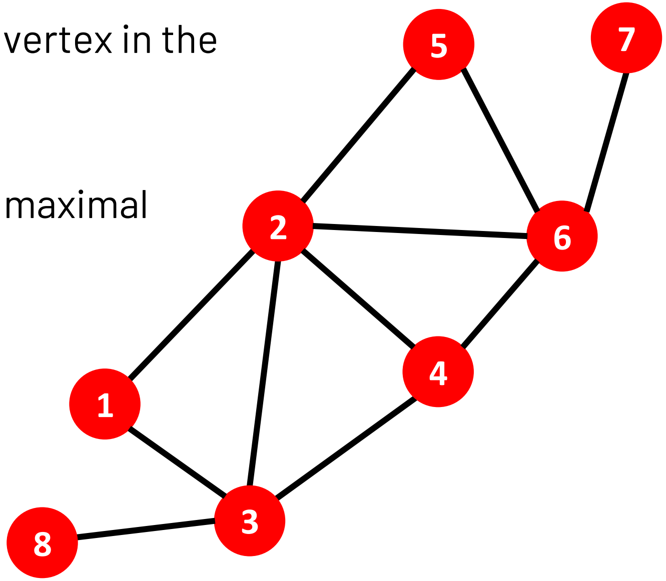
$$D = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)}$$

Undirected

$$D = \frac{|E|}{2\binom{|V|}{2}} = \frac{|E|}{|V|(|V|-1)}$$

Directed

These measures ignore directions of edges (except for density)



https://en.wikipedia.org/wiki/Combination

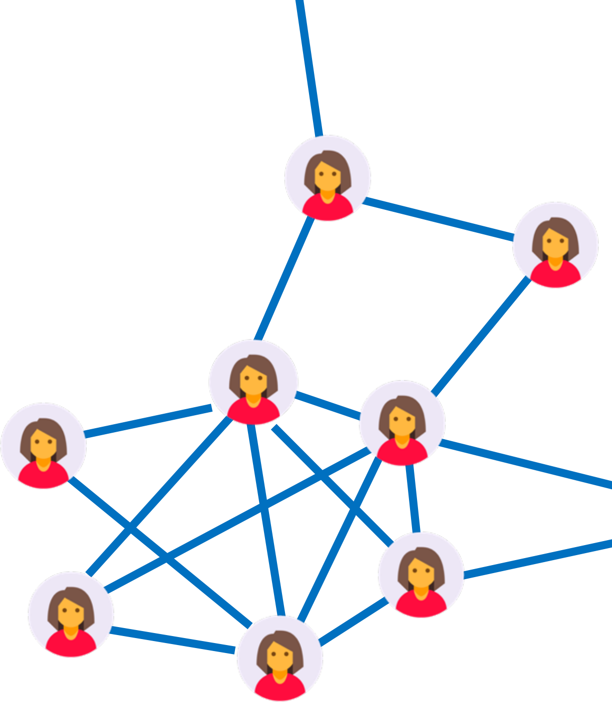
Wiener Index: Closeness of a graph

How tightly connected is a graph?

Wiener Index:
the sum of pairwise shortest-path-distances between nodes in the graph G

$$\sum_{(u,v) \in G} d(u, v)$$

d(u, v) is the shortest-path distance

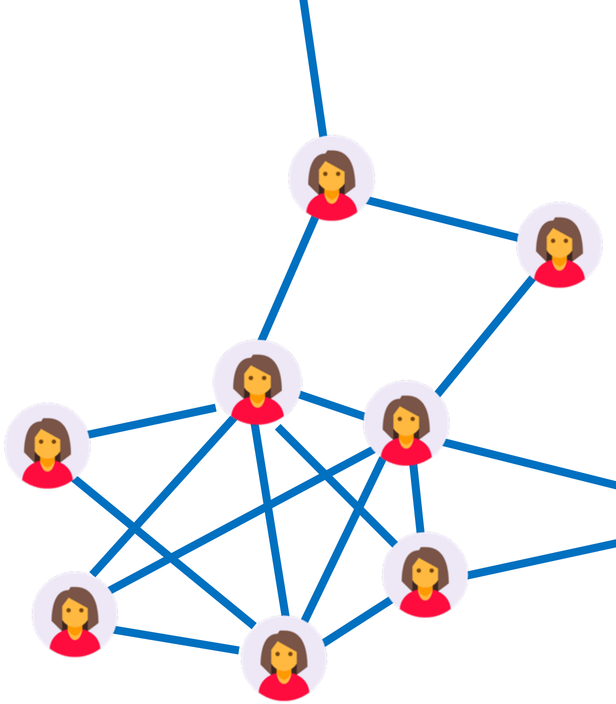


How Compact is a Graph? (II)

Characterize the structure of a graph:

1. **Average Diameter L:** *average length of the shortest paths connecting any two nodes*
2. *Effective Diameter: 90th Percentile of shortest path length*
3. **Clustering coefficient C:** *the average local density* (see next slide).

Small World Graphs have relatively small *L* & a relatively large *C*.



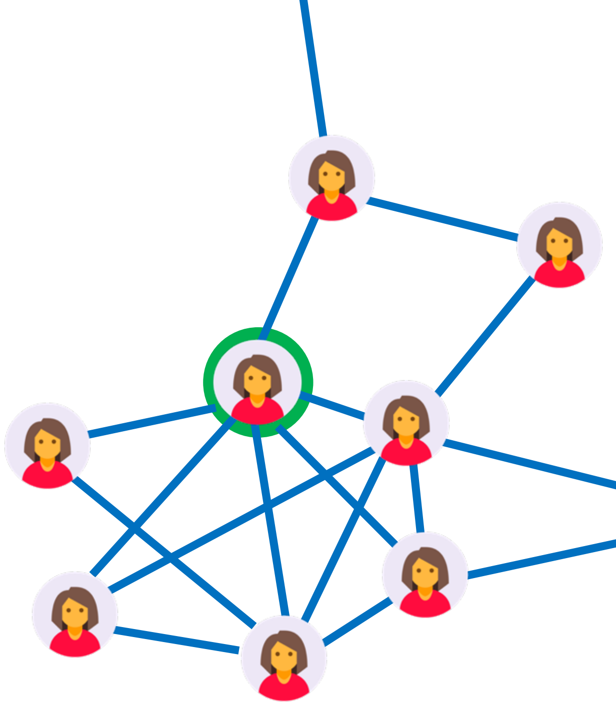
Clustering Coefficient: Local Density

How dense is the neighborhood of a <u>node</u>:
The fraction pairs of neighbors of the node that are themselves connected

Density of a graph: fraction between number of **<u>existing</u>** edges and **<u>maximal</u>** number of edges

The clustering coefficient is Equivalent to the density of the subgraph when considering ONLY the neighbors of n *(ignoring* n *)*

Given node **n**



$$C_n = \frac{\#\text{ edges between the neighbors of } \mathbf{n}}{\text{degree}(\mathbf{n}) * (\text{degree}(\mathbf{n}) - 1)}$$

Directed $D = \frac{|E|}{2\binom{|V|}{2}} = \frac{|E|}{|V|(|V|-1)}$

$$C_n = \frac{2 * (\#\text{ edges between the neighbors of } \mathbf{n})}{\text{degree}(\mathbf{n}) * (\text{degree}(\mathbf{n}) - 1)}$$

Undirected $D = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)}$

Clustering Coefficient: Average Local Density

How dense is the neighborhood of a node:
The fraction pairs of neighbors of the node that are themselves connected

Density of a graph: fraction between number of edges and maximal number of edges

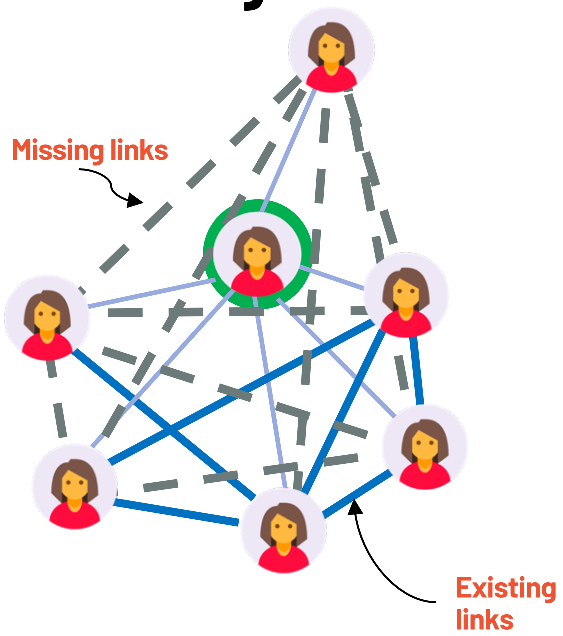
Given node **n**

$$C_n = \frac{2 * \#\ \text{edges between neighbors of } \mathbf{n}}{\text{degree}(\mathbf{n}) * (\text{degree}(\mathbf{n}) - 1)}$$

Undirected

Clustering Coefficient *as average of the entire graph*

$$C = \frac{1}{|N|} \sum_{n \in N} C_n$$



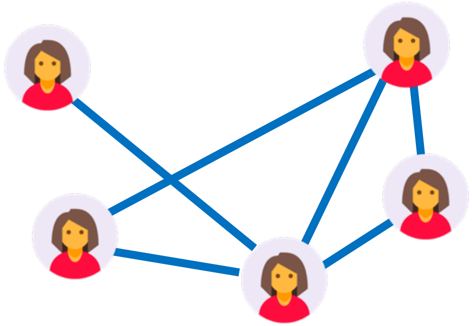

```
Degree(n) = 6
Max connections = (6*5) = 30
Existing links = 6
```

C_n = 2*6/30 = 0.4

Clustering Coefficient: Average Local Density

How dense is the neighborhood of a node:
The fraction pairs of neighbors of the node that are themselves connected

Subgraph considering only the neighbors of n



Density of a graph: fraction between number of edges and maximal number of edges

Given node **n**

$$C_n = \frac{2 * \#\ \text{edges between neighbors of } \mathbf{n}}{\text{degree}(\mathbf{n}) * (\text{degree}(\mathbf{n}) - 1)}$$

Undirected

```
Consider Undirected
Degree(n) = 6
Max connections = (6*5) = 30
Existing links = 6
```
C_n = 2*6/30 = 0.4

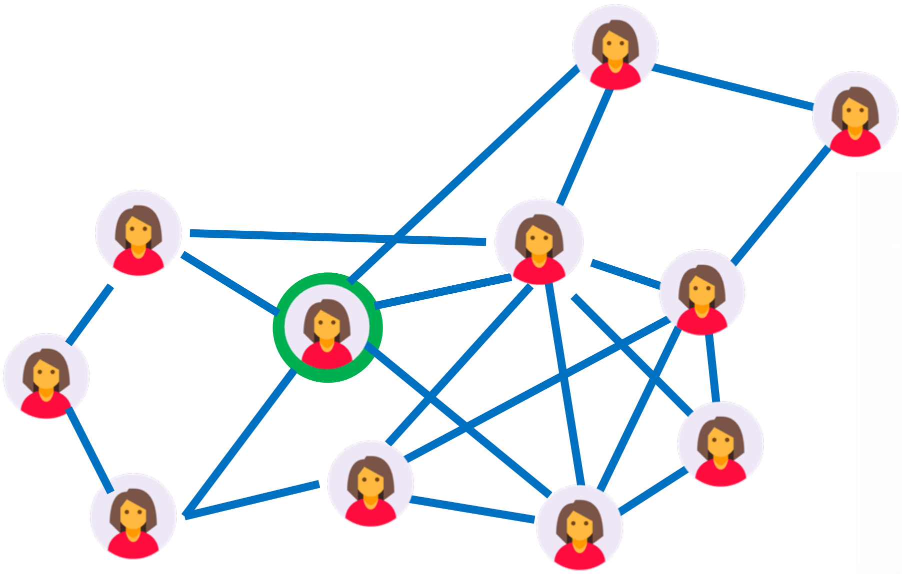
Clustering Coefficient
as average of the entire graph

$$C = \frac{1}{|N|} \sum_{n \in N} C_n$$

Compute the Clustering Coefficient

$$C_n = \frac{2 * \#\ \text{edges between neighbors of}\ \mathbf{n}}{\text{degree}(\mathbf{n}) * (\text{degree}(\mathbf{n}) - 1)}$$

Undirected



Outline

1. **Graph Properties**
   - Scale Free Networks
   - Preferential Attachment
   - Small world property
   - Erdös Number
   - Density/Diameter/Eccentricity
   - Clustering Coefficient/ Wiener Index

2. **Centrality Measures**
   - Degree/Closeness
   - Betweenness Centrality
   - Katz Centrality
   - Prestige / H-index

3. **Page Rank**
   - Random Walk & Transition Probability
   - Markov Model
   - Algebraic representation
   - Power Iteration
   - Personalized Page Rank
   - Particle Filtering
   - SimRank

Outline

1. **Graph Properties**
   - Scale Free Networks
   - Preferential Attachment
   - Small world property
   - Erdös Number
   - Density/Diameter/Eccentricity
   - Clustering Coefficient/ Wiener Index

2. **Centrality Measures**
   - Degree/Closeness
   - Betweenness Centrality
   - Katz Centrality
   - Prestige / H-index

3. **Page Rank**
   - Random Walk & Transition Probability
   - Markov Model
   - Algebraic representation
   - Power Iteration
   - Personalized Page Rank
   - Particle Filtering
   - SimRank

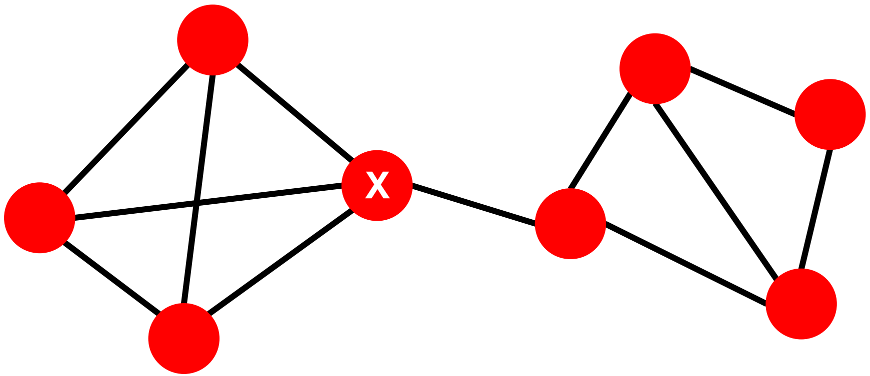
Centrality Measures

— Measuring Importance

Importance of a Node

How important is a node in a graph?

Centrality intuition:
The importance of a node depends on its role in "keeping the graph connected"



Basic Centrality measures

Directed Degree:
In a directed graph we can differentiated in-degree vs. out-degree.

How important is a node in a graph?

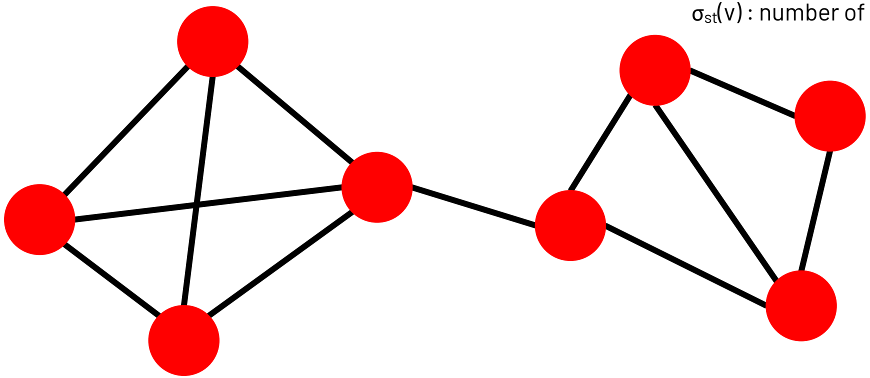
1. **Degree centrality:** number of neighbors of node v
2. **Closeness centrality:** reciprocal of the total distance from a node v to all the other nodes in a network
3. **Betweenness centrality:** ratio of the number of shortest paths passing through a node v out of all shortest paths between all node pairs in a network

$$C_c(v) = \frac{1}{\sum_{u \in V} \delta(u, v)}$$

δ(u,v) is the distance between node *u* and *v*.

$$C_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σ_{st} : number of shortest paths between node s and t
$\sigma_{st}(v)$: number of shortest paths passing on a node v out σ_{st}



Connected graphs:
These measure have meaning only when referring to a connected graphs

Outline

1. **Graph Properties**
   - Scale Free Networks
   - Preferential Attachment
   - Small world property
   - Erdös Number
   - Density/Diameter/Eccentricity
   - Clustering Coefficient/ Wiener Index
2. **Centrality Measures**
   - Degree/Closeness
   - Betweenness Centrality
   - Katz Centrality
   - Prestige / H-index

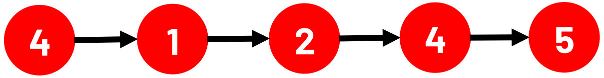
3. **Page Rank**
   - Random Walk & Transition Probability
   - Markov Model
   - Algebraic representation
   - Power Iteration
   - Personalized Page Rank
   - Particle Filtering
   - SimRank

Outline

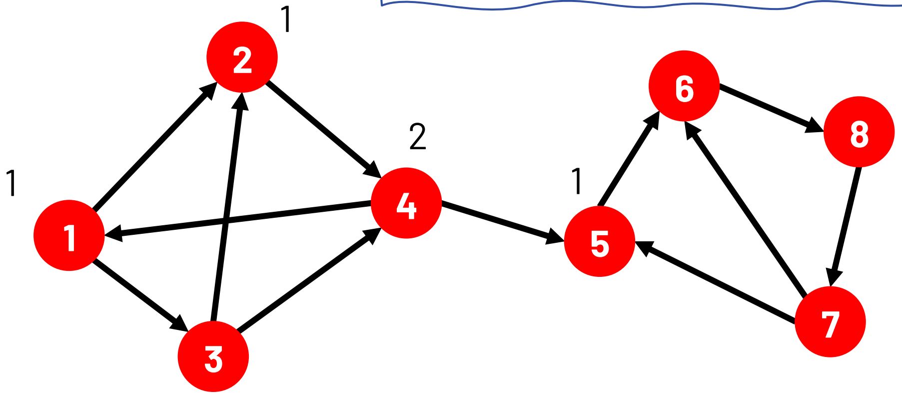
1. **Graph Properties**
   - Scale Free Networks
   - Preferential Attachment
   - Small world property
   - Erdös Number
   - Density/Diameter/Eccentricity
   - Clustering Coefficient/ Wiener Index
2. **Centrality Measures**
   - Degree/Closeness
   - Betweenness Centrality
   - Katz Centrality
   - Prestige / H-index
3. **Page Rank**
   - Random Walk & Transition Probability
   - Markov Model
   - Algebraic representation
   - Power Iteration
   - Personalized Page Rank
   - Particle Filtering
   - SimRank

The Random Walk

1. **Pick a node**
2. **Select a neighbour at random:** take a step
3. Keep making steps until we are "tired"
4. **Take note of the node where we stop and how often we visit each node**



Random Walk: *traversal of the graph by selecting neighbours at random. It is possible to visit the same edge/node multiple times. We keep note of the "frequency" with which each node is visited*



Directed Graph:
We need to follow the directions

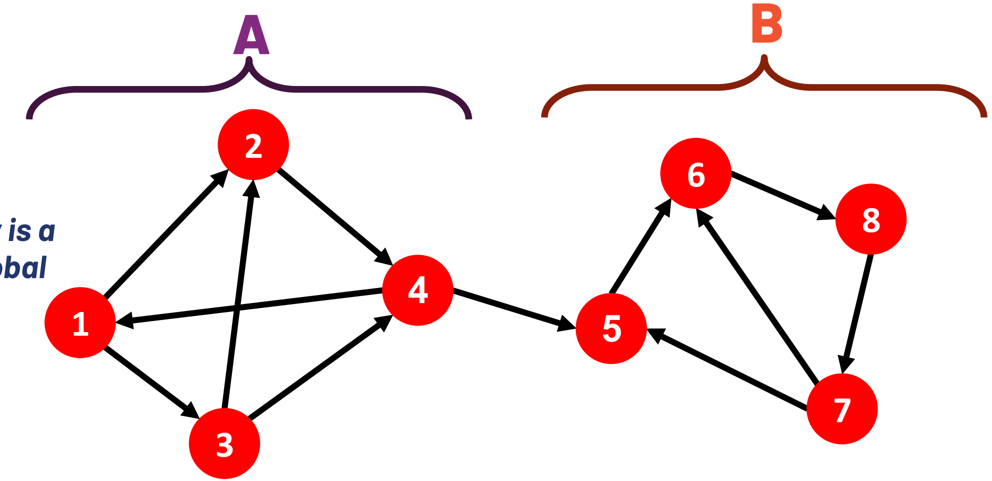
The Random Walk Gamble

Let's play a game

1. **I pick a random node** (not telling which one)

2. **I perform a random walk** (not telling how many steps, let's say >3)
3. Your guess: where am I on the graph? Group A or Group B

This probability is a measure of "global centrality"



Algebraic Representation via the Markov model

Given the **Transition probability matrix** T & the initial **vector of probabilities** v of each node

We can account for **the teleport probability** α so that

- 1 **step of the process from time** $\mathrm{t_i}$ **to time** $\mathrm{t_{i+1}}$ **corresponds to the multiplication:** $(1\text{-}\alpha)\mathrm{T}^{\mathrm{T}}\times v_i + \alpha\times v_0$

$\alpha = 0.1$

$$
(1-\alpha)\underbrace{\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & 0 & 0 & 1 \\
2 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\
3 & \frac{1}{2} & 0 & 0 & 0 \\
4 & 0 & 1 & \frac{1}{2} & 0
\end{array}}_{\mathrm{T}^{\mathrm{T}}}
\times
\underbrace{\begin{array}{c|c}
 & p \\
\hline
1 & \frac{1}{4} \\
2 & \frac{1}{4} \\
3 & \frac{1}{4} \\
4 & \frac{1}{4}
\end{array}}_{v_0}
+ \alpha
\underbrace{\begin{array}{c|c}
 & p \\
\hline
1 & \frac{1}{4} \\
2 & \frac{1}{4} \\
3 & \frac{1}{4} \\
4 & \frac{1}{4}
\end{array}}_{v_0}
=
\underbrace{\begin{array}{c|c}
 & p \\
\hline
1 & \frac{1}{4} \\
2 & \frac{1}{4} \\
3 & 29/80 \\
4 & 11/80
\end{array}}_{v_1}
$$

During iteration the vector for the teleport stays the same we update only the vector multiplying the matrix

Power iteration method

Page Rank: Importance Flow

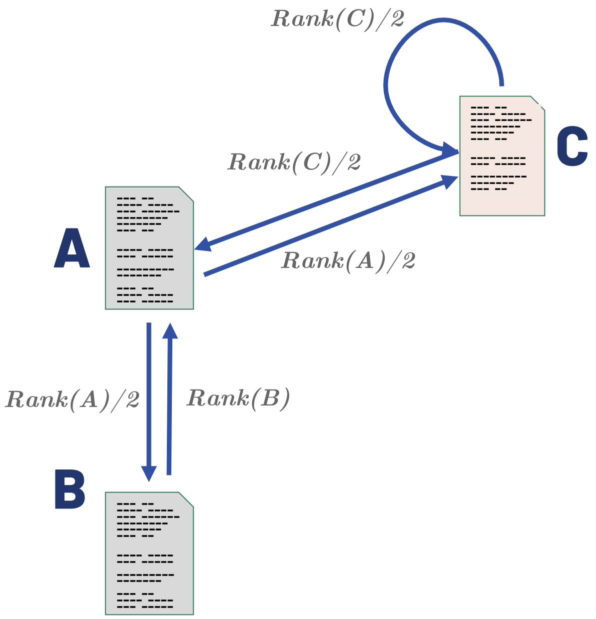
Intuition:

- A "vote" from an important page is worth more
- A page is important if it is pointed to by other important pages

Define a "rank" r_j for page j

$$r_j = \sum_{i \to j} \frac{r_i}{\mathrm{d_i}}$$

d_i ... out-degree of node i



Rank "Flow" equations:

$$\mathbf{r_C} = \mathbf{r_C}/2 + \mathbf{r_A}/2$$
$$\mathbf{r_A} = \mathbf{r_C}/2 + \mathbf{r_B}$$
$$\mathbf{r_B} = \mathbf{r_A}/2$$

To solve these equations, we can set

$$\mathbf{r_C} + \mathbf{r_A} + \mathbf{r_B} = 1$$

and solve analytically to find $\mathbf{r_C} = 2/5 \quad \mathbf{r_A} = 2/5 \quad \mathbf{r_B} = 1/5$

Page Rank: Rearranging the Equation

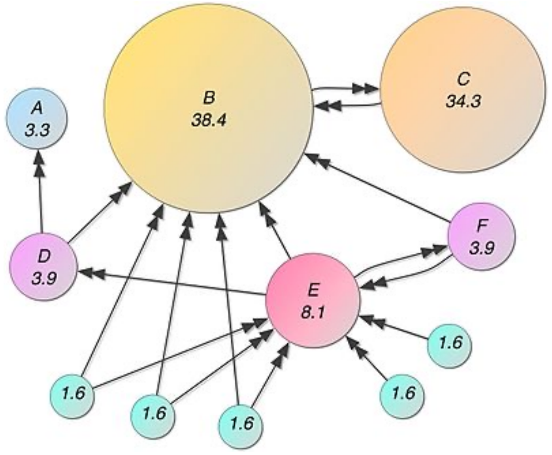
- $\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{r}$ – where $\boldsymbol{A_{ji}} = \boldsymbol{\beta}\, \boldsymbol{M_{ji}} + \frac{\boldsymbol{1-\beta}}{\boldsymbol{N}}$
- $r_j = \sum_{\mathrm{i}=1}^{N} A_{ji} \cdot r_i$
- $r_j = \sum_{i=1}^{N} \left[\beta\, M_{ji} + \frac{1-\beta}{N}\right] \cdot r_i$
- $= \sum_{\mathrm{i}=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N} \sum_{\mathrm{i}=1}^{N} r_i$
- $= \sum_{\mathrm{i}=1}^{N} \beta\, M_{ji} \cdot r_i + \frac{1-\beta}{N}$ since $\sum r_i = 1$
- **So we get:** $\boldsymbol{r} = \boldsymbol{\beta}\, \boldsymbol{M} \cdot \boldsymbol{r} + \left[\frac{\boldsymbol{1-\beta}}{\boldsymbol{N}}\right]_{\boldsymbol{N}}$ *[x]$_N$* … a vector of length *N* with all entries *x*
- **Compare to:** $(1-\alpha)\cdot \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}^T \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_{t_i} + \alpha \cdot \begin{bmatrix} \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_{t_{i+1}}$

Personalized Page Rank: Topic-Specific PageRank

- **Page Rank measures a "generic" popularity of a page, is no specific for a search query or a topic**
- **Instead of generic popularity, can we measure popularity within a topic?**
- **Goal:** Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g., "sports" or "history", defined as a specific **subset of pages**
- **Allows search queries to be answered based on interests of the user**
  - **Example:** A programmer looking for "*library for graph traversal*" wants different pages depending on the programming language they use the most

Assume there is a special subset of pages **S** that we care about

Personalized Page Rank: Topic-Specific PageRank



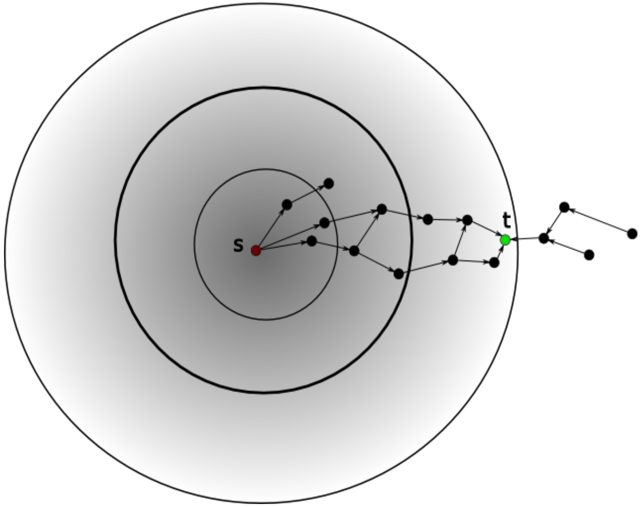
Global Page Rank

Starting from a random node, traversing randomly, **random restart point** anywhere in the graph

The role of the teleport: To avoid dead-end and spider-trap problems

Standard PageRank: Any page with equal probability

Topic Specific PageRank: A topic-specific set of "relevant" pages (teleport set)



Personalized Page Rank

Starting from a **limited set of nodes**, traversing randomly, restart point is one in **the initial set**. <u>Bound not to travel too far</u>

Personalized Page Rank: Topic-Specific PageRank

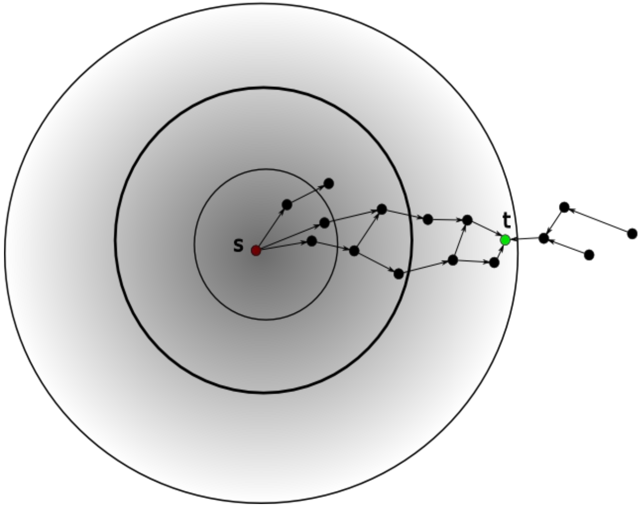
Idea: Bias the random walk

1. When walker teleports, she pick a page from a set ***S***
2. The set ***S*** contains only pages that are relevant to the topic
   E.g., pages with documentation of python libraries
3. For each teleport set ***S***, we get a different vector $\boldsymbol{r_S}$

$$(1-\alpha)\cdot \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}^T \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_{t_i} + \alpha \cdot \begin{bmatrix} 0 \\ \vdots \\ \frac{1}{|S|} \\ 0 \\ \vdots \\ \frac{1}{|S|} \\ \vdots \\ 0 \end{bmatrix}$$

← **We change *the teleport Vector!***

When we teleport back to a single node is called: Random Walk with Restart



Personalized Page Rank

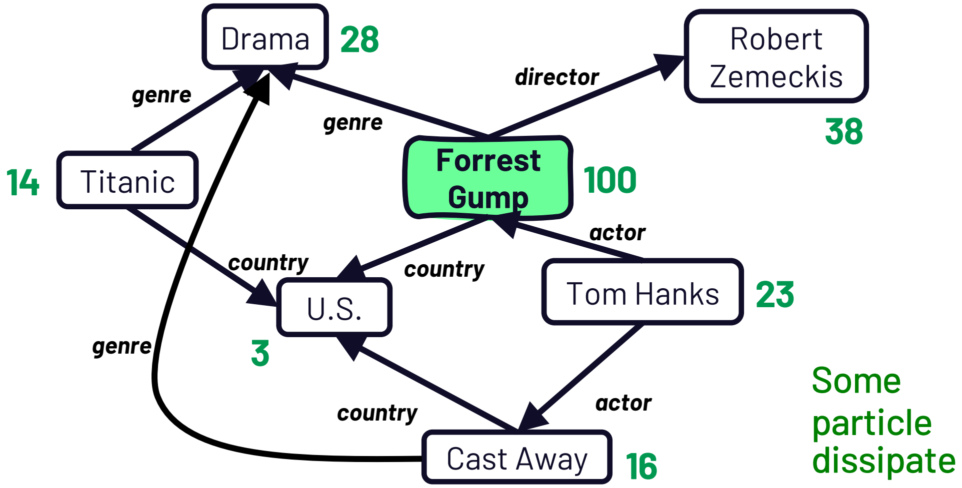
Starting from a **limited set of nodes**, traversing randomly, restart point is one in **the initial set**. <u>Bound not to travel too far</u>

Particle Filtering Approach

Speed up PPR computation

Simulate a set of particles navigating the graphs

Particle spread not-uniformly following edge importance



Edge Importance: *Director > Genre > Actor > Country*

Edge weights outgoing each node should sum to 1 !

Particles start from the query nodes

Edges are traversed based **on priority**

Particles are **split non-uniformly + dissipation**

Require: Graph G; Query nodes $\mathbf{Q}$
Require: Restart probability $c \in [0,1]$; Threshold $\tau \in [0,1]$
Require: Query value k
Ensure: Ranked Top-K nodes

$\mathbf{p} \leftarrow \{\}$
for each $q_i \in \mathbf{Q}$ **do**
  $\mathbf{p}[q_i] \leftarrow 1/\tau$ ▷ Initialize Particles
while $\exists\, n_i \in \mathbf{p} \mid \mathbf{p}[n_i] \neq 0$ **do**
  $\mathbf{temp} \leftarrow \{\}$
  for each $n_i \in \mathbf{p} \mid \mathbf{p}[n_i] \neq 0$ **do**
    $particles \leftarrow \mathbf{p}[n_i] \times (1-c)$
    for each $e : (n_i \to n_j) \in G$ **do** ▷ Sorted by Weight
      if $particles \leq \tau$ **then**
        break
      $passing \leftarrow$ MAX($particles \times e$.weight(),τ)
      $\mathbf{temp}[n_j] \leftarrow \mathbf{temp}[n_j] + passing$
      $particles \leftarrow particles - passing$
  $\mathbf{p} \leftarrow \mathbf{temp}$
  for each $n_i \in \mathbf{p}$ **do**
    $\mathbf{v}[n_i] \leftarrow \mathbf{v}[n_i] + \mathbf{p}[n_i] \times c$ ▷ Update score
return top-k($\mathbf{v}$)

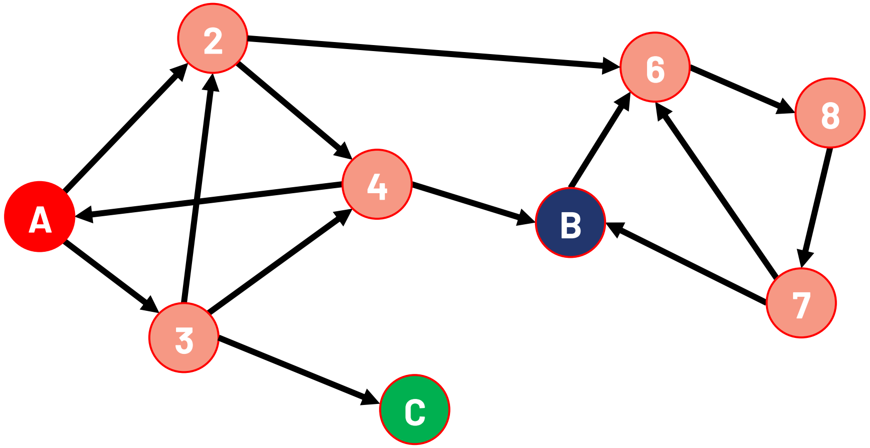
Personalized page rank on knowledge graphs: Particle Filtering is all you need!
Gallo, D., Lissandrini, M., Velegrakis, Y. (*EDBT 2020*)

Personalized Page Rank as a Proximity Measure

What is the probability to reach node B *given that we start from node A? Compared to C?*

What are the most “relevant” nodes for A ranked by “closeness”

a.k.a.: Relevance, ‘Relatedness’...



- **Multiple connections**
- **Quality of connection**
    - **Direct & Indirect connections**
    - **Length & ”quantity”**

Another Measure: **hitting time**

$\mathbf{h(A\rightarrow\ B)}$ is the average number of steps to walk from node $\mathbf{A}$ to node $\mathbf{B.}$

Hitting time is asymmetric $\mathbf{h(A\rightarrow\ B)}$ is not always the same as $\mathbf{h(A\rightarrow\ B)}$

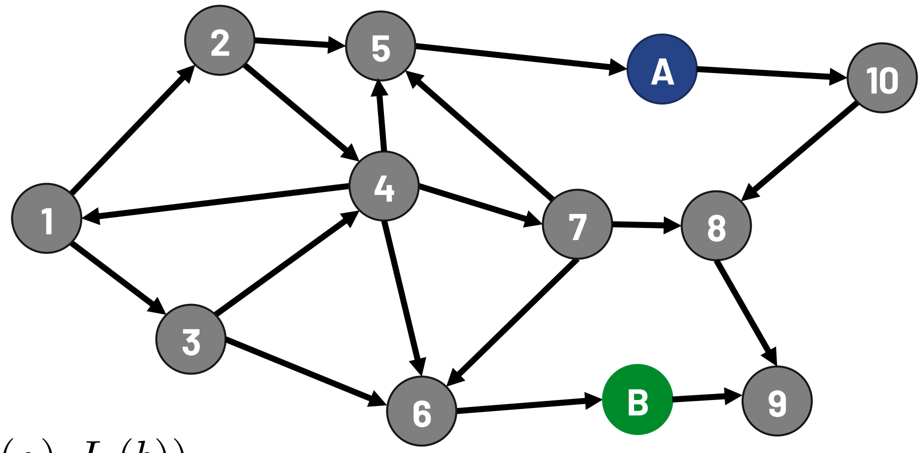
http://www.cs.cornell.edu/courses/cs4850/2009sp/Scribe%20Notes/Lecture%2024%20Friday%20March%2013.pdf

SimRank: A recursive definition of similarity

Measure the similarity of two objects:

Intuition: *Two objects are similar if they are related to similar objects*

A recursive definition of similarity based on graph structure:



$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s\left(I_i(a), I_j(b)\right)$$

$I(a)$ ← *Incoming nodes to a* $I_i(a)$ ← the i-th *incoming node of a*

Where C is a constant between 0 and 1 When $I(a) = \emptyset$ or $I(b) = \emptyset$ then $s(a,b) = 0$

How similar is the "role" of A and B in this graph?

Further References

Davis Shurbert, An Introduction to Graph Homomorphisms
http://buzzard.ups.edu/courses/2013spring/projects/davis-homomorphism-ups-434-2013.pdf

Andreas Schmidt, Iztok Savnik, Conference on Advances in Databases, Knowledge, and Data Applications, Overview of Regular Path Queries in Graphs
https://www.iaria.org/conferences2015/filesDBKDA15/graphsm_overview_of_regular_path_queries_in_graphs.pdf

Jure Leskovec, CS224W: Machine Learning with Graphs | 2019 |
Lecture 3-Motifs and Structural roles in networks
http://snap.stanford.edu/class/cs224w-2019/slides/03-motifs.pdf

Davide Mottin and Emmanuel Müller, Graph Exploration:
Let me Show what is relevant in your graph, KDD tutorial
https://mott.in/slides/KDD2018-Tutorial-Compressed.pdf

Kolaczyk, E.D., Statistical Analysis of Network Data, Chapter 5: Sampling and Estimation in Network Graphs.
https://link.springer.com/chapter/10.1007/978-0-387-88146-1_5

Jure Leskovec, Christos Faloutsos, Sampling from Large Graphs
https://dl.acm.org/doi/pdf/10.1145/1150402.1150479