Graph Databases and Complex Networks

Luiz Celso Gomes Jr - André Santanchè
MC536 2013/2

Outline

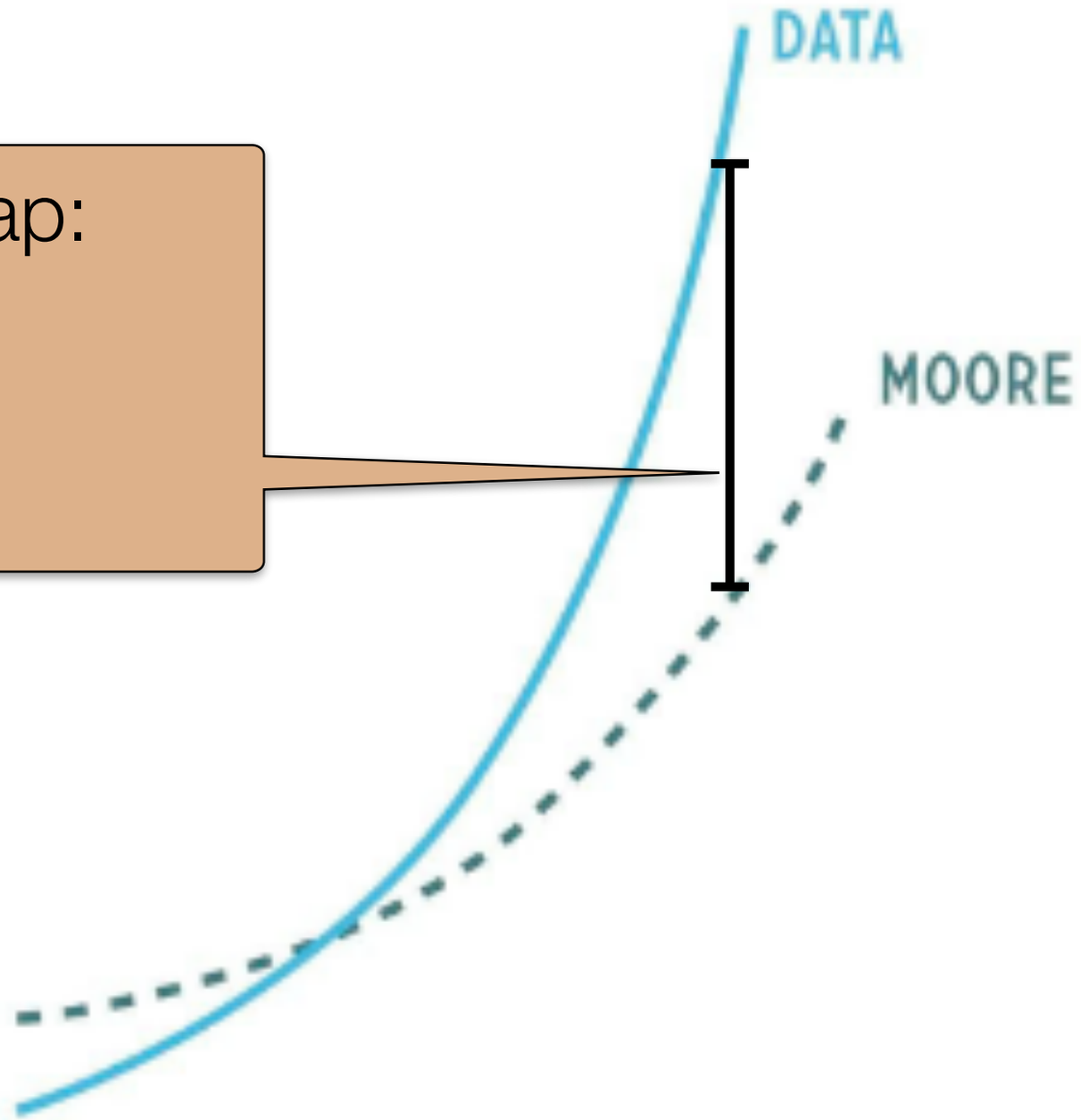
- Big Data
- Graph Databases
- Neo4j
- Graph querying (Cypher, Gremlin)
- Complex Networks

small computers

~~Big~~ Data vs Smart programmers

bridging the gap:

- distribution
- architecture
- **data model**



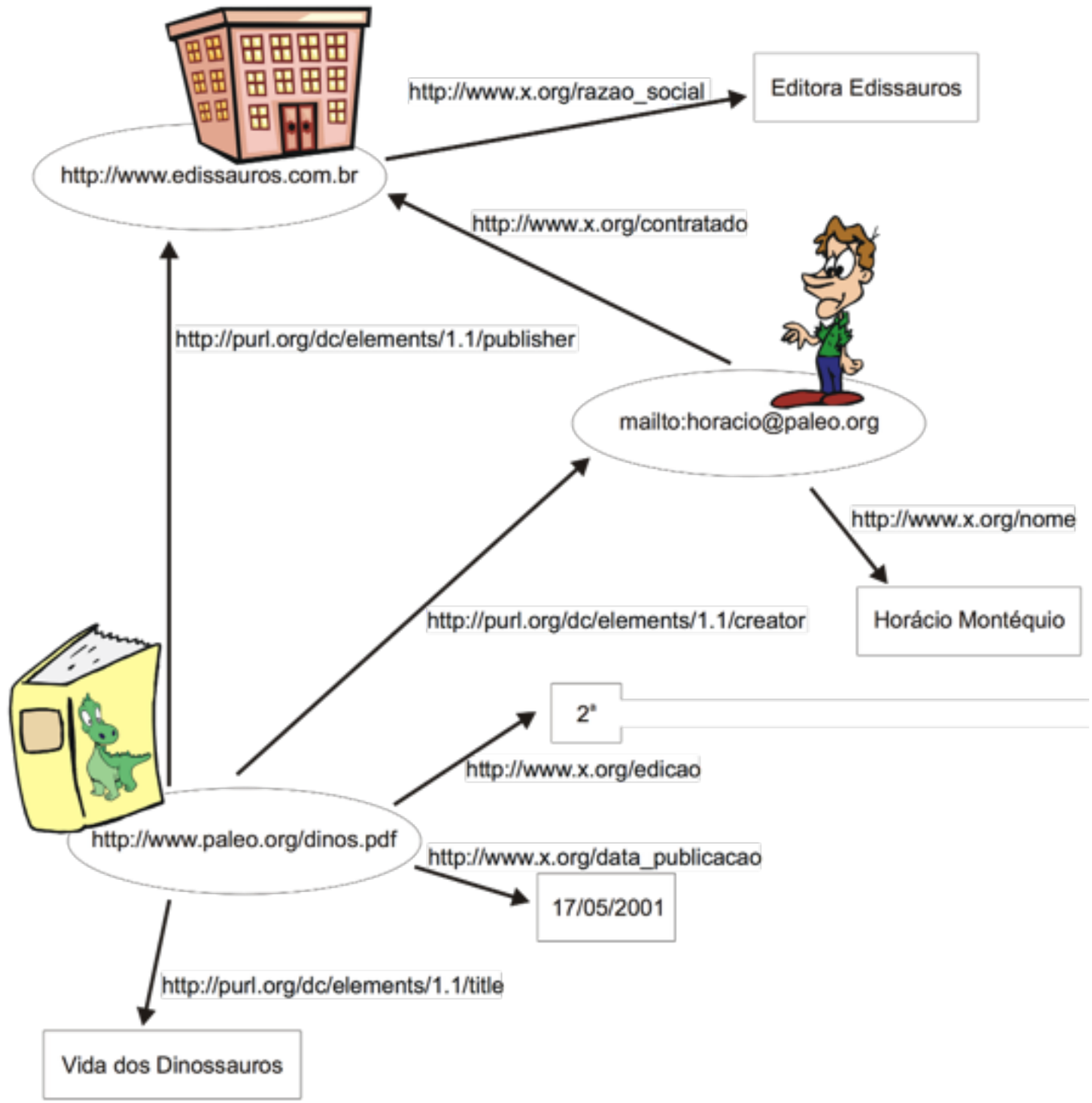
NoSQL Databases

NoSQL database classes

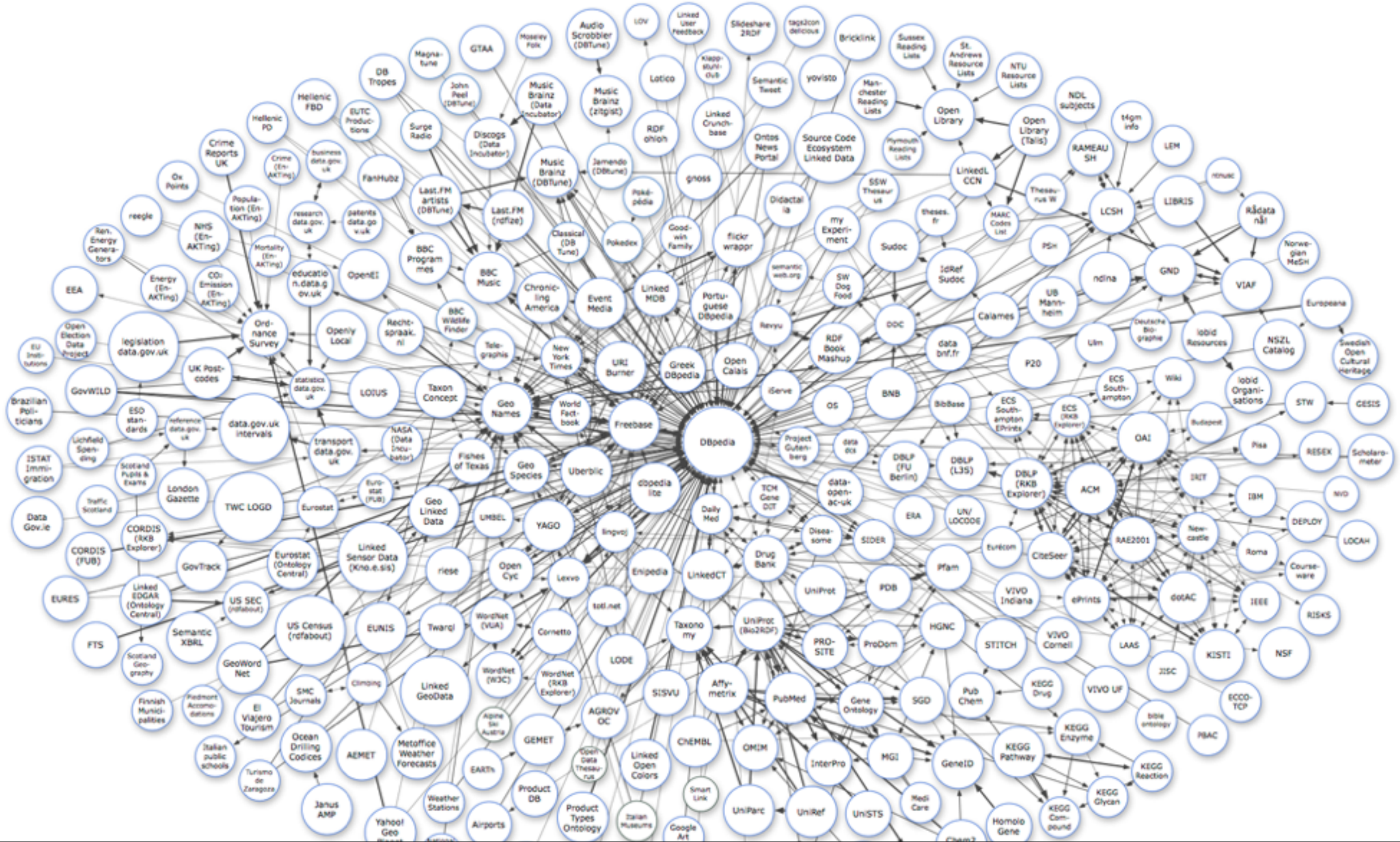
- Key-value stores
- Document-oriented databases
- **Graph databases**

Graph Data - Social Networks

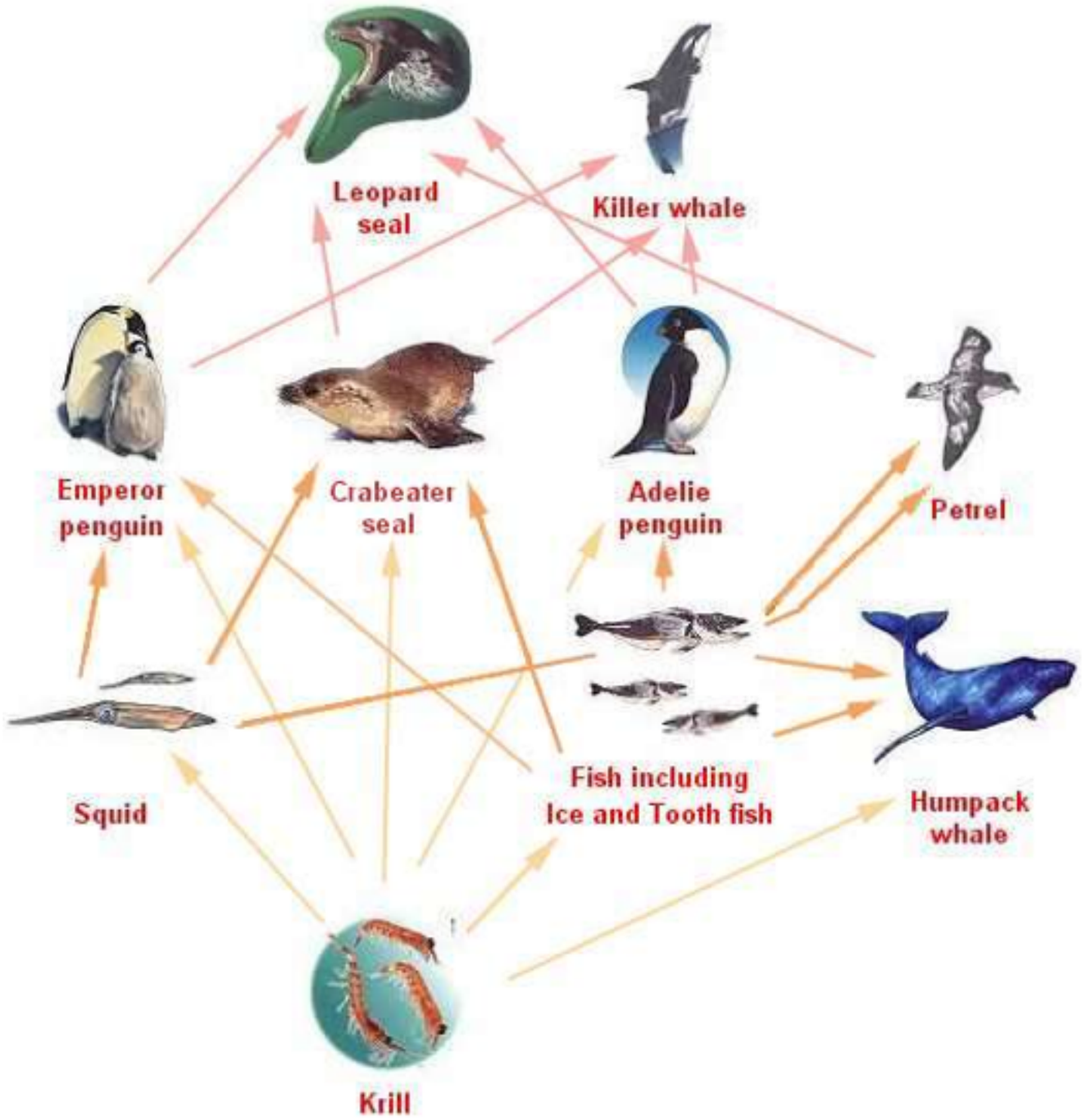
Graph Data - Ontologies



Graph Data - Linked Data

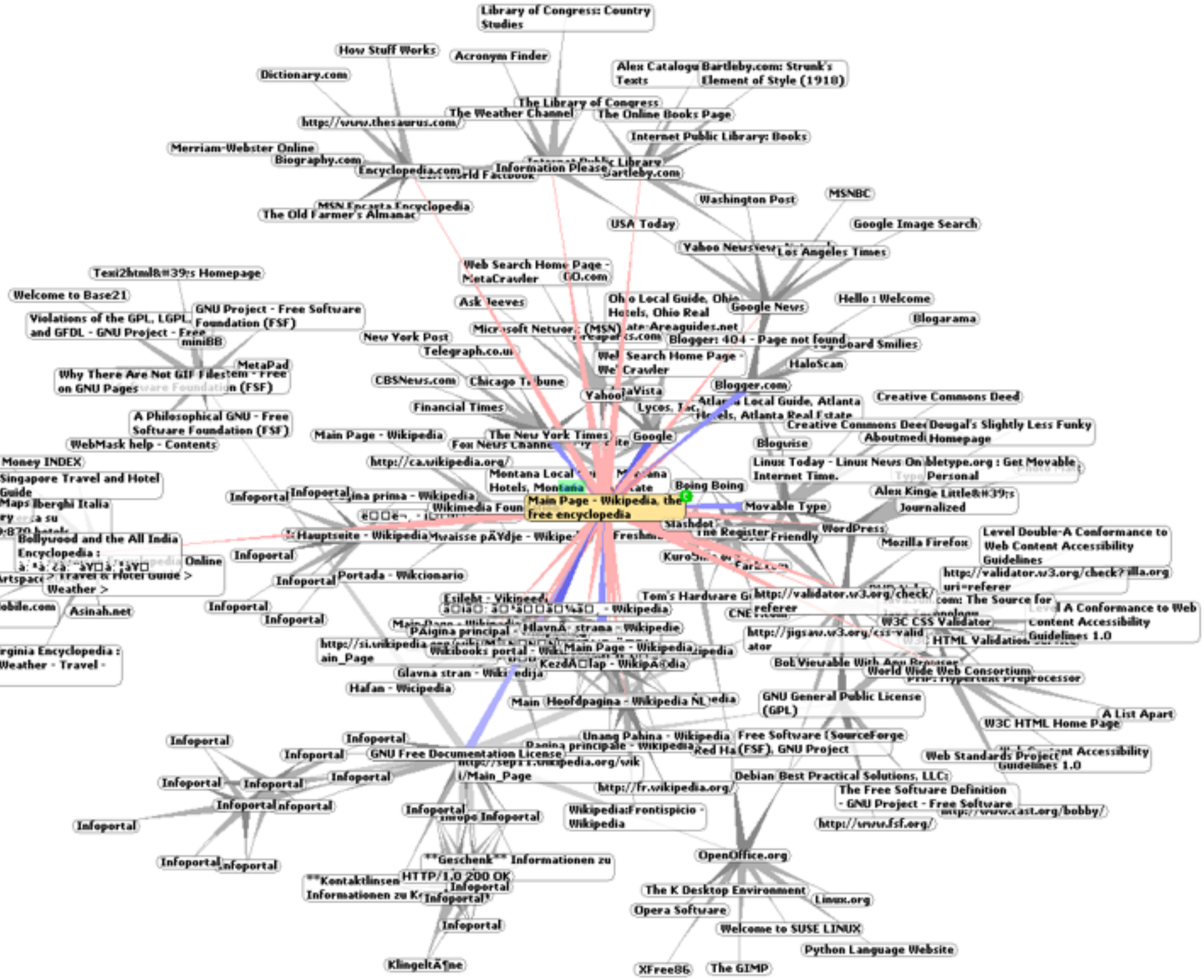


Graph Data - Biological Food Webs

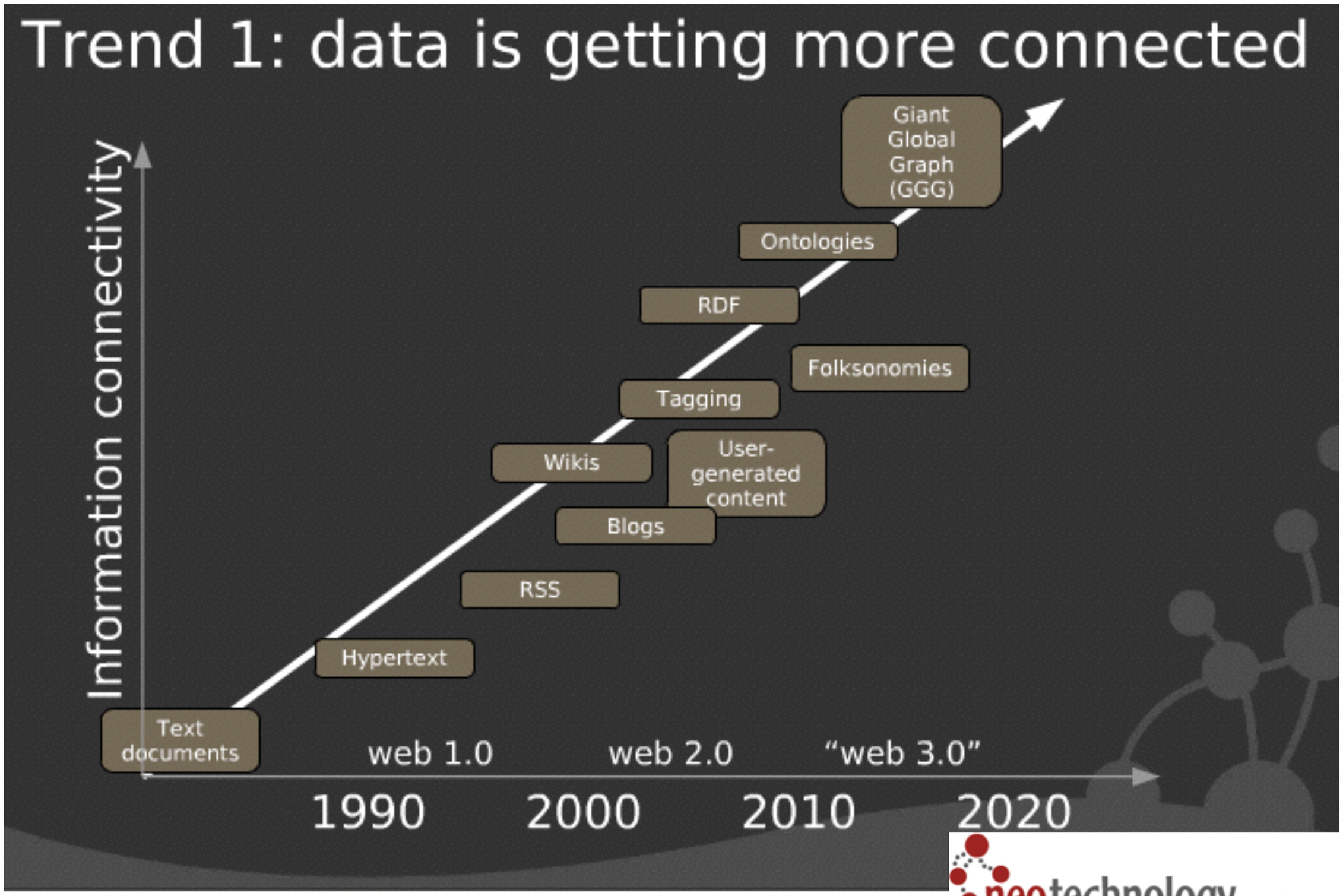


Graph Data - Scientific Citation Networks

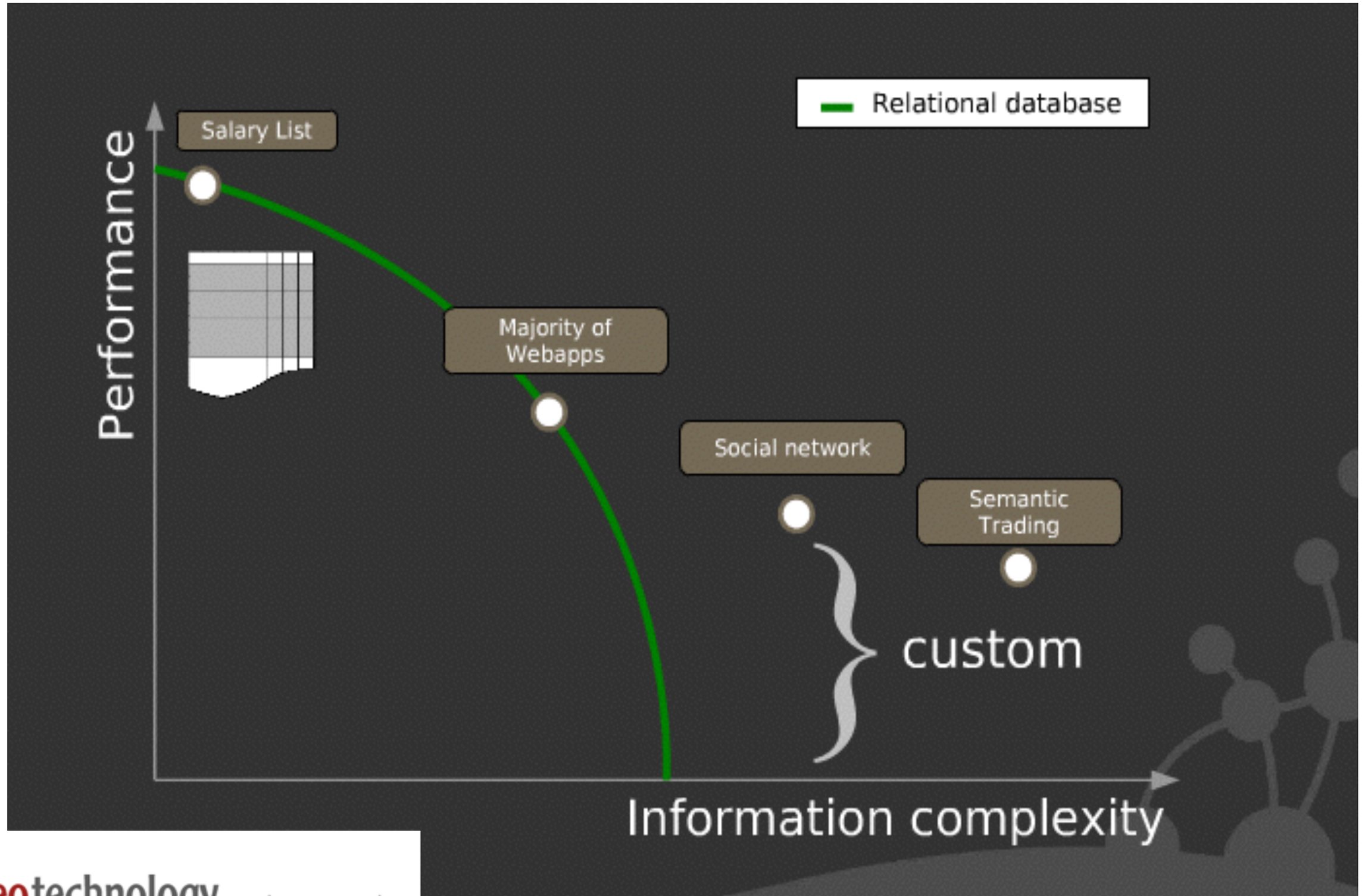
Graph Data - WWW



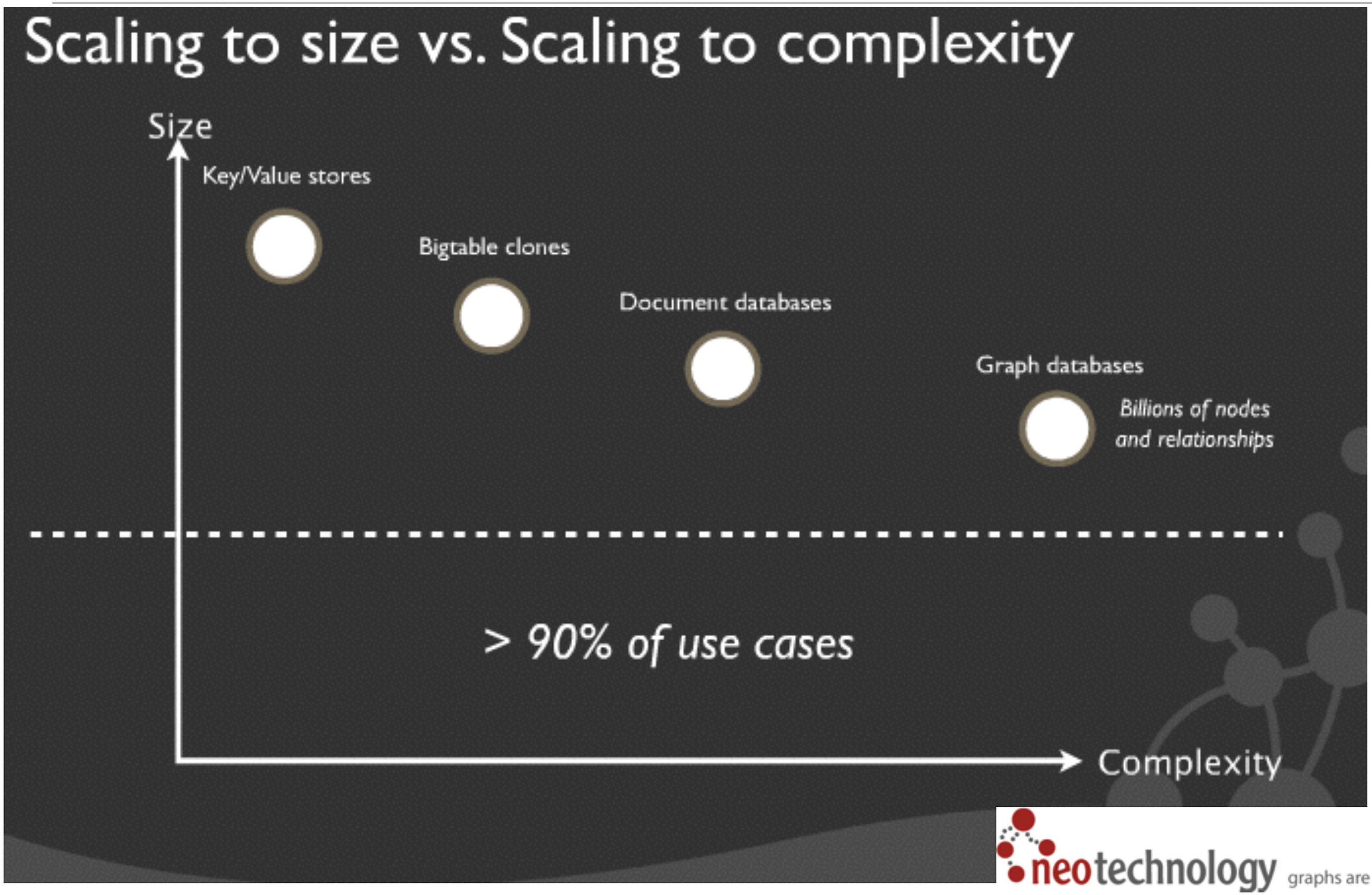
Big/Connected Data



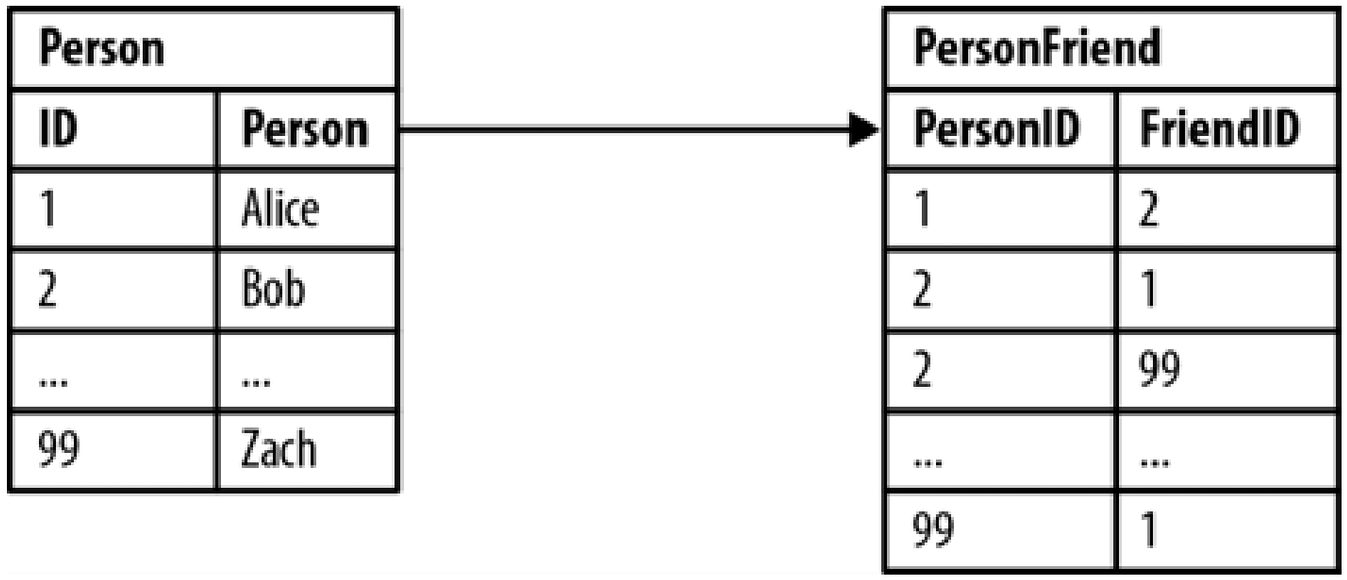
Performance limits of RDBMSs



Scaling Complexity



Relational DBs for connected data



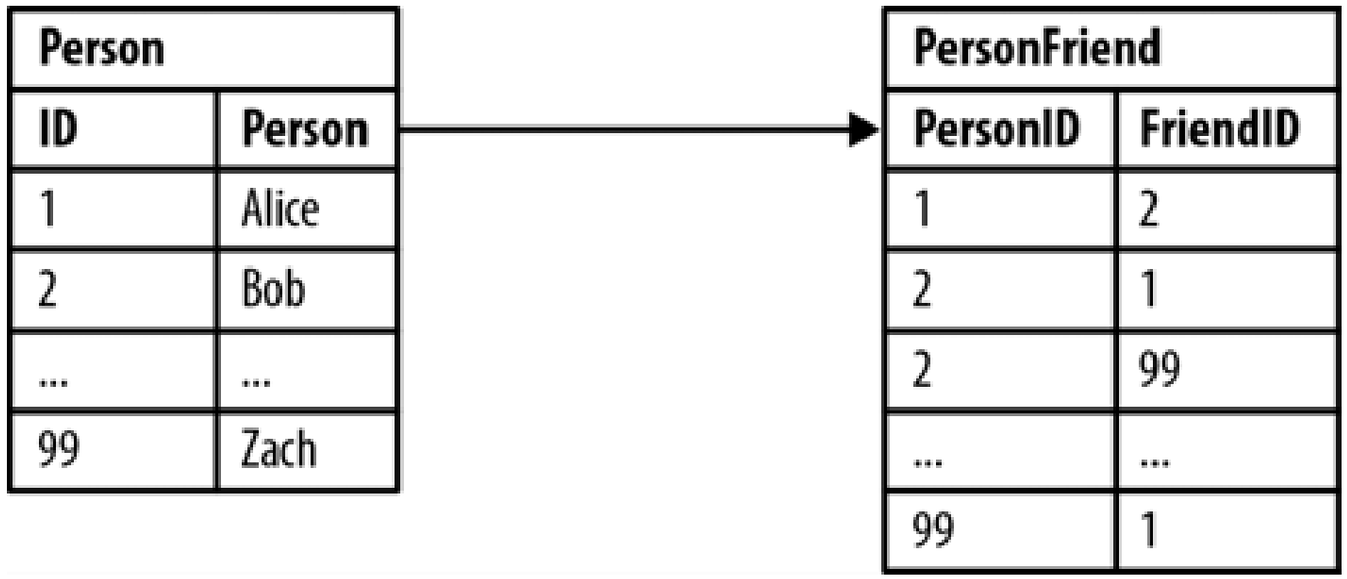
Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

Example 2-1. Bob's friends

```
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
  ON PersonFriend.FriendID = p1.ID
JOIN Person p2
  ON PersonFriend.PersonID = p2.ID
WHERE p2.Person = 'Bob'
```

Relational DBs for connected data



Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

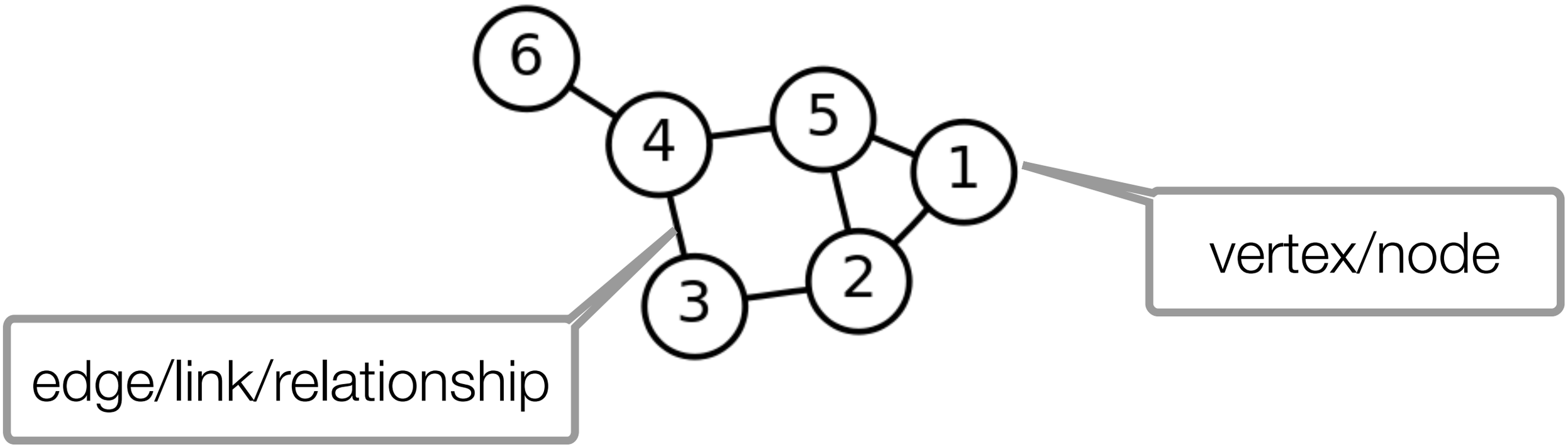
PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

Example 2-3. Alice's friends-of-friends

```
SELECT p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
FROM PersonFriend pf1 JOIN Person p1
  ON pf1.PersonID = p1.ID
JOIN PersonFriend pf2
  ON pf2.PersonID = pf1.FriendID
JOIN Person p2
  ON pf2.FriendID = p2.ID
WHERE p1.Person = 'Alice' AND pf2.FriendID <> p1.ID
```

Graphs

a more natural model for connected data



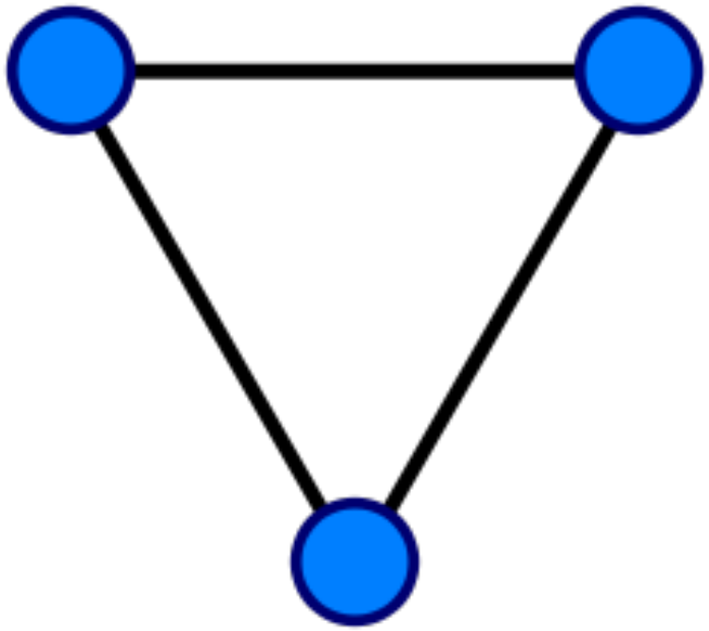
G = (V, E)

G -> graph

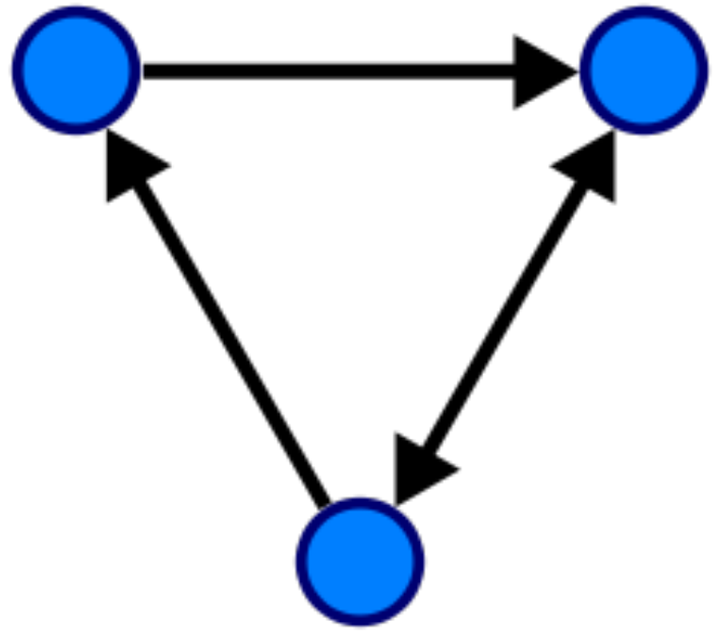
V -> set of vertices or nodes

E -> set of edges or lines

Graphs



Undirected



Directed

Graph Databases

- provide index-free adjacency (definition)
- faster for associative data sets (ontologies, social networks, computer networks, circuits, traffic systems, file systems...)
- allow fast implementations of important graph algorithms (shortest path, path exists, degree of separation...)
- other models can be conveniently represented
- hard to distribute

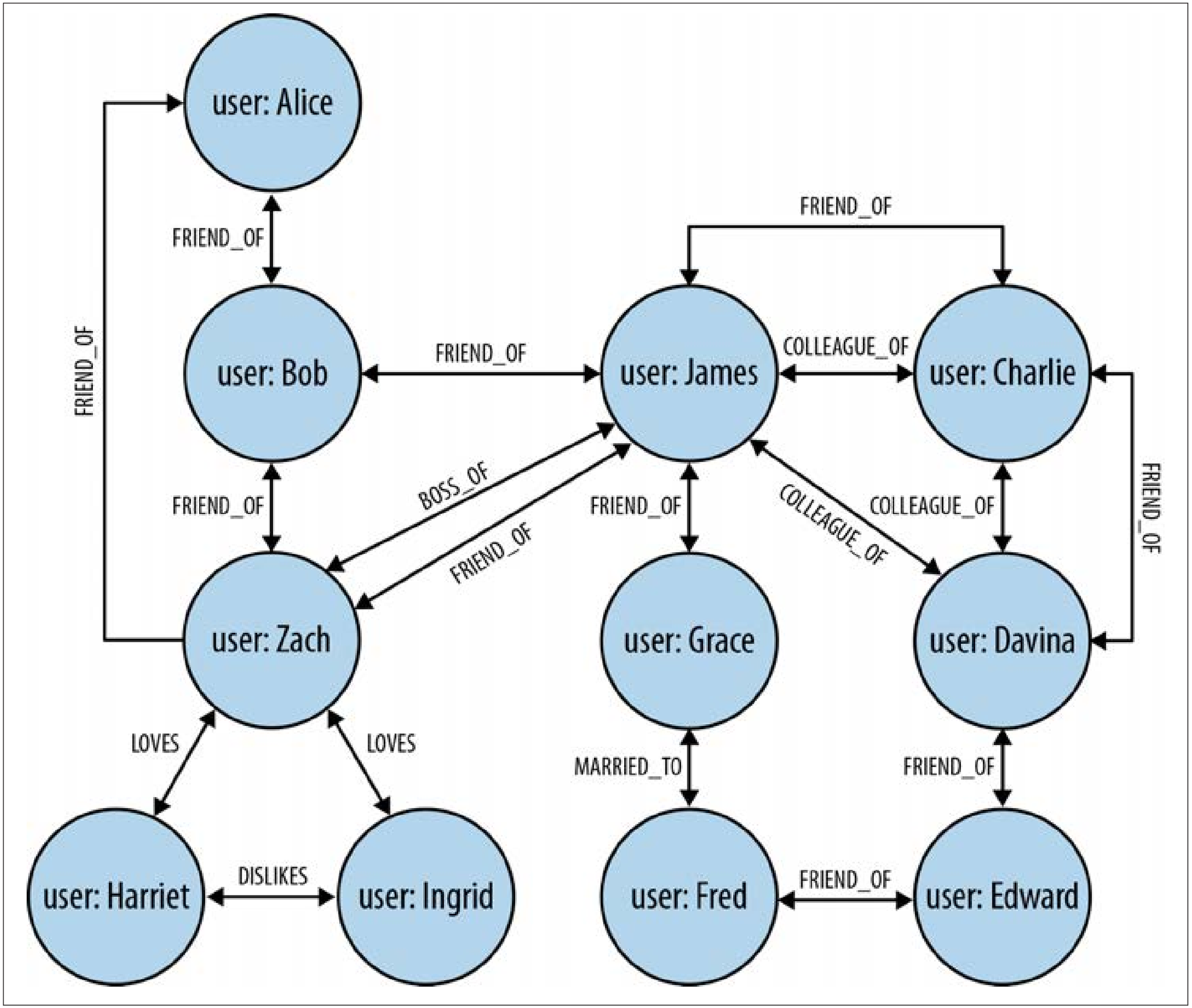
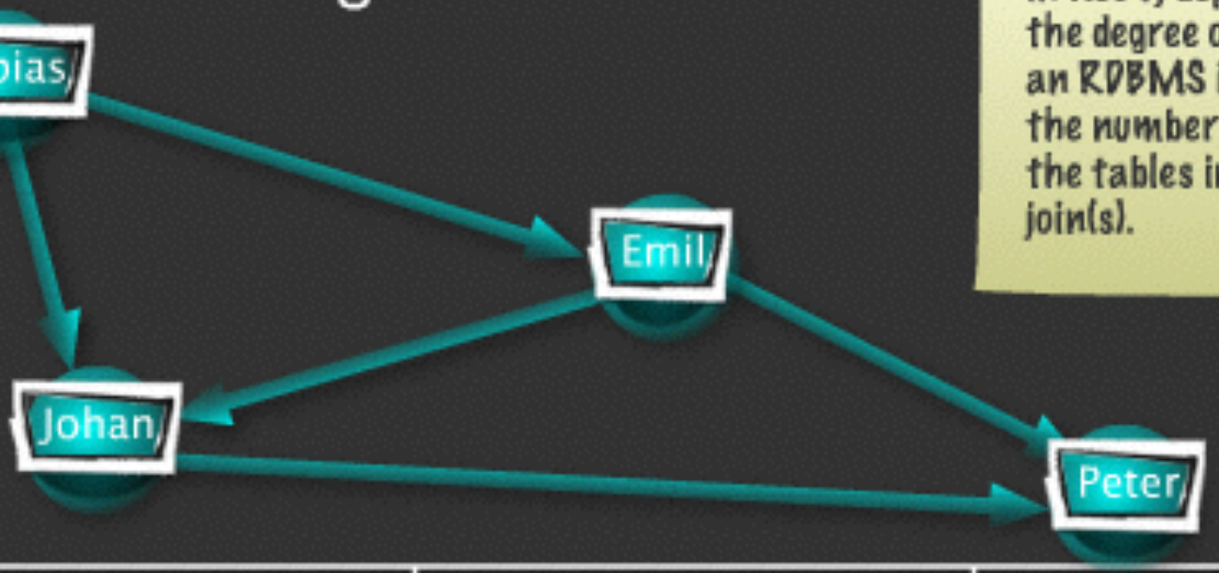


Figure 2-5. Easily modeling friends, colleagues, workers, and (unrequited) lovers in a graph

Performance comparison

Path exists in social network

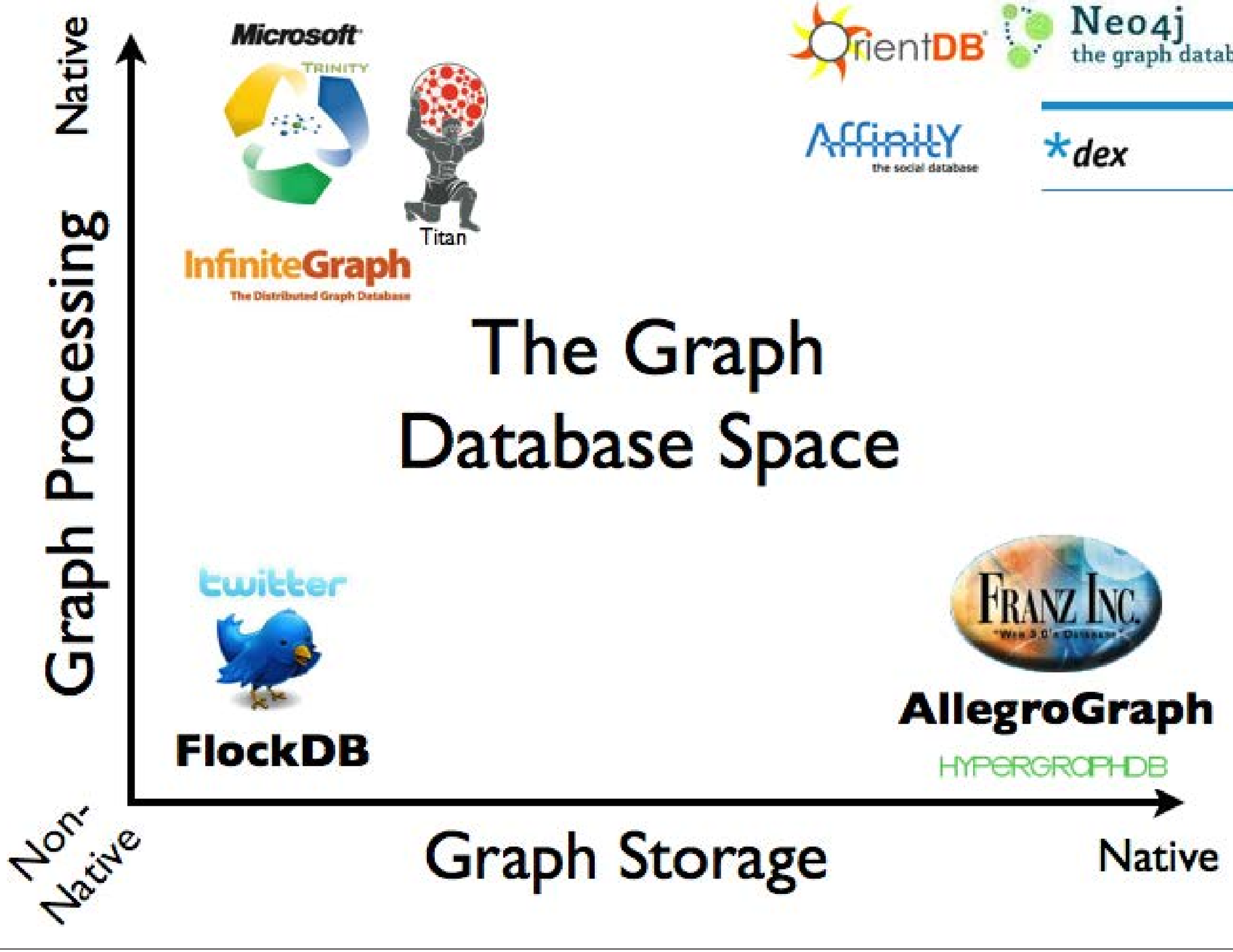
- Each person has on average 50 friends



The performance impact in Neo4j depends only on the degree of each node. in an RDBMS it depends on the number of entries in the tables involved in the join(s).

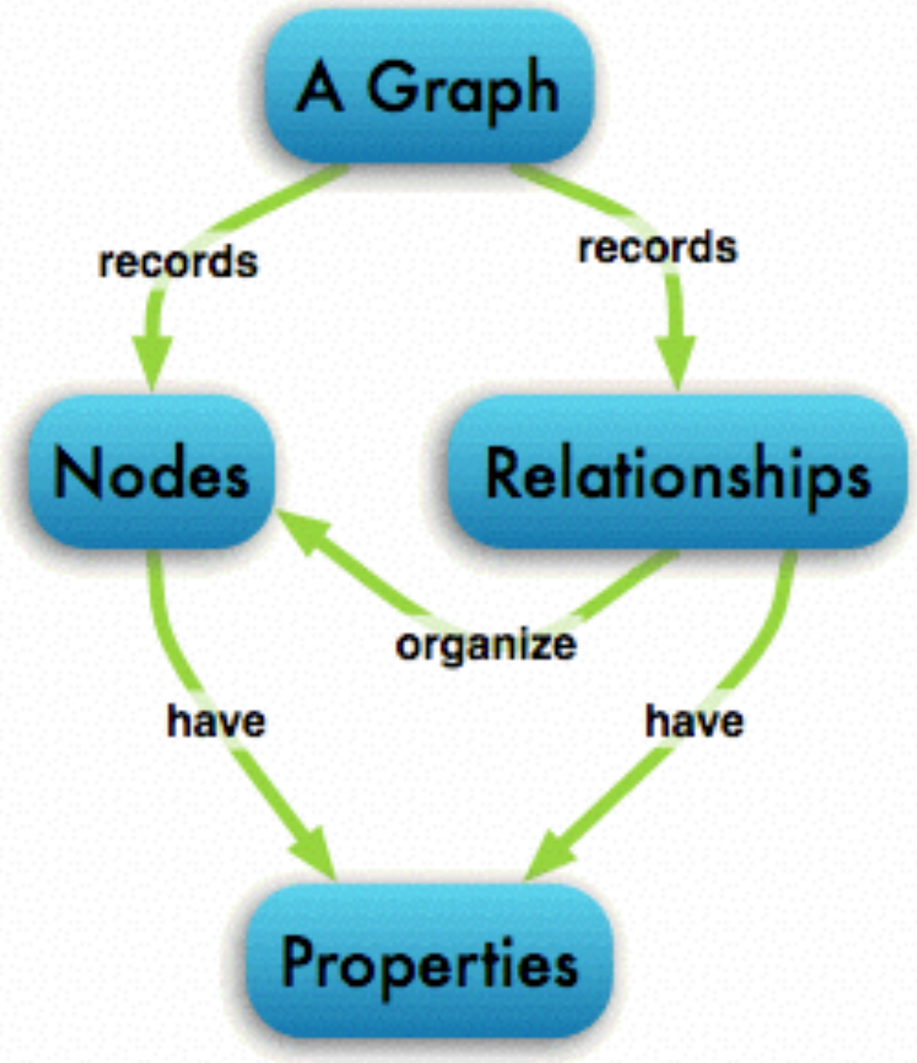
Database	# persons	query time
Relational database	1 000	2 000 ms
Neo4j Graph Database	1 000	2 ms
Neo4j Graph Database	1 000 000	2 ms
Relational database	1 000 000	way too long...

The Graph Database Space



Case - Neo4j

- most popular graph db
- true ACID transactions
- high availability
- scales to billions of nodes and relationships
- high speed querying through traversals
- simple, intuitive data model
- declarative DDL and DML



The Property Graph Model

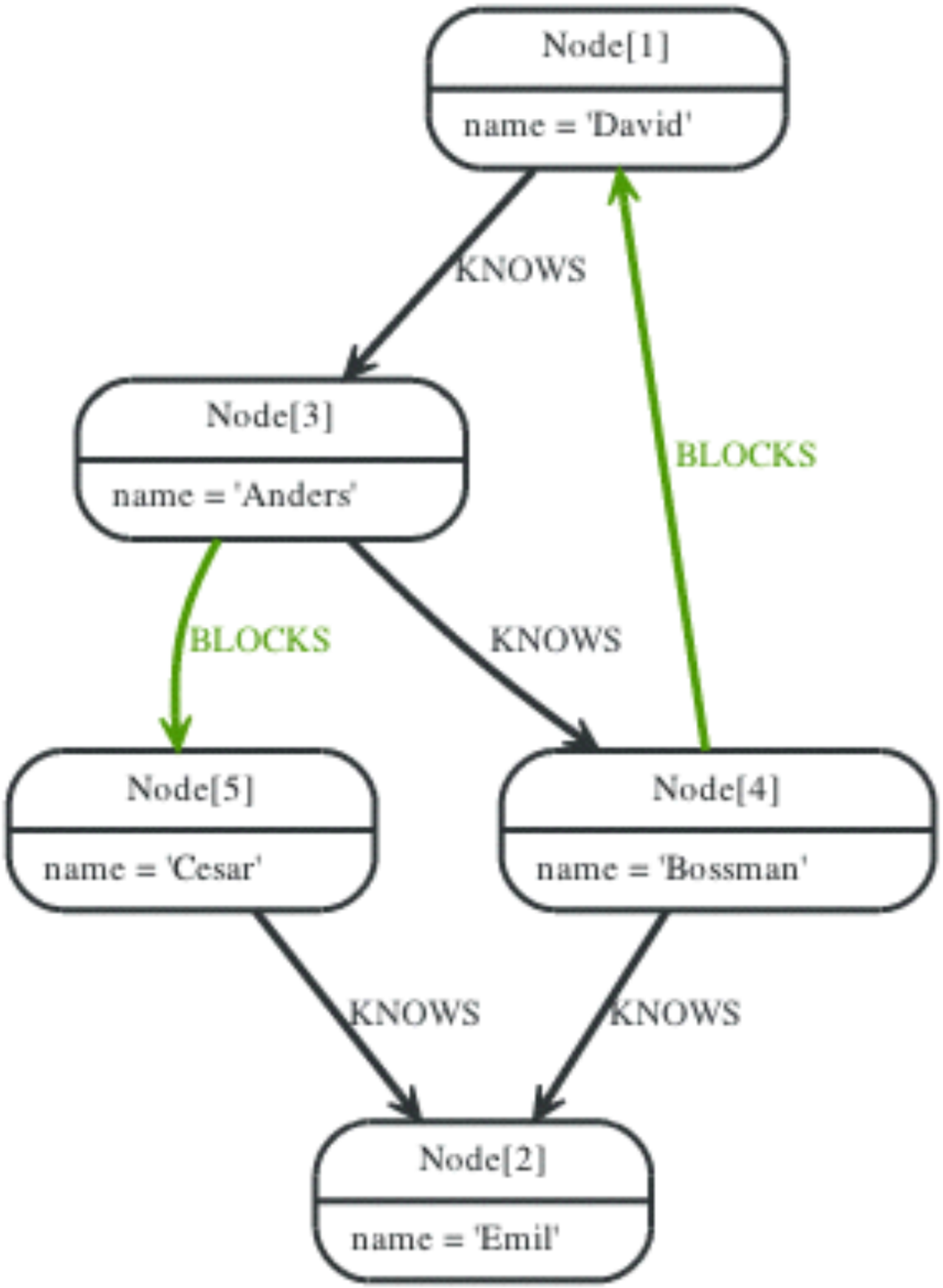
- A property graph is made up of nodes, relationships, and properties.

- Nodes contain properties in the form of arbitrary key-value pairs. The keys are strings and the values are arbitrary data types.

- Relationships connect and structure nodes. A relationship always has a direction, a label, and a start node and an end node.

- Like nodes, relationships can also have properties.

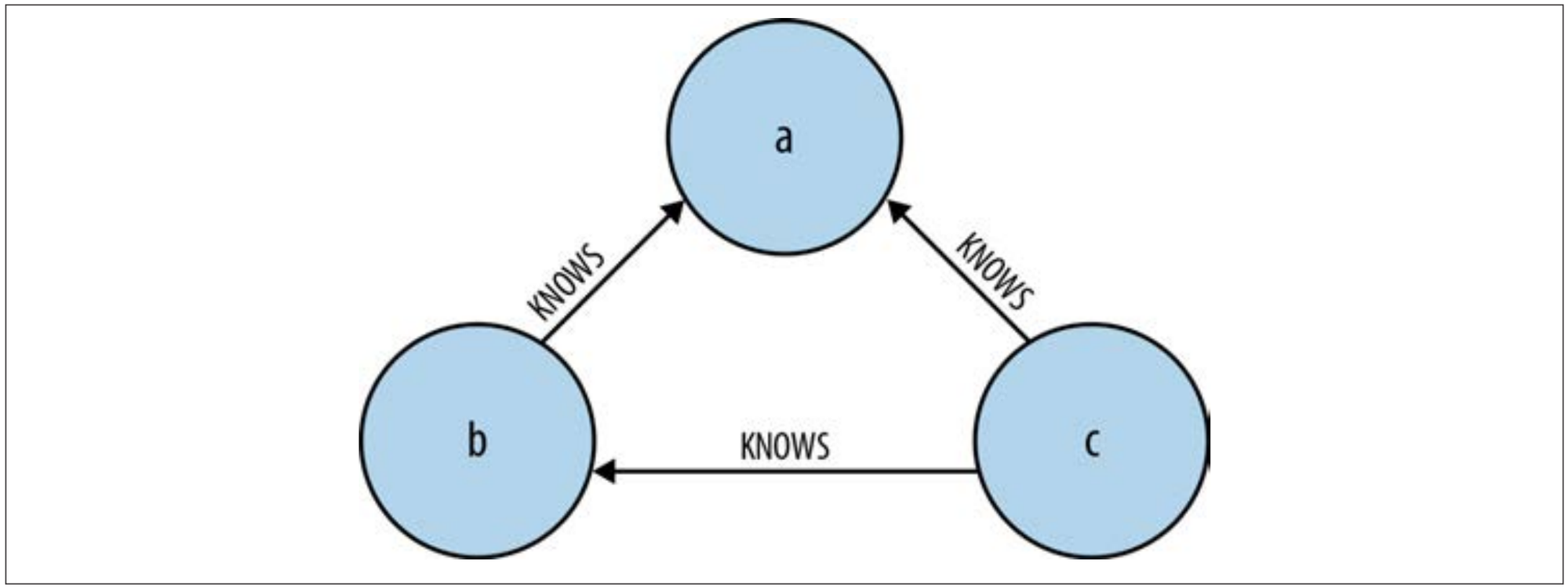
The Property Graph Model



Querying graph databases

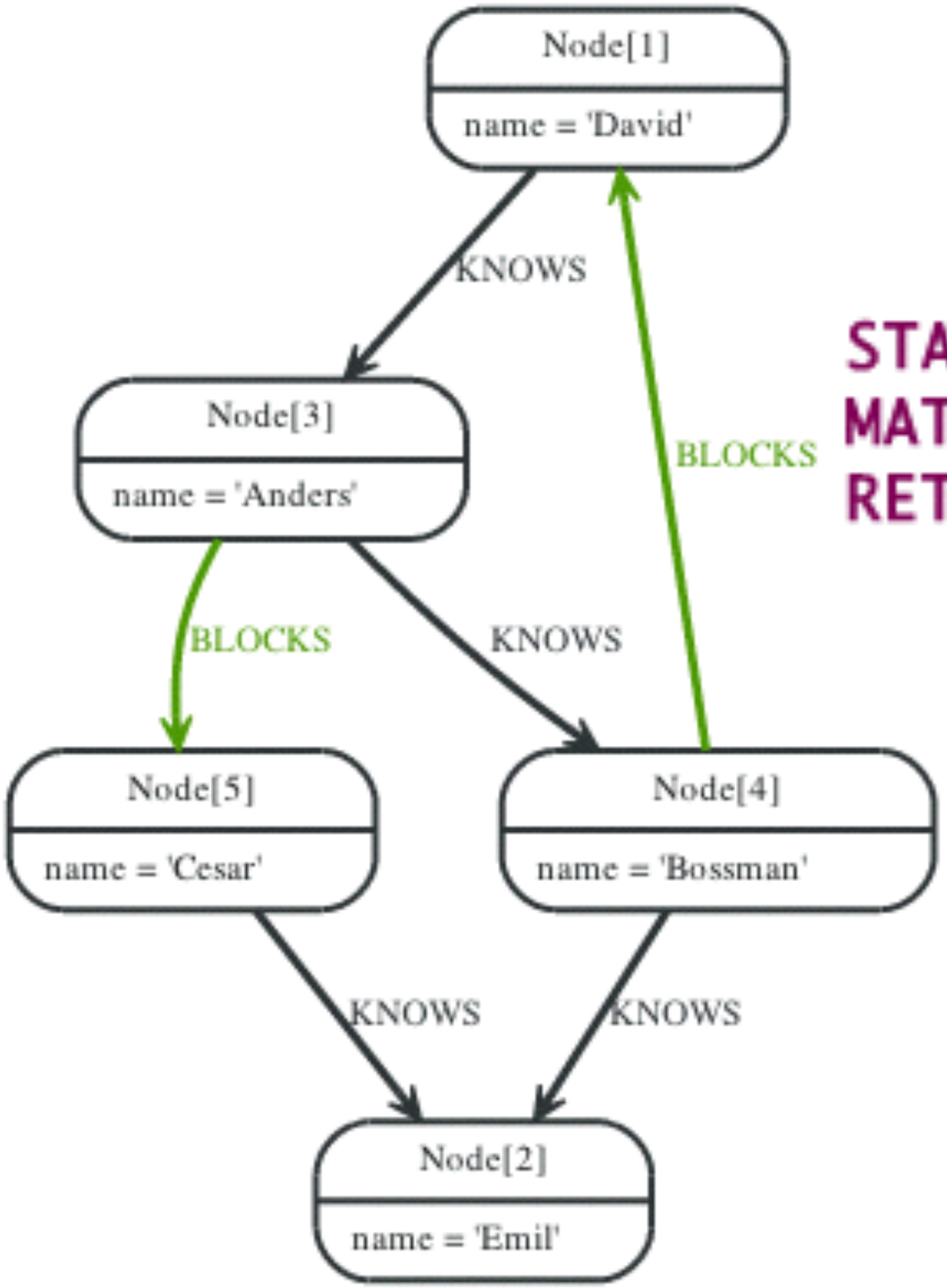
- Declarative languages
  - Graph patterns - Conjunctive Regular Path Queries
  - Examples: SPARQL, Cypher, GraphLog…
- Iterative languages
  - Graph traversal patterns
  - Examples: Gremlin, diverse APIs

Cypher Queries - Graph Patterns



(a)-[:KNOWS]->(b)-[:KNOWS]->(c), (a)-[:KNOWS]->(c)

Cypher examples




```
START me=node(1)
MATCH me-->friend-[?:parent_of]->children
RETURN friend, children
```

friend	children
Node[3]{name->"Anders"}	<null>
1 row, 1 ms	

Cypher examples

```
START a=node(3)
MATCH (a)-[:KNOWS]->(b)-[:KNOWS]->(c)
RETURN a,b,c
```

The three nodes in the path.

Result

a	b	c
Node[3]{name->"Anders"}	Node[4]{name->"Bossman"}	Node[2]{name->"Emil"}
1 row, 0 ms		

Cypher examples

```
START a=node(3), x=node(2, 4)
MATCH a-[:KNOWS*1..3]->x
RETURN a,x
```

Returns the start and end point, if there is a path between 1

Result

a	x
Node[3]{name->"Anders"}	Node[2]{name->"Emil"}
Node[3]{name->"Anders"}	Node[4]{name->"Bossman"}
2 rows, 1 ms	

Gremlin

g.V('name','hercules').out('father').out('father').name

- Traversal patterns

- Traverse the graph and returns data according to the path in the query

- Loops, aggregations, and side-effects possible

Gremlin Graph Traversals

Exploring Wikipedia with Gremlin Graph Traversals

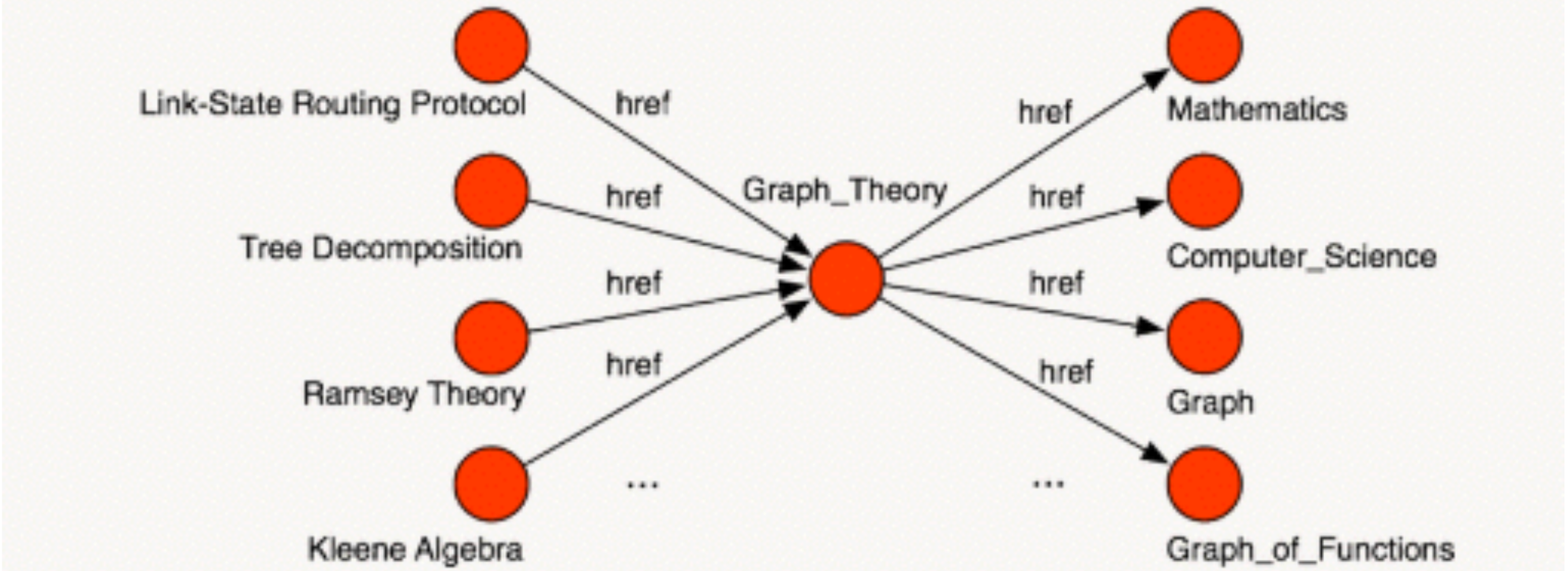
http://markorodriguez.com/2012/03/07/exploring-wikipedia-with-gremlin-graph-traversals/

Opening the graph

```
          \,,,/
          (o o)
-----oOOo-(_)-oOOo-----
gremlin> g = new Neo4jGraph('/data/dbpedia')
==>neo4jgraph[/data/dbpedia]
gremlin> g.V.count()
==>30962172
gremlin> g.E.count()
==>191767951
```

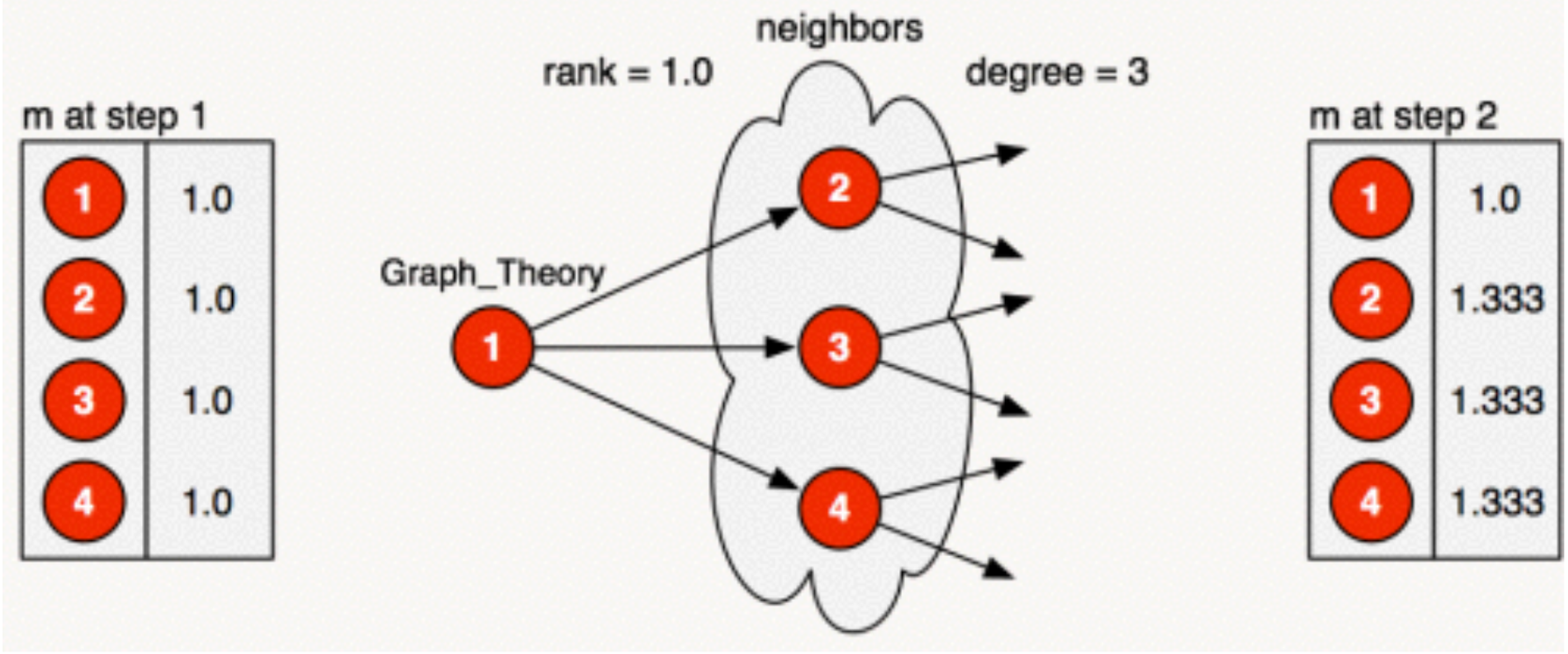
Exploring neighbors

```
gremlin> v.in('href').name[0..4]
==>semiotics of the structure
==>sones graphdb
==>trapezoid graph
==>alpha centrality
==>block graph
```

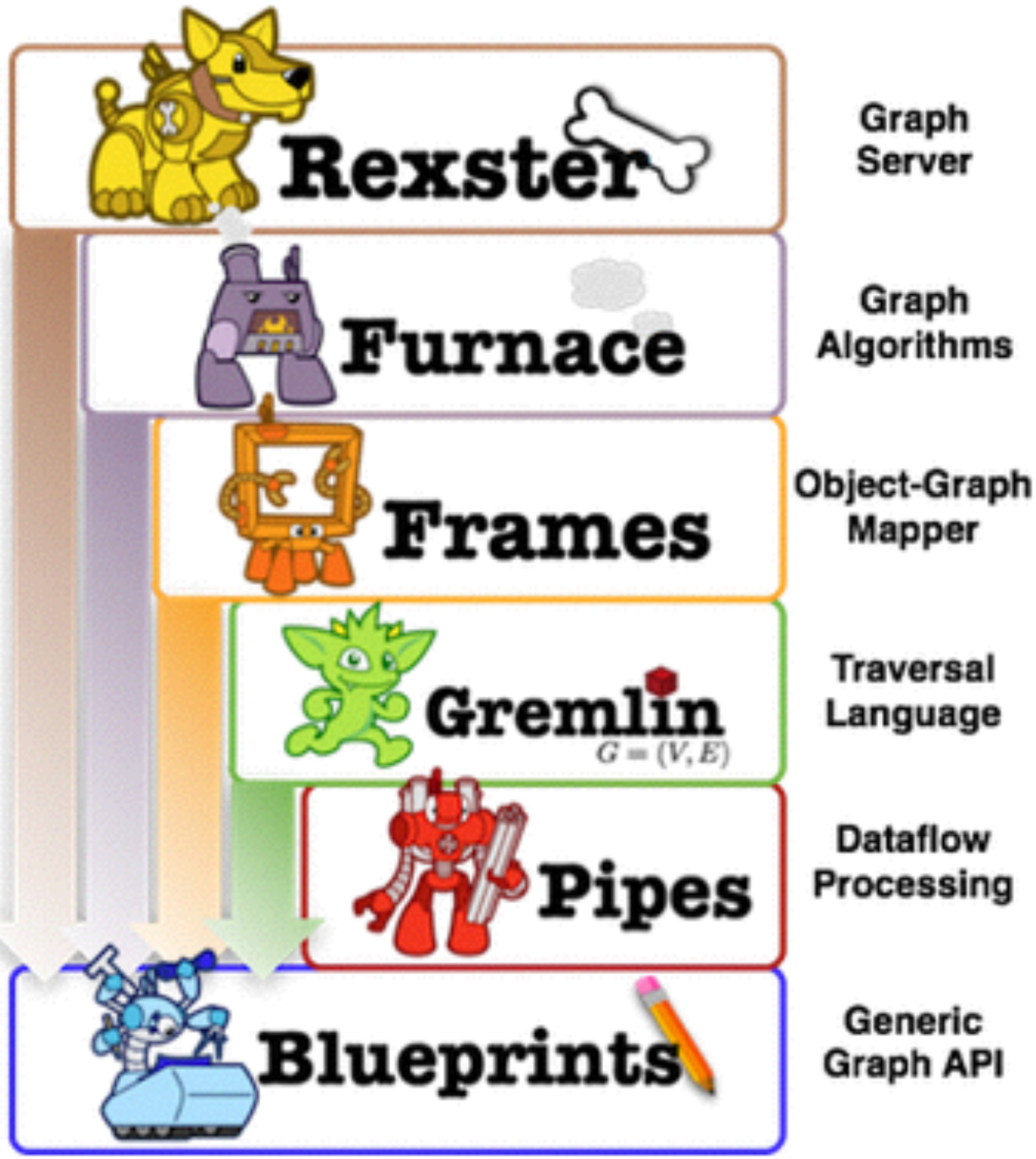


Random walk

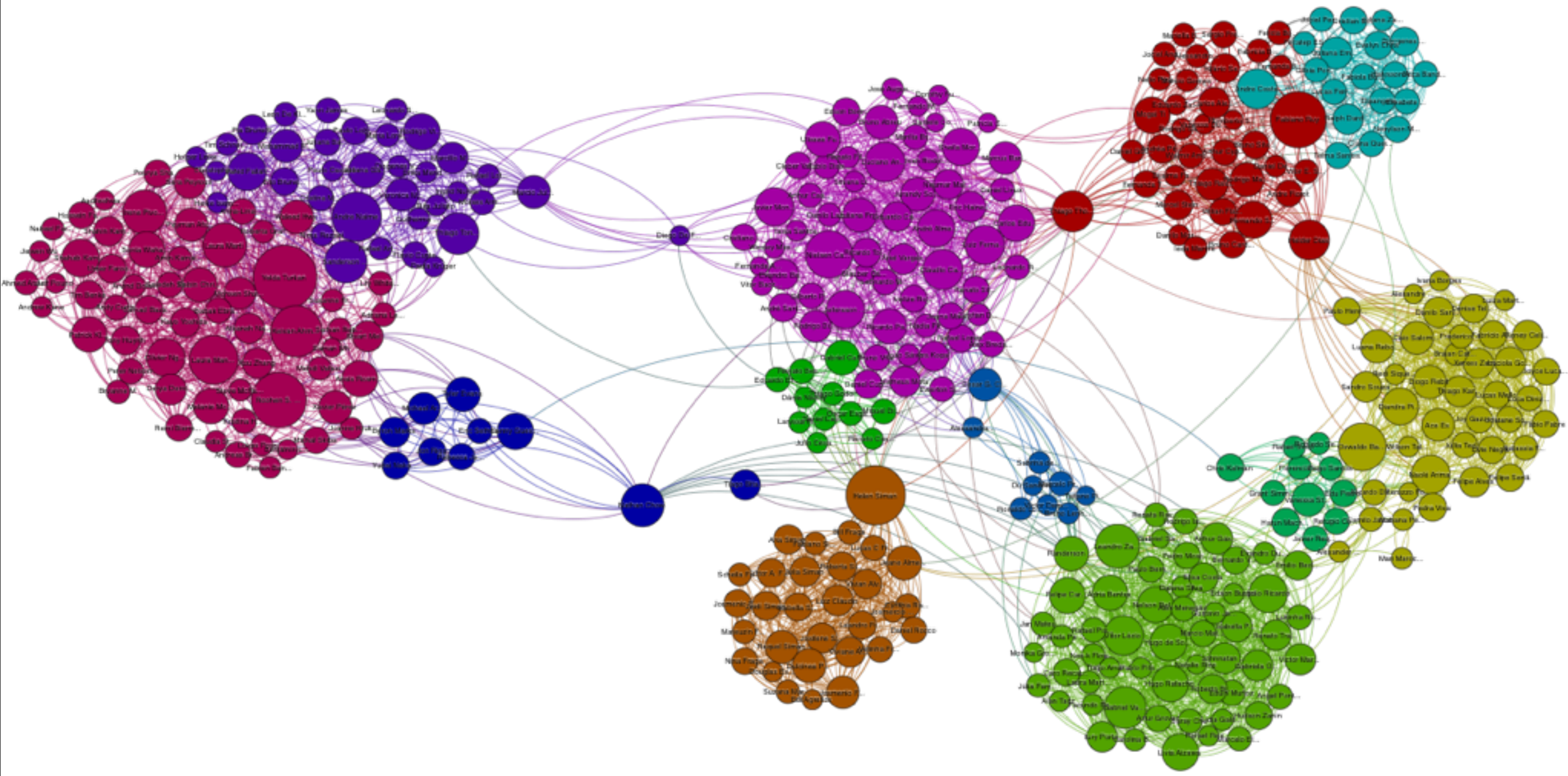
```
gremlin> m = [:].withDefault{1}
gremlin> v.transform{
  rank = m[it.name];
  neighbors = it.out('href').toList();
  degree = neighbors.size();
  neighbors.each {
    m[it.name] = m[it.name] + (rank/degree);
  }
  neighbors;
}.scatter.range(0,100000).loop(3){true}.iterate()
==>null
```



Graph DB Connectivity and Analysis

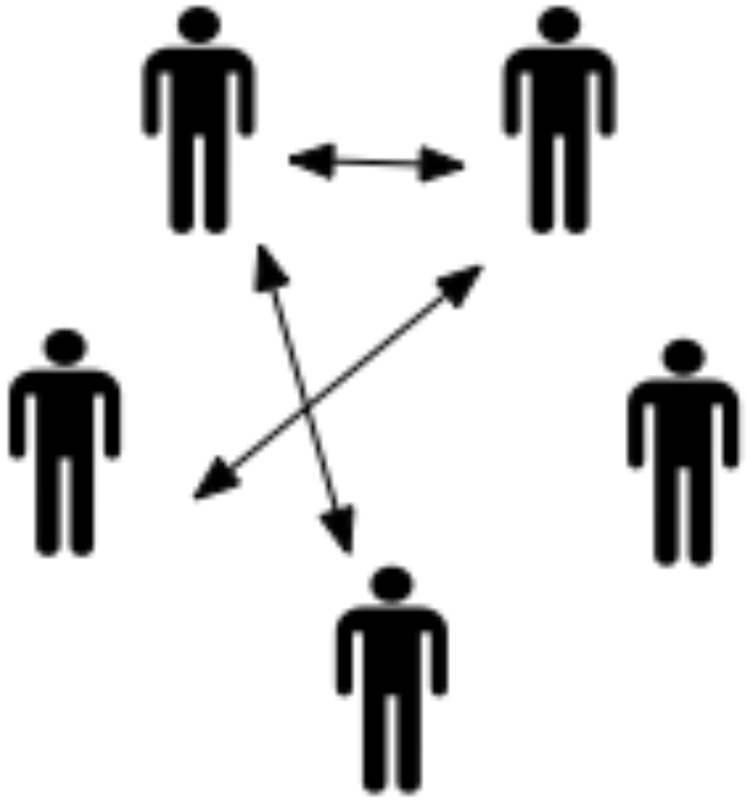


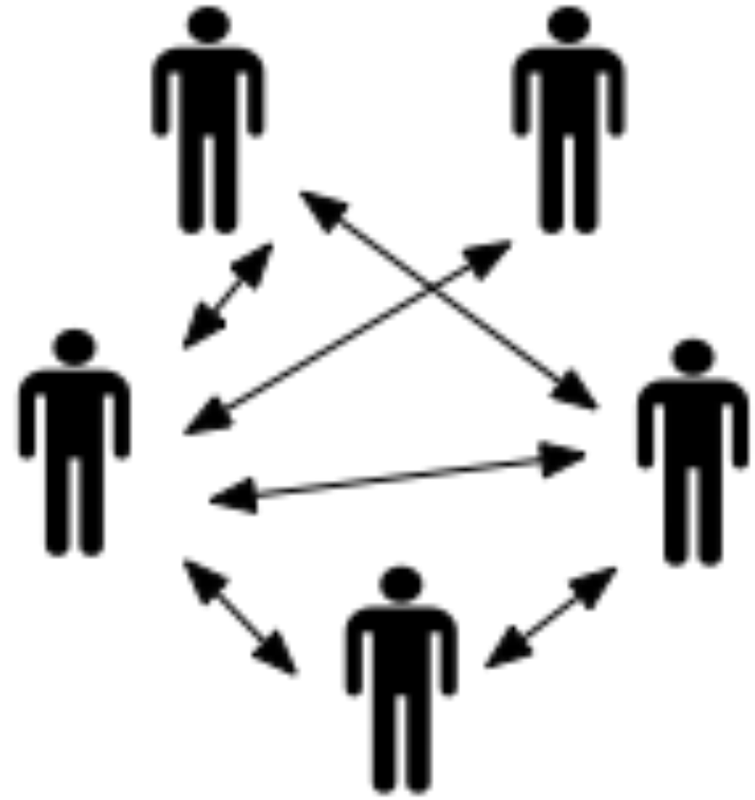
Graph Analysis - Complex Networks



Equal teams?

Equal teams?

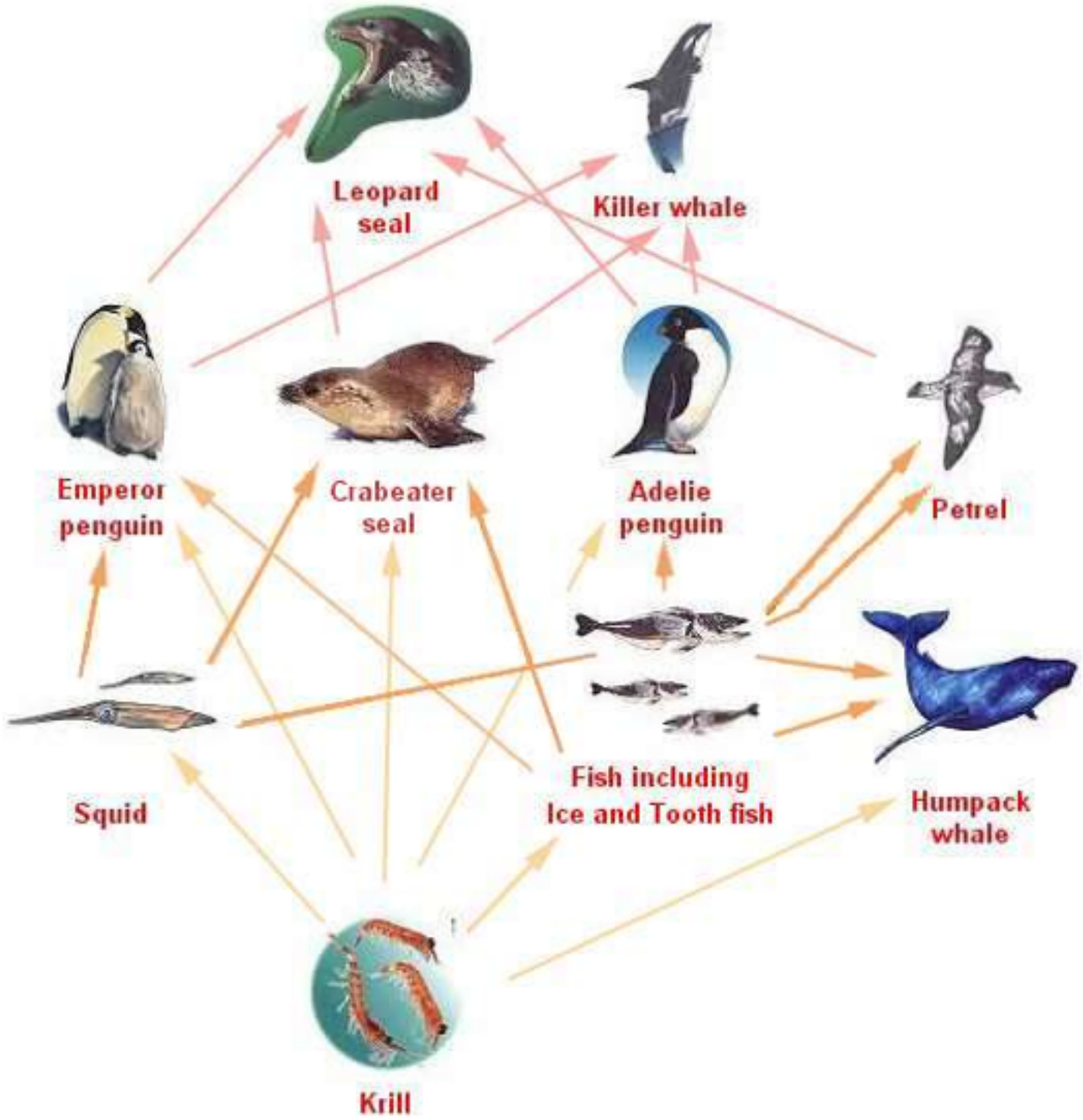




Social Networks

Who are the most influential employees in a company?

Biological Food Webs



What species would be more impacted by the extinction of Squids?

Citation Graphs

What are the research groups with the highest rate of collaboration?

Complex Networks - Definition

- A complex network is a graph (network) with non-trivial topological features

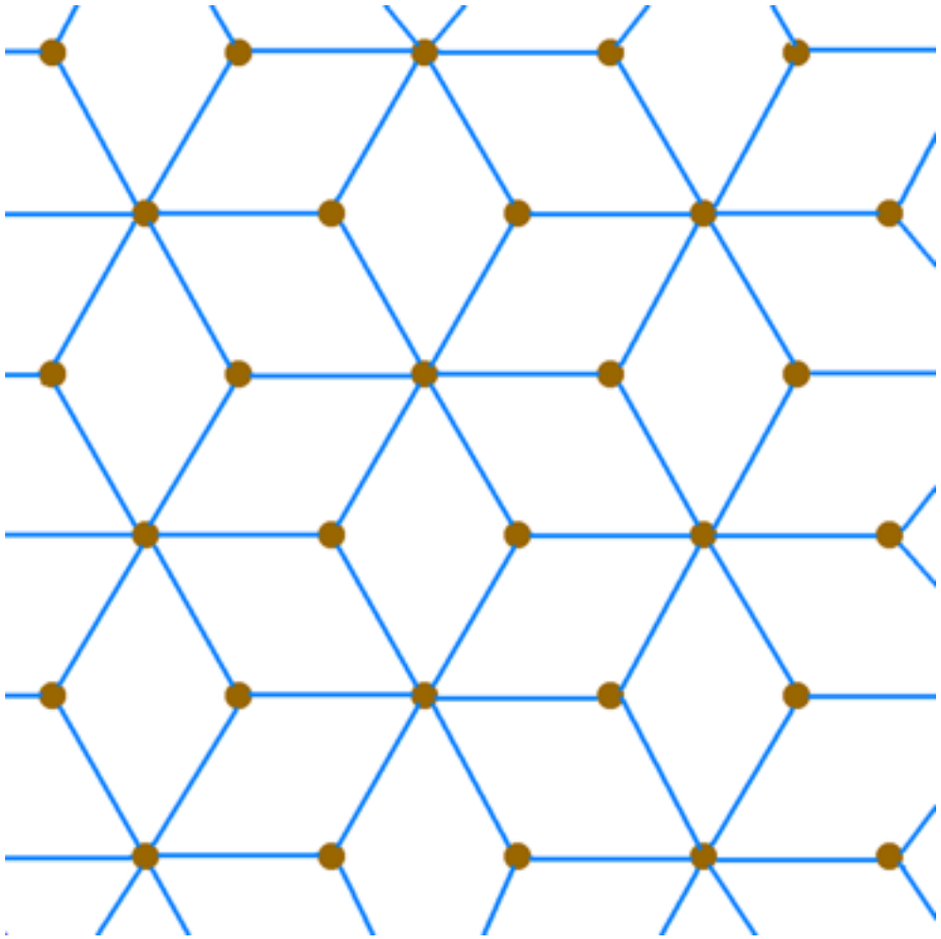
- A complex network IS NOT:

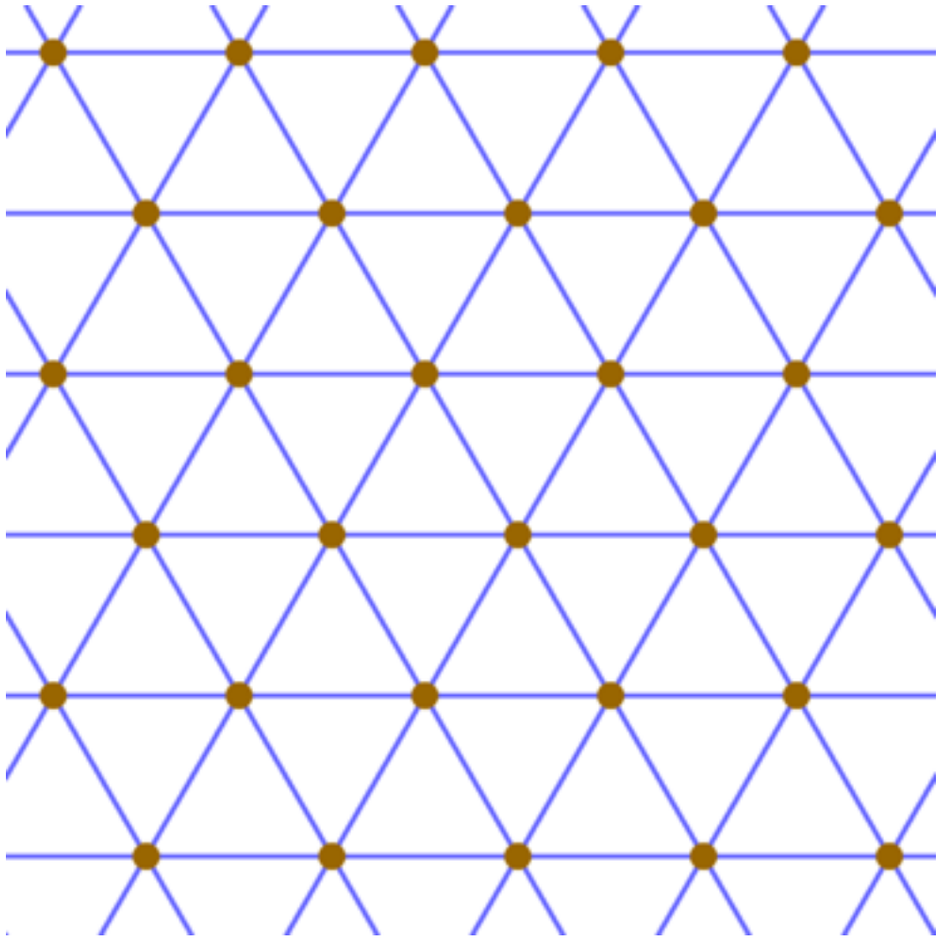
  - A lattice

  - A random graph

Lattices

- Regular topology
- Fixed node degrees (number of neighbors)

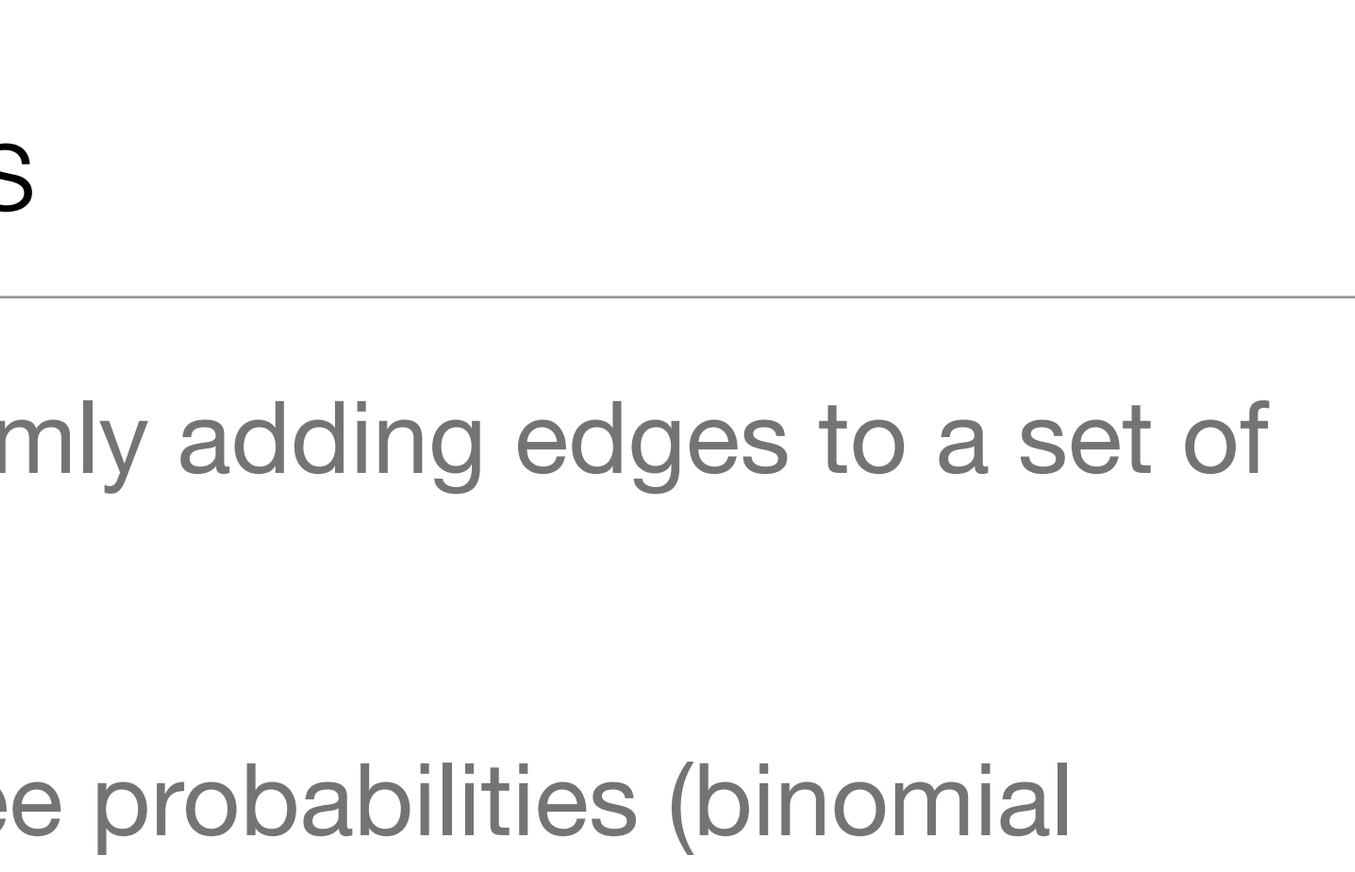




Random graphs

- Formed by randomly adding edges to a set of vertices

- Predictable degree probabilities (binomial distribution)



Complex Networks - Definition (Cont.)

- Formation of complex networks does not follow predictable plans
- Formation is based on localized phenomena
- Emergent behavior (cannot be assessed based merely on the analysis of parts of the system)
- Scale-free and small-world phenomena

Scale-free networks

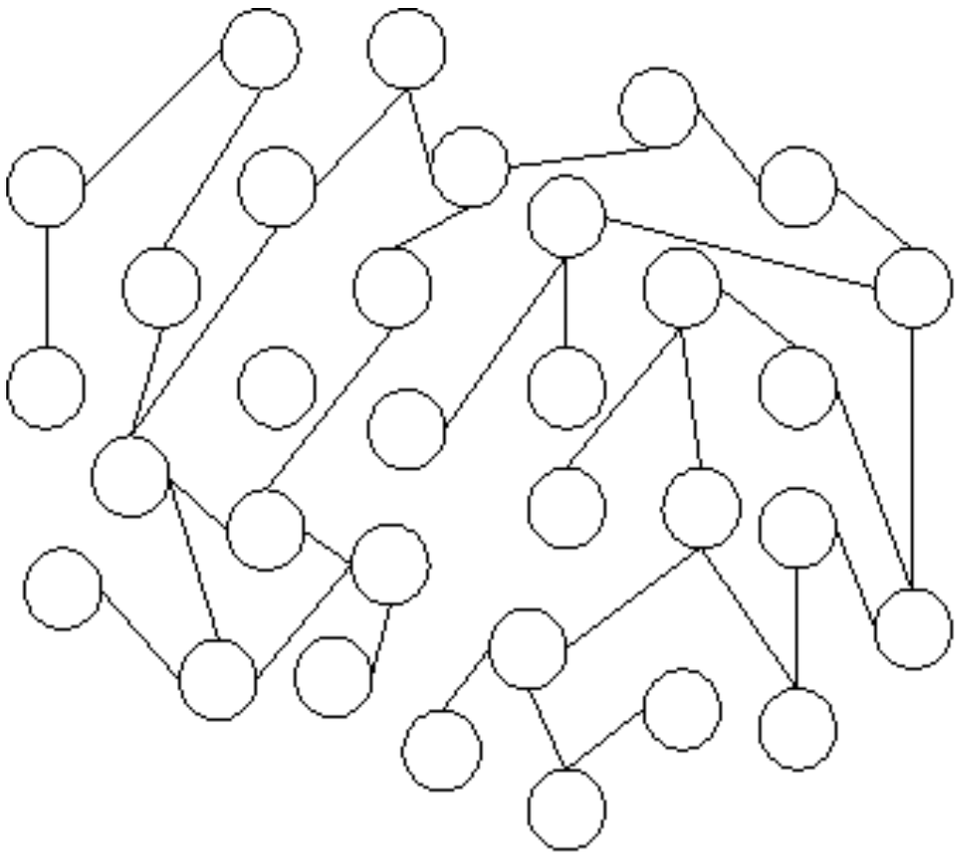
- Degree distribution follows a power law
  - Heavy-tailed distribution

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

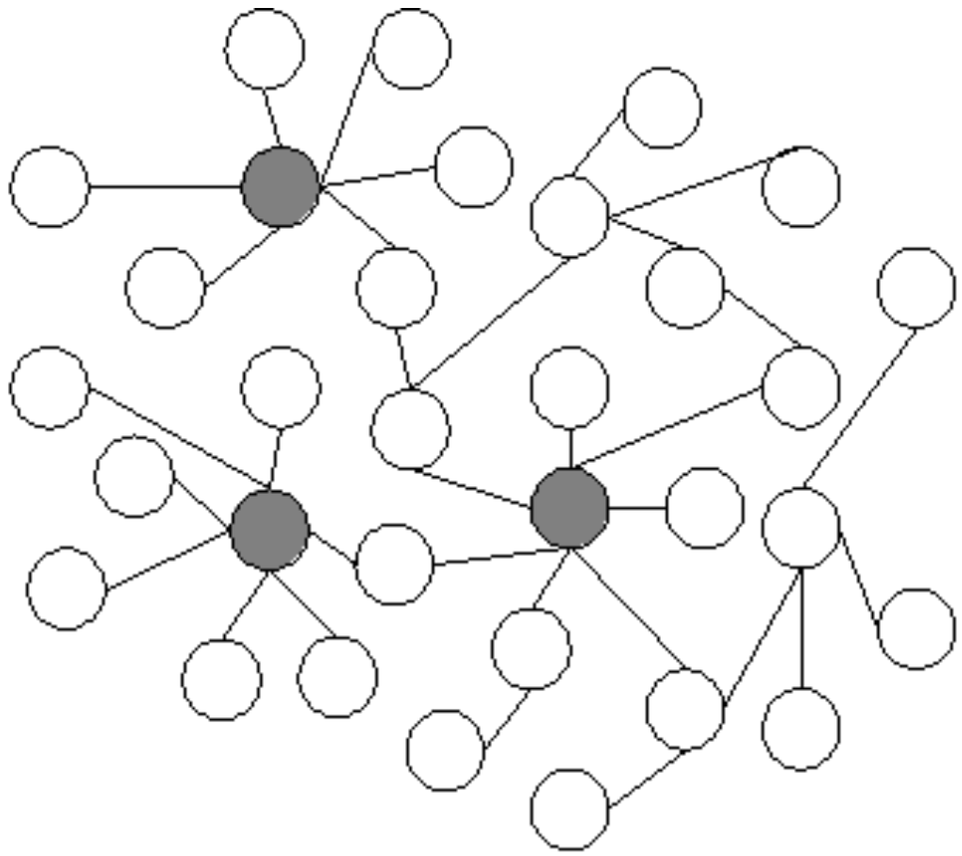
Prob. of a node having k connections

- Preferential attachment model
- Characteristics:
  - Robustness to failure (fault tolerant behavior)
  - Clustering coefficient distribution, which decreases as the node degree increases

Scale-free vs. Random Network

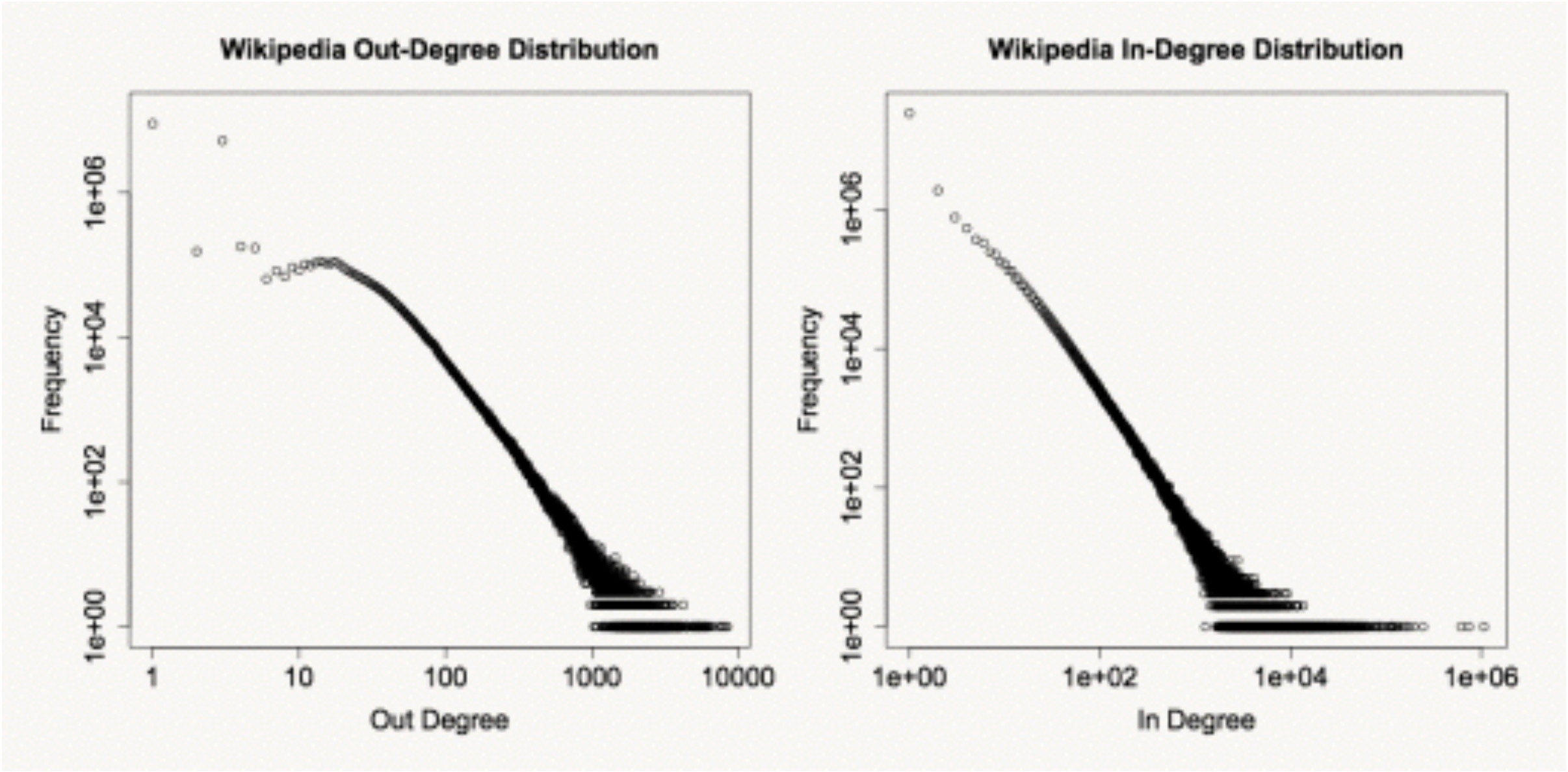


(a) Random network



(b) Scale-free network

Wikipedia page degree distributions



Scale-free networks - examples

- Social networks (facebook, movie actors, co-authorship, etc)
- Computer networks (internet, WWW)
- Biological networks (protein-protein interactions)
- Airline networks
- . . .

Small-world network

- Most pairs of nodes will be connected by at least one short path

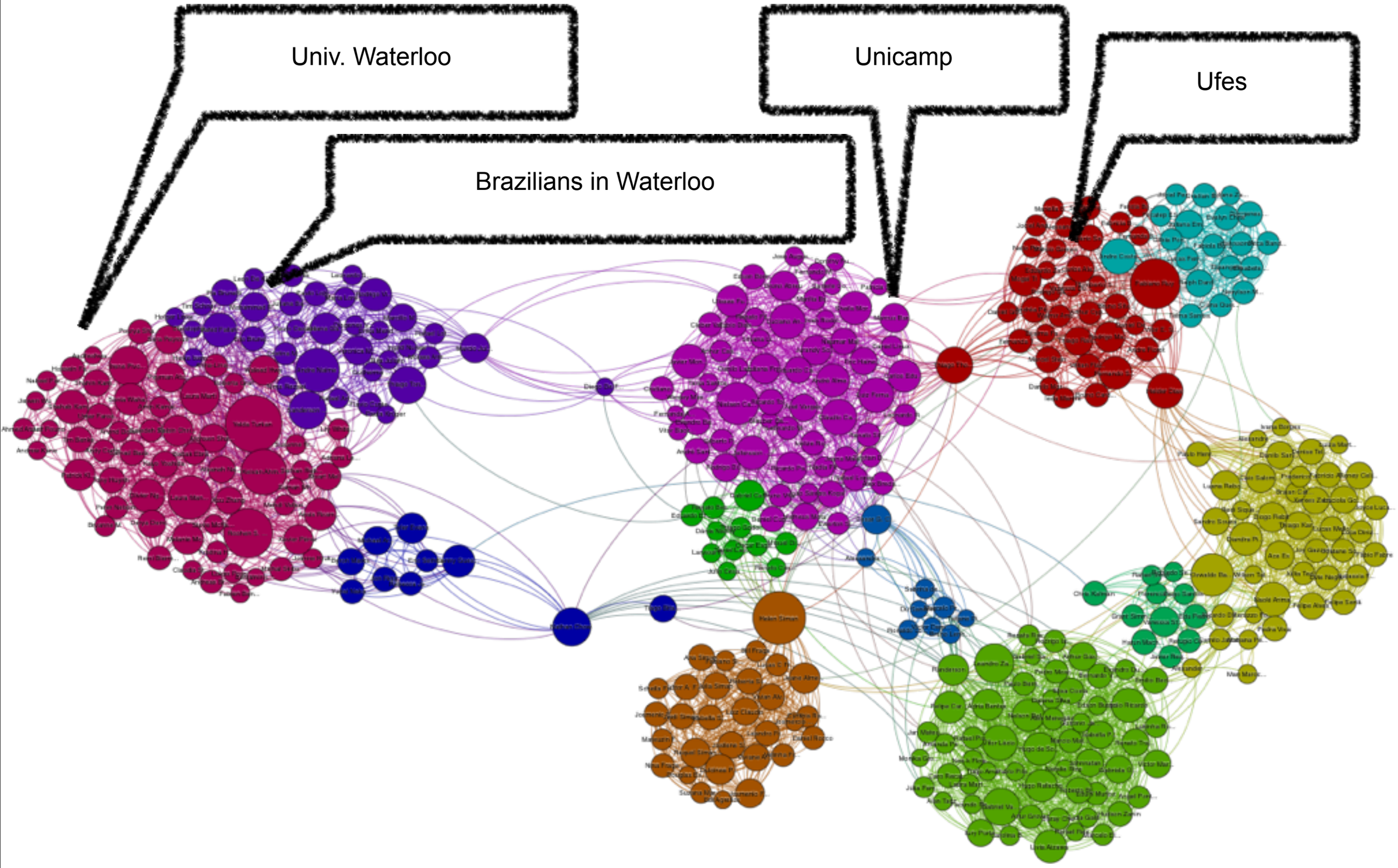
$$L \propto \log N$$

typical distance between two nodes

- High clustering coefficient
- "Six degrees of separation", Kevin Bacon
- Examples: road maps, food chains, electric power grids, metabolite processing networks, networks of brain neurons, voter networks, telephone call graphs, and social influence networks.

Complex Network Analysis

- Clustering and cycles
- Degree distribution
- Entropy
- Centrality
- . . .



http://persuasionradio.wordpress.com/2010/05/06/using-netvizz-gephi-to-analyze-a-facebook-network/

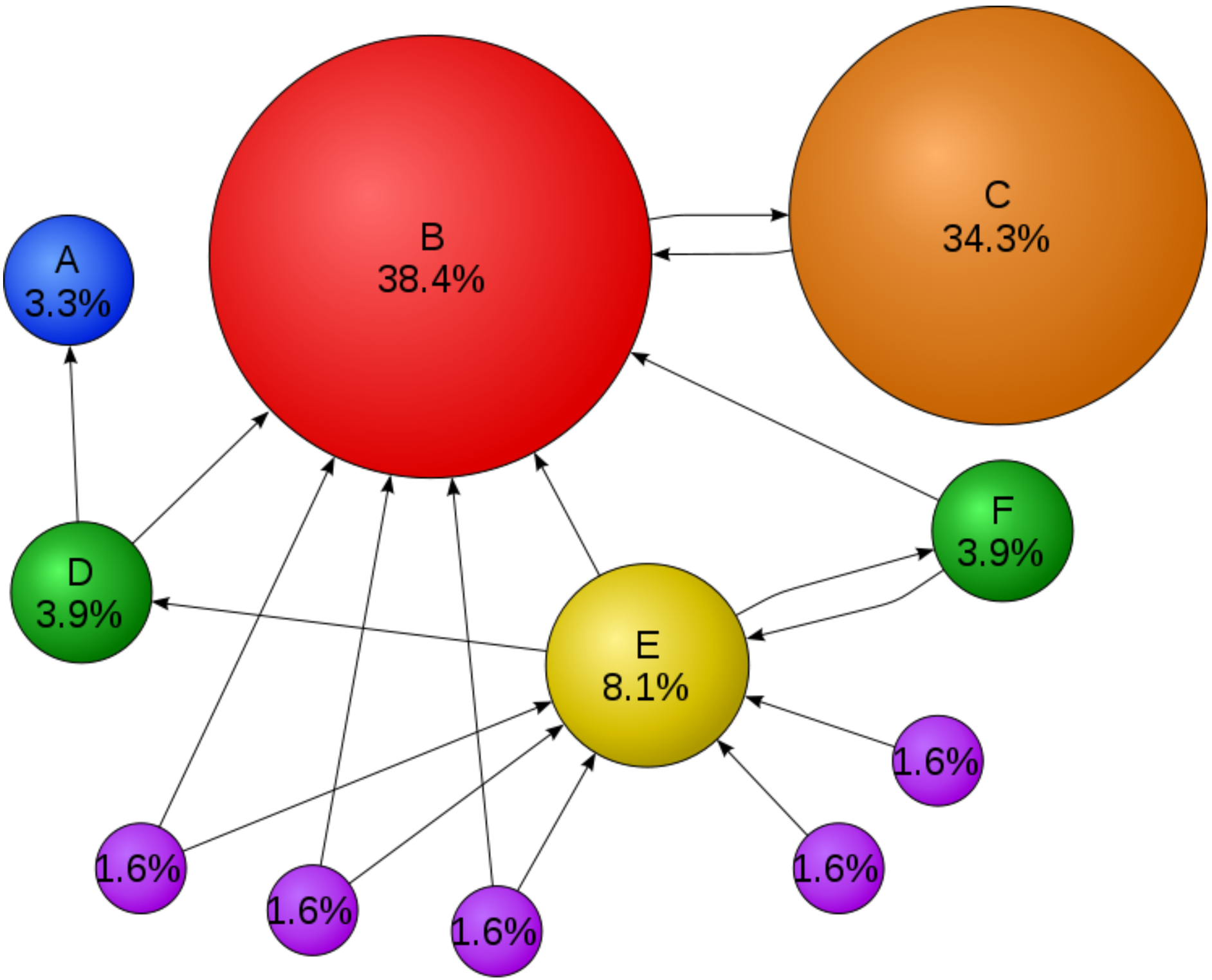
Network Centrality

- Degree centrality: how connected a node is
- Closeness centrality: how close to all other nodes
- Betweenness centrality: number of times shortest paths pass through a node
- Eigenvector centrality: measure of importance of a node (captures the notion that a connection with an important node makes a node more important)

PageRank

- Variation of Eigenvector centrality
- Models a random web surfer
  - Lands on a page -> clicks on links moving from page to page -> eventually gets bored and jumps to a random page, restarting the process
- PageRank is just one example of metric from one type of analysis… Many many more metrics have been proposed.

PageRank example



PageRank - simplified formula

$$PageRank\ of\ site = \sum \frac{PageRank\ of\ inbound\ link}{Number\ of\ links\ on\ that\ page}$$

OR

$$PR(u) = (1 - d) + d \times \sum \frac{PR(v)}{N(v)}$$

Exercício

- Em cada um dos níveis de um SGBD listados abaixo, descreva propriedades específicas que os diferenciaria num banco de dados de grafos.

  - Linguagem de Consulta
  - Processamento de Consultas
  - Estruturas/modelo de armazenamento

Summary

- Graphs are a more natural model for data in several application scenarios

- Graph databases provide adequate query languages and better performance

- The Complex Networks field analyses patterns that emerge form the topology of the graph

- Network analysis is an important tool in many applications

Exercício

- Em cada um dos níveis de um SGBD listados abaixo, descreva propriedades específicas que os diferenciaria num banco de dados de grafos.

  - Linguagem de Consulta: Permite a especificação de padrões de casamento de subgrafos (declarativas), permite navegação iterativa nos grafos (iterativas).

  - Processamento de Consultas: Processamento eficiente de consultas com atravessamento do grafo. Acesso eficiente aos vizinhos sem necessidade de índices.

  - Estruturas/modelo de armazenamento: Modelo/esquema flexível. Relacionamento explícito entre elementos.