INFSCI 1022 - Database Management Systems

NoSQL - Graph Databases

The FOUR V's of Big Data

From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?

As a leader in the sector, IBM data scientists break big data into four dimensions: **Volume, Velocity, Variety and Veracity**

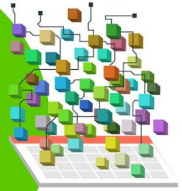
Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States

Volume
SCALE OF DATA

40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005

It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day



6 BILLION PEOPLE
have cell phones

WORLD POPULATION: 7 BILLION

Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored

Variety
DIFFERENT FORMS OF DATA

As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]

By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS

4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month

30 BILLION PIECES OF CONTENT
are shared on Facebook every month

400 MILLION TWEETS
are sent per day by about 200 million monthly active users

Velocity
ANALYSIS OF STREAMING DATA

The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session

Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure

By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth

Veracity
UNCERTAINTY OF DATA

1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions

Poor data quality costs the US economy around
$3.1 TRILLION A YEAR

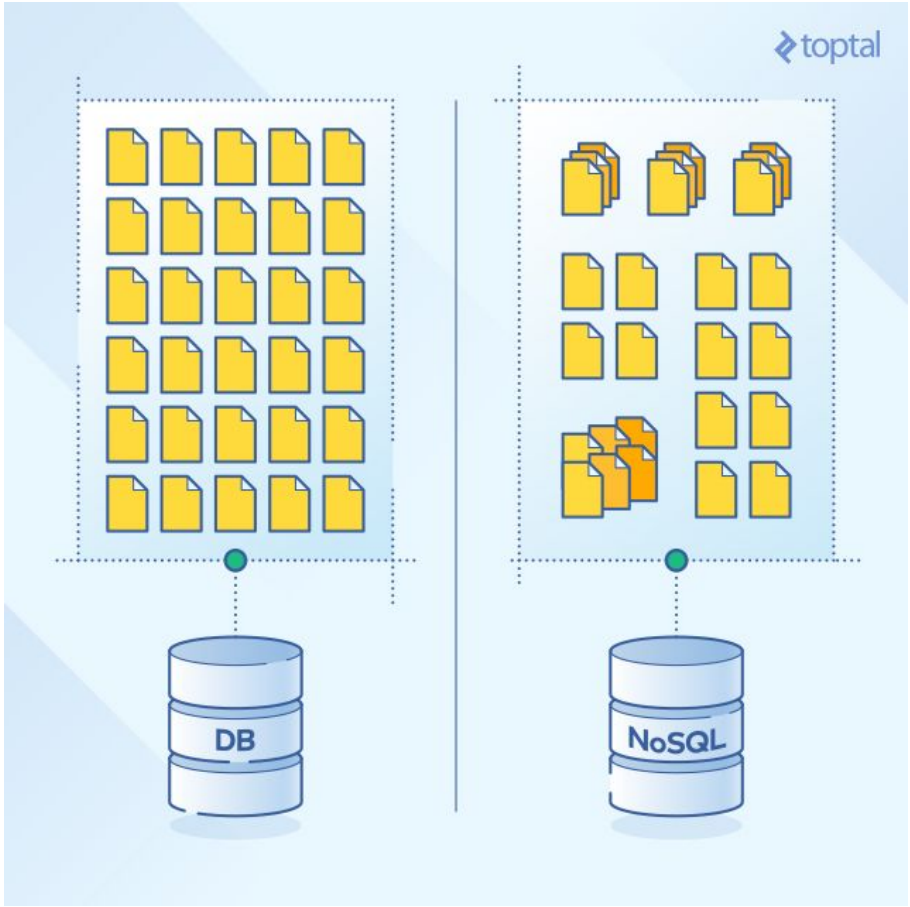
in one survey were unsure of how much of their data was inaccurate

Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS

IBM

NoSQL Databases

- NoSQL - NOT ONLY SQL
- NoSQL is a form of *unstructured storage*
- NoSQL databases do *not* have a fixed table structure like the ones found in relational databases.



NoSQL Advantages

- NoSQL databases have a simple and flexible structure. They are **schema-free**
- Unlike relational databases, NoSQL databases are based on key-value pairs
- Some store types of NoSQL databases include column store, **document store**, key value store, **graph store**, object store, XML store, and other data store modes.

NoSQL Advantages

- Usually, each value in the database has a key.
- Some NoSQL database stores also allow developers to store **serialized objects** into the database, not just simple string values.
- Expansion is easier and cheaper than when working with relational databases. This is because it’s done by **horizontally scaling** and distributing the load on all nodes, rather than the type of **vertical scaling** that is usually done with relational database systems, which is replacing the main host with a more powerful one.
-

NoSQL Disadvantages

- Most NoSQL databases do not support ***reliability features*** that are natively supported by relational database systems.
- These reliability features can be summed up as **atomicity, consistency, isolation, and durability**.
- This also means that NoSQL databases, which don’t support those features, trade **consistency** for **performance and scalability**.

NoSQL Disadvantages

- In order to support reliability and consistency features, developers must implement their own proprietary code, which adds more complexity to the system.
- This might limit the number of applications that can rely on NoSQL databases for secure and reliable transactions, like banking systems.
- Incompatibility with SQL queries. This means that a manual or proprietary querying language is needed, adding even more time and complexity.

NoSQL vs. Relational Databases

Feature	NoSQL Databases	Relational Databases
Performance	High	Low
Reliability	Poor	Good
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimized for huge data	Medium sized to large
Scalability	High	High (but more expensive)

NoSQL Store Type: Key-Value Store

Key	Value
"Belfast"	{“University of Ulster, Belfast campus, York Street, Belfast, BT15 1ED”}
“Coleraine"	{“University of Ulster, Coleraine campus, Cromore Road, Co. Londonderry, BT52 1SA”}

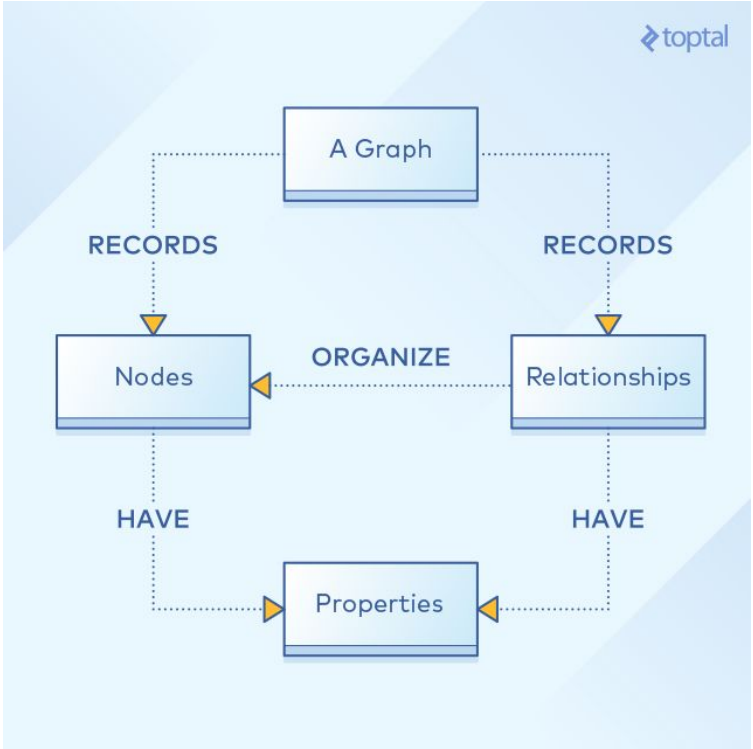
- In the Key Value store type, a hash table is used in which a unique key points to an item.
- Data is stored in a form of a string, JSON (Javascript Object Notation), or BSON (Binary Object Notation)
- One of the biggest flaws in this form of database is the lack of consistency at the database level.
- Example: Amazon’s DynamoDB

NoSQL Store Type: Key-Value Store

- Document stores are similar to key value stores in that they are schema-less and based on a key-value model.
- In Document Stores, the values (documents) provide encoding for the data stored.
- Those encodings can be XML, JSON, or BSON (Binary encoded JSON).
- Querying based on data can be done.
- Example: MongoDB.

NoSQL Store Type: Graph Base

- A directed graph structure is used to represent the data.
- The graph is comprised of edges and nodes.



ACID vs. BASE

ACID

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

BASE

- **B**asically Available
- **S**oft State
- **E**ventually Consistent

ACID: Atomicity

- The database transaction must completely succeed or completely fail.
- Partial success is not allowed.

ACID: Consistency

- During the database transaction, the RDBMS progresses from one valid state to another.
- The state is never invalid.

ACID: Isolation

- The client's database transaction must occur in isolation from other clients attempting to transact with the RDBMS.

ACID: Durability

- The data operation that was part of the transaction must be reflected in ***nonvolatile storage*** and **persist after** the transaction successfully completes.
- Transaction failures cannot leave the data in a partially committed state.

BASE: Basically Available

- The system is guaranteed to be available for querying by all users.
- No isolation.

BASE: Soft State

- The values stored in the system may change because of the eventual consistency model (next slide)

BASE: Eventually Consistent

- As data is added to the system, the system's state is gradually replicated across all nodes.
- Example:
  - In Hadoop, when a file is written to the HDFS, the replicas of the data blocks are created in different data nodes after the original data blocks have been written.
  - For the short period before the blocks are replicated, the state of the file system isn't consistent.

BASE

- Most NoSQL data stores don’t completely abandon *all* the ACID characteristics
- The Soft State and Eventually Consistent characteristics amount to the same thing
- By relaxing consistency, the system can horizontally scale (many nodes) and ensure availability.

ROMEO AND JULIET

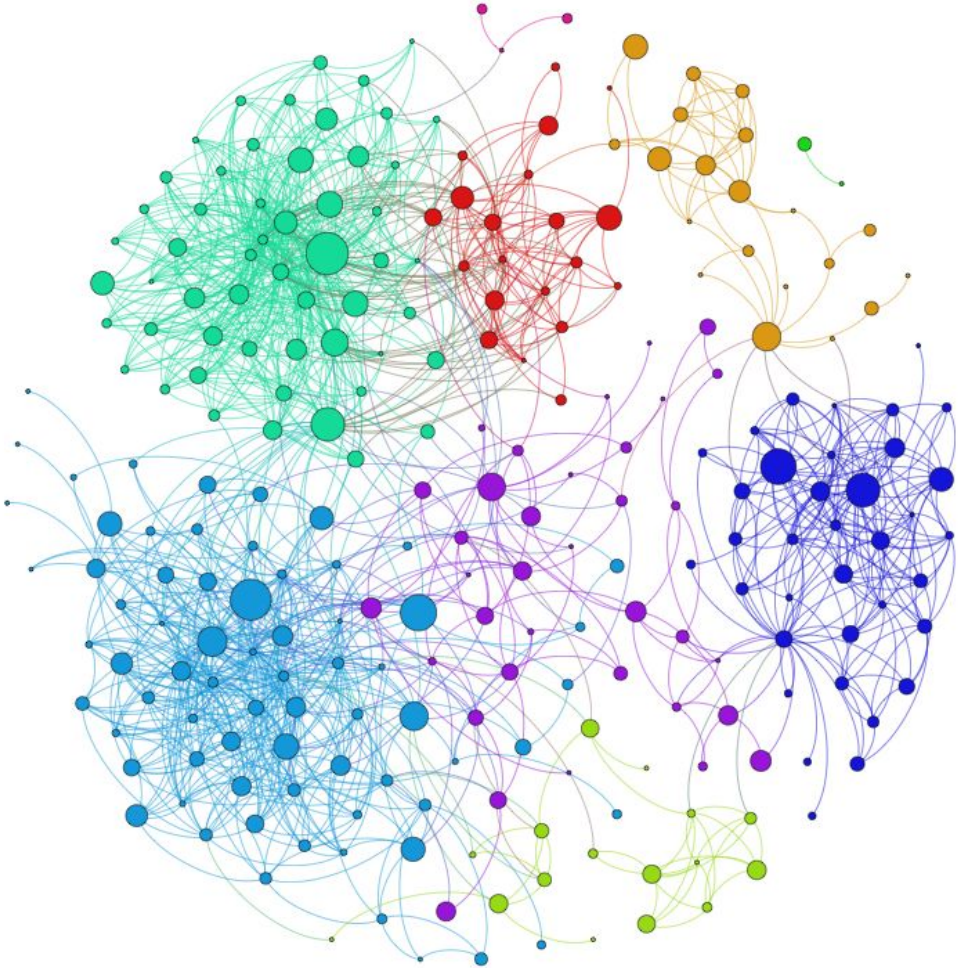
Number of characters **41** | **37%** Network density

MACBETH

Number of characters **46** | **25%** Network density

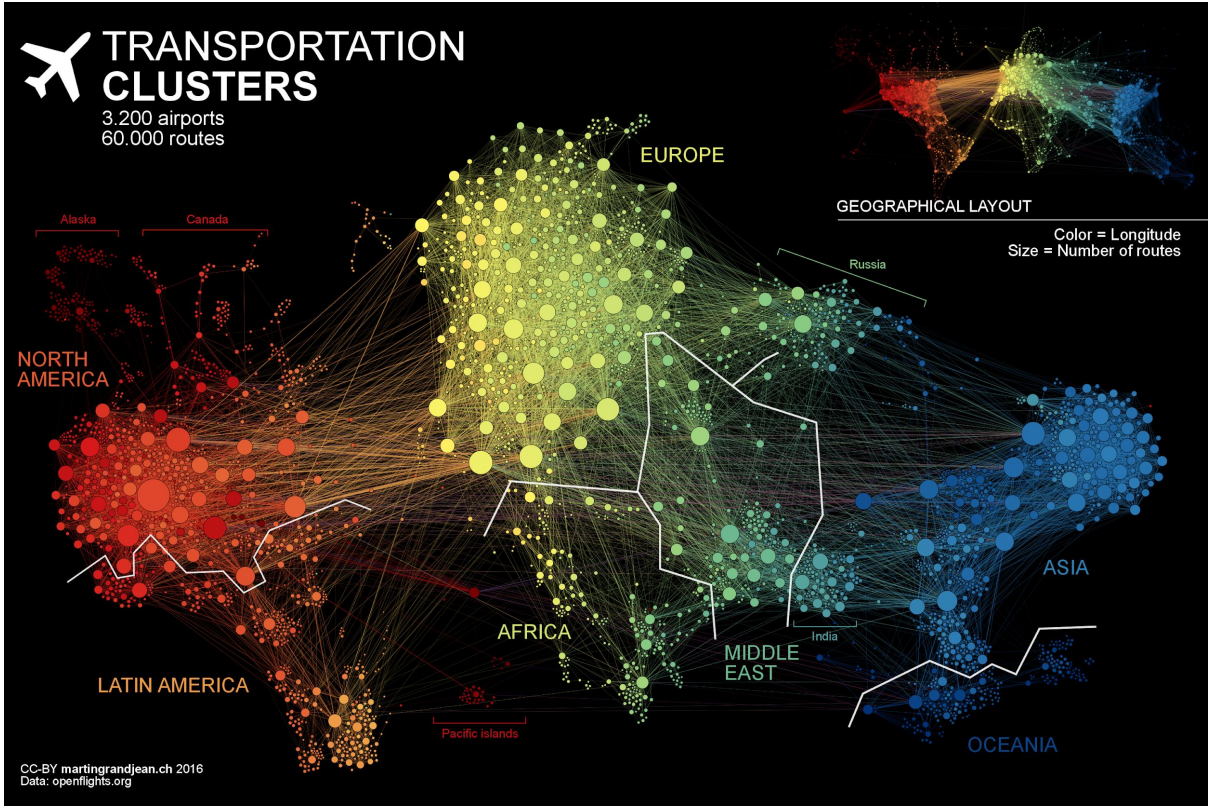
http://www.martingrandjean.ch/network-visualization-shakespeare/

LinkedIn Network



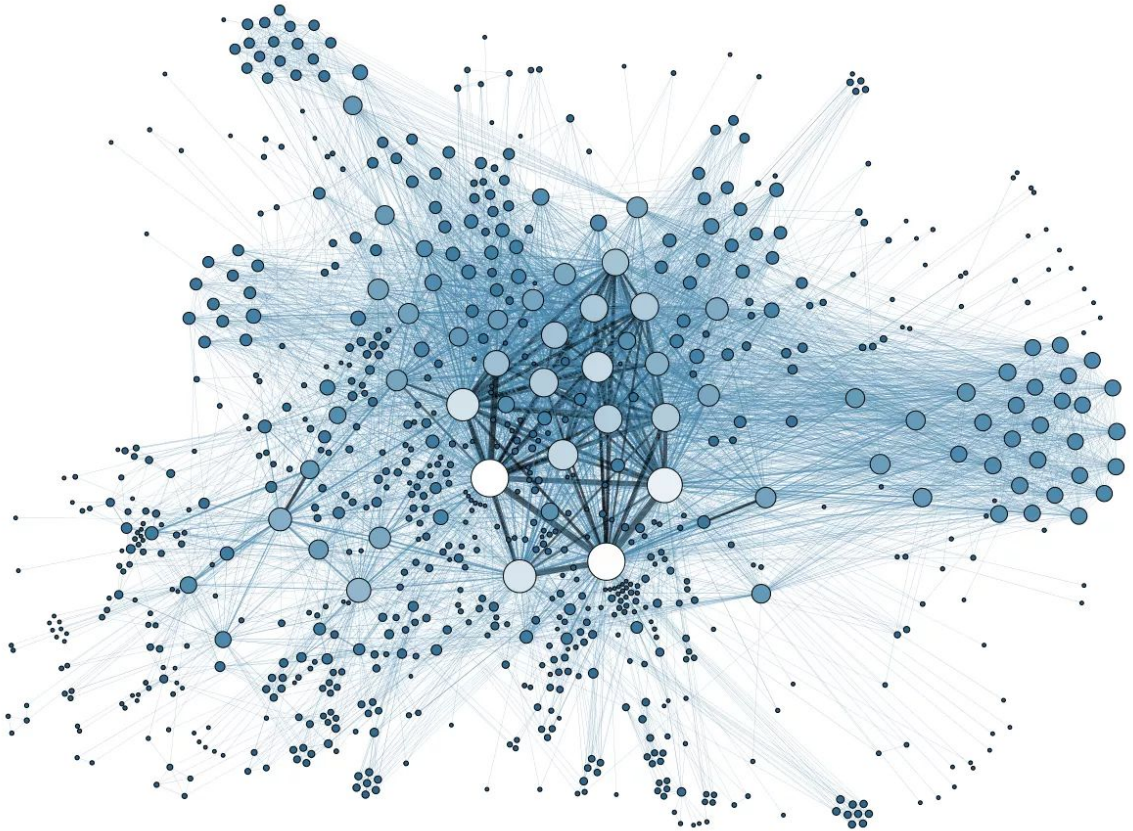
http://allthingsgraphed.com/2014/10/16/your-linkedin-network/

Global Air Travel Network



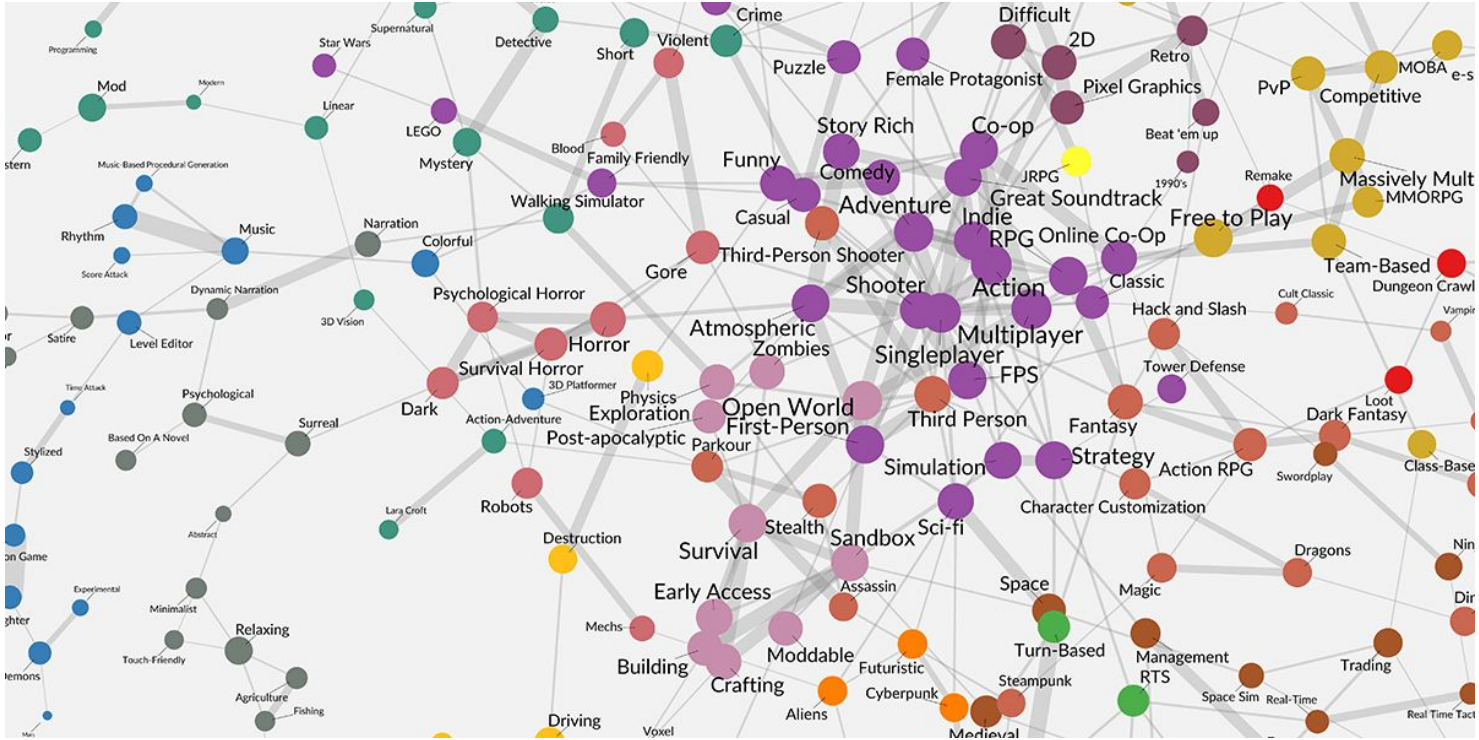
http://coolinfographics.com/blog/2016/6/3/the-global-air-transportation-network.html

Twitter Network Visualization



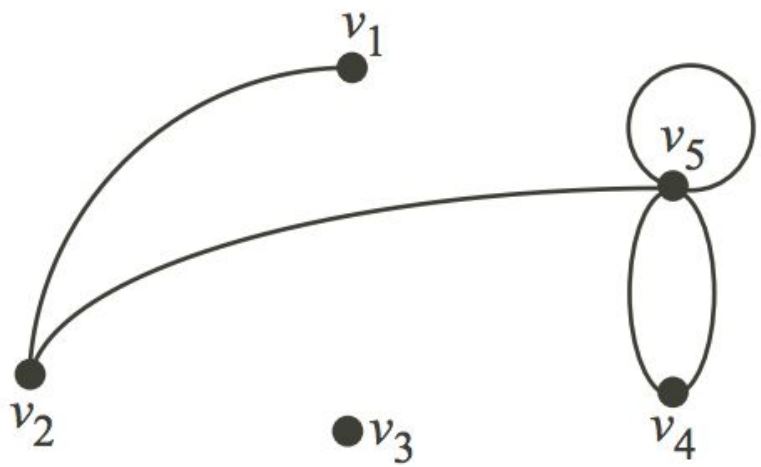
http://www.mikekujawski.ca/2014/08/13/two-free-twitter-network-visualization-tools/

Visualizing How Steam Tags Are Related



https://quanticfoundry.com/2018/01/24/visualizing-steam-tags-related/

Introduction to Graphs



- V = {v1, v2, v3, v4, v5}
- E = {(v1,v2), (v2,v5), (v5, v5), (v4,v5), (v4,v4)}

Graphs

- Graphs are mathematical structures used to study pairwise relationships between objects and entities.
- It is a branch of Discrete Mathematics and has found multiple applications in Computer Science, Chemistry, Linguistics, Operations Research, Sociology etc.
- Graphs are used in data mining to model various structures and problems.

Graphs

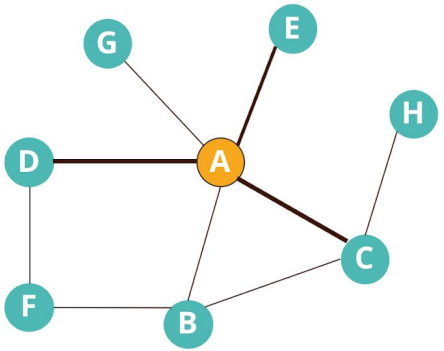
- A Graph is a pair of sets. G = (V,E).
  - V is the set of vertices.
  - E is a set of edges.
    - E is made up of pairs of elements from V (unordered pair)

DiGraphs (directed graphs)

- A DiGraph is also a pair of sets. D = (V,A).
  - V is the set of vertices.
  - A is the set of arcs.
    - A is made up of pairs of elements from V (ordered pair)
- In digraphs, there is a distinction between `(u,v)` and `(v,u)`.
- Usually the **edges** are called **arcs** in such cases to indicate a notion of direction.

COMPONENTS OF A NETWORK

SAMPLE NETWORK



COMPONENTS OF A NETWORK

Vertex

set of objects (also called nodes) that are connected together

1. Vertex attributes define a vertex based on its characteristics. E.g.: For airline routes, if Vertex are cities, attributes could be the city's population

Edge

The connections between the nodes are called edges or links

1. If the edges in a network are directed, i.e., pointing in only one direction, the network is called a directed network.
2. If all edges are bidirectional, or undirected, the network is an undirected network
3. Thickness of the edge determines the relationship between the 2 related vertices

EXAMPLES OF NETWORKS AND THEIR COMPONENTS

NETWORK	VERTICES	VERTEX ATTRIBUTES	EDGES	EDGE ATTRIBUTES
Airlines Network	**Airports**	Footfall, Terminals, Staff, City population, International/Domestic, Freight, Hangar capacity	**Airplanes / Routes**	Frequency, # Passengers, Plane Type, Fuel Usage, Distance covered, Empty seats
Banking Network	**Account Holders**	Name, demographics, KYC Document, Products, Account status, balance and other details	**Transactions**	Type, Amount, Authentication (pass/OTP), Time, Location, Device
Social Network	**Users**	Name, demographics, # connections, likes, circles belong to, subscriptions	**Interactions**	Medium (like/comment/direct message), time, duration, type of content, topic
Physician Network	**Doctors**	Demographics, speciality, experience, affiliation (type and size), Weekly patient intake	**Patients**	Demographics, Diagnosis history, visit frequency, purpose, referred to, insurance
Supply Chain Network	**Warehouses**	Location, size, capacity, storage type, connectivity, manual/automated	**Trucks**	Load capacity, # wheels, year of make, geographical permit, miles travelled. Maintenance cost, driver experience

Applications of Networks Examples

- **Marketing Analytics –** Graphs can be used to figure out the most influential people in a Social Network. Advertisers and Marketers can estimate the biggest bang for the marketing buck by routing their message through the most influential people in a Social Network
- **Banking Transactions –** Graphs can be used to find unusual patterns helping in mitigating Fraudulent transactions. There have been examples where Terrorist activity has been detected by analyzing the flow of money across interconnected Banking networks

Applications of Networks Examples

- **Supply Chain –** Graphs help in identifying optimum routes for your delivery trucks and in identifying locations for warehouses and delivery centres
- **Pharma –** Pharma companies can optimize the routes of the salesman using Graph theory. This helps in cutting costs and reducing the travel time for salesman
- **Telecom –** Telecom companies typically use Graphs (Voronoi diagrams) to understand the quantity and location of Cell towers to ensure maximum coverage

Why Graphs?

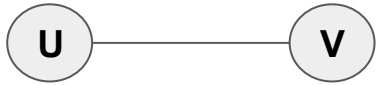
- Graphs provide a better way of dealing with abstract concepts like relationships and interactions.
- They also offer an intuitively visual way of thinking about these concepts.
- Form a natural basis for analyzing relationships in a Social context

Why Graphs?

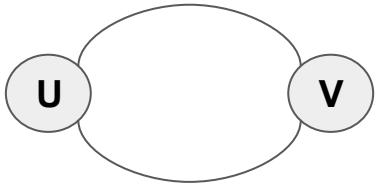
- Graph Theory concepts are used to study and model
  - Social Networks
  - Fraud patterns
  - Power consumption patterns
  - Virality and Influence in Social Media.
- Social Network Analysis (SNA) is probably the best known application of Graph Theory for Data Science

Terminology

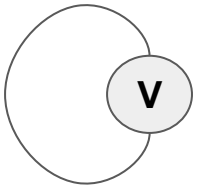
- The vertices u and v are called the end vertices of the edge (u,v)



- If two edges have the same end vertices they are Parallel

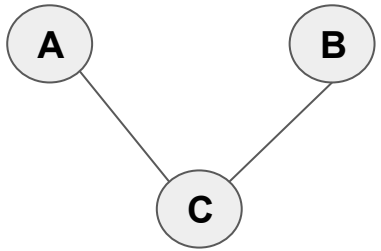


- An edge of the form (v,v) is a loop



Terminology

- A Graph is simple if it has **no parallel edges and loops**

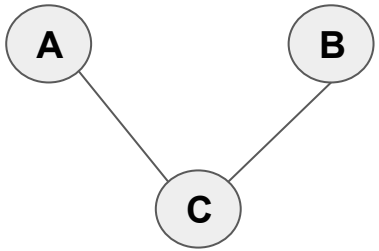


- A Graph is said to be Empty if it has **no edges**. Meaning E is empty

- A Graph is a Null Graph if it has **no vertices**. Meaning V and E is empty
- A Graph with only **1 Vertex** is a Trivial graph

Terminology

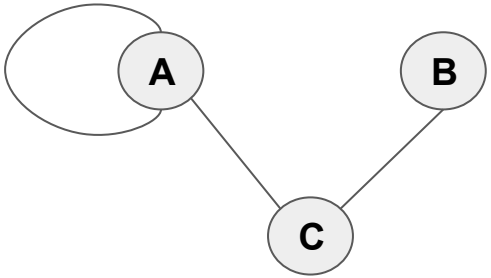
- **Edges** are Adjacent if they have a common vertex. *{A,C} and {B,C} are adjacent*



- **Vertices** are Adjacent if they have a common edge. *A and B are adjacent*

Terminology

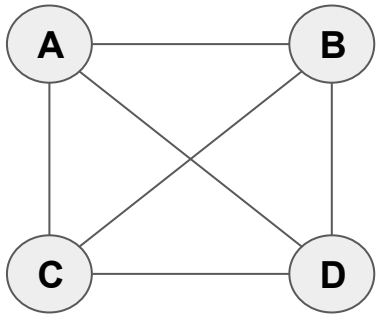
- The **degree** of the vertex v, written as d(v), is the number of **edges** with v as an end vertex. By convention, we count a loop twice and parallel edges contribute separately.
  Ex:
  - d(C) = 2
  - d(A) = 3



- **Isolated Vertices** are vertices with degree 1. d(1) vertices are isolated

Terminology

- A Graph is **Complete** if its edge set contains every possible edge between ALL of the vertices



Average Path Length

- The average of the shortest path lengths for all possible node pairs.
- Gives a measure of “tightness” of the Graph C
- Can be used to understand how quickly/easily something flows in this Network

Centrality

- Centrality aims to find the most important nodes in a network.
- There may be different notions of “**important**” and hence there are many centrality measures.
- Centrality measures themselves have a form of classification (or Types of centrality measures).

Centrality Measures: **Degree Centrality**

- The number of edges connected to a node.
- In the case of a directed graph, we can have 2 degree centrality measures. Inflow and Outflow Centrality
- Closeness Centrality – Of a node is the average length of the shortest path from the node to all other nodes
- Betweenness Centrality – Number of times a node is present in the shortest path between 2 other nodes

Centrality Measures: **Closeness Centrality**

- The average length of the shortest path from the node to all other nodes

Centrality Measures: **Betweenness Centrality**

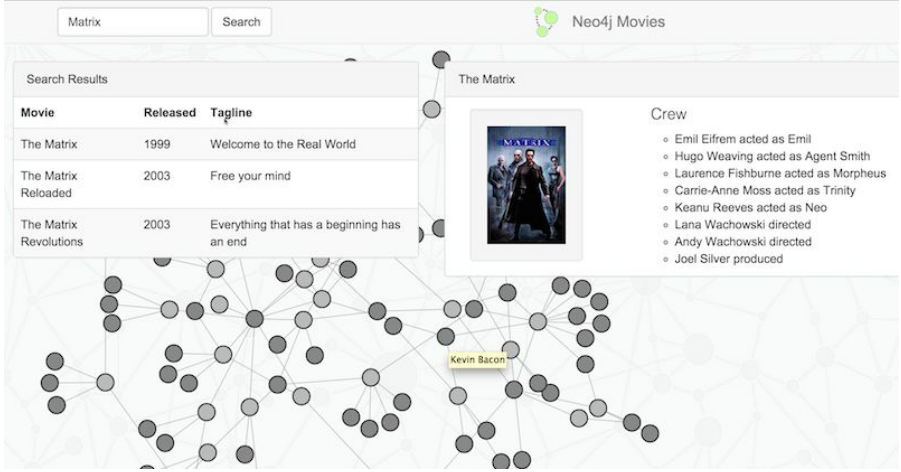
- Number of times a node is present in the shortest path between 2 other nodes

Network Density

- A measure of how many edges a Graph has.
- The actual definition will vary depending on type of Graph and the context in which the question is asked.
- For a complete undirected Graph the Density is 1, while it is 0 for an empty Graph.
- Graph Density can be greater than 1 in some situations (involving loops).

Neo4J

- Graph Databases have become common computational tools and alternatives to SQL and NoSQL databases
- Neo4J: https://neo4j.com/
- Example: http://my-neo4j-movies-app.herokuapp.com/



Neo4J Querying vs. SQL

```
SELECT actor, COUNT(m.movie_id)
FROM movies m
JOIN movies_actors ma ON m.movie_id = ma.fk_movie_id
JOIN actors a ON ma.fk_actor_id = a.actor_id
GROUP BY actor HAVING COUNT(m.movie_id) < 3
```

```
MATCH (a:Actor)-[:ACTS_IN]->(m:Movie)
WITH a, count(m) AS movie_count
WHERE movie_count < 3
RETURN a, movie_count
ORDER BY movie_count DESC LIMIT 5;
```

Attribution

This lecture is largely based on the following two articles:

- "An Introduction to Graph Theory and Network Analysis": https://www.analyticsvidhya.com/blog/2018/04/introduction-to-graph-theory-network-analysis-python-codes/
- The Definitive Guide to NoSQL Databases: https://www.toptal.com/database/the-definitive-guide-to-nosql-databases

References

- History of Graph Theory || S.G. Shrinivas et. al: http://www.cs.xu.edu/csci390/12s/IJEST10-02-09-124.pdf
- Networkx reference documentation: https://networkx.github.io/documentation/stable/reference/index.html
- Graphviz download: http://www.graphviz.org/download/
- Pygraphvix: http://pygraphviz.github.io/
- Star visualization: https://github.com/pygraphviz/pygraphviz/blob/master/examples/star.py
- Dijkstra Algorithm: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm