# Big Data Systems

Djellel Difallah

Spring 2023

Lecture 7 – More NOSQL:

Graph DBMS, Array DBMS

# Agenda

- Graph DBMS
  - Introduction
  - Storage and Indexing
  - Partitioning

- Array DBMS
  - Introduction
  - Physical Array Storage
  - Space filling curves

# Small vs Large Graphs

- Small graphs
  - Manage a collection of small graphs
  - Bioinformatics and Cheminformatics
  - Usually, data fit in memory (well studied)

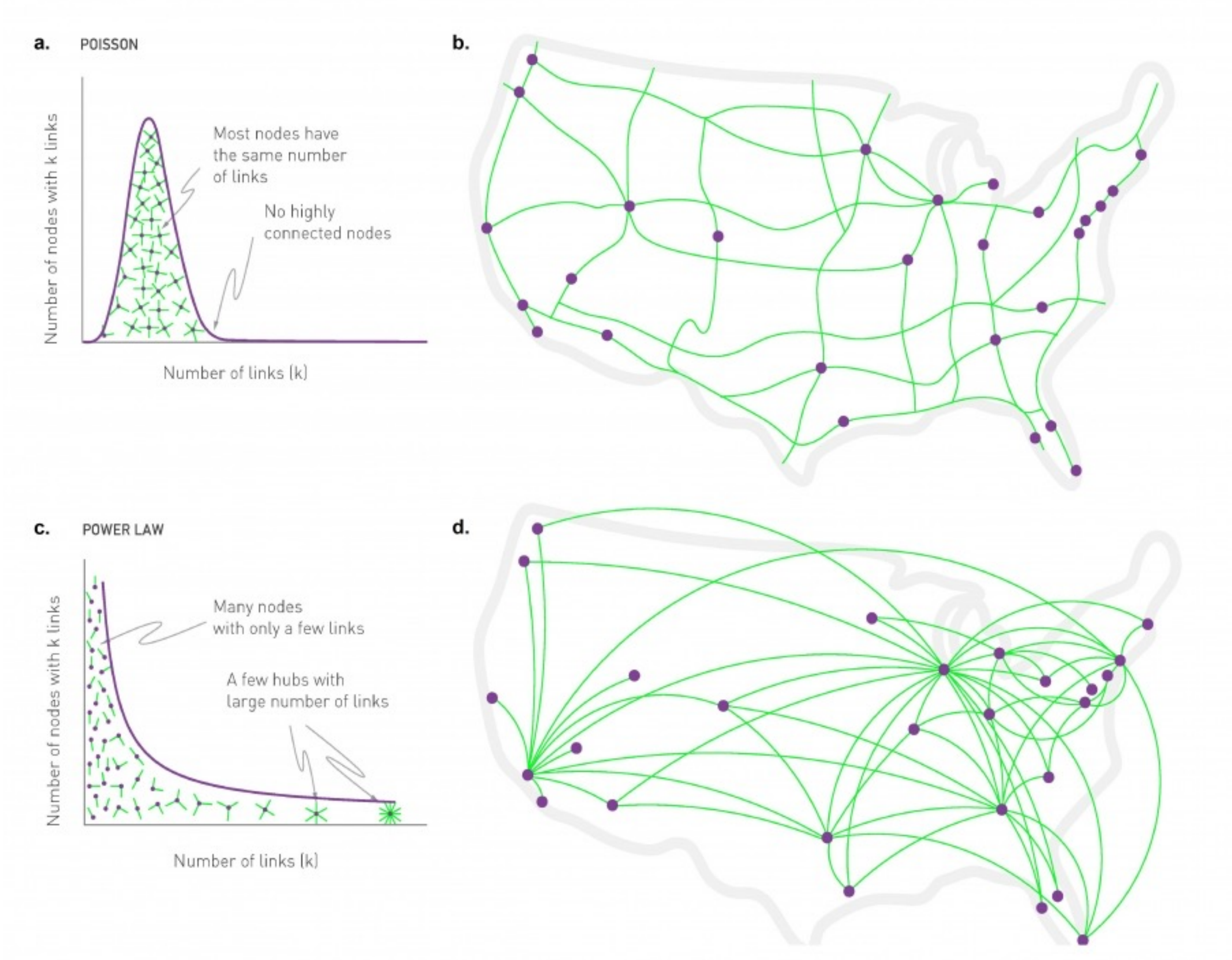
- Large graphs
  - One large graph, aka “network”
  - Social network
  - Knowledge graphs
  - Active area of research

# Random vs Scale-free Graphs

- Random graphs
  - Node degree is constrained

- Scale-free graphs
  - Distribution of node degree follows a power law distribution
  - Most large graphs are scale-free
  - Small world phenomena & hubs
  - hard to partition

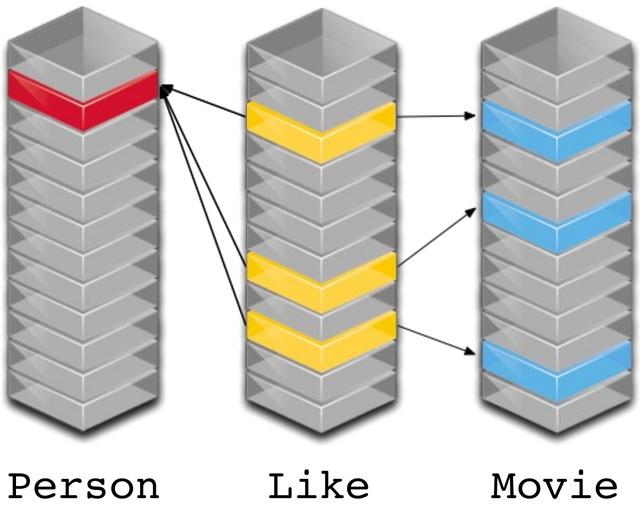
# Random vs Scale-free Graphs

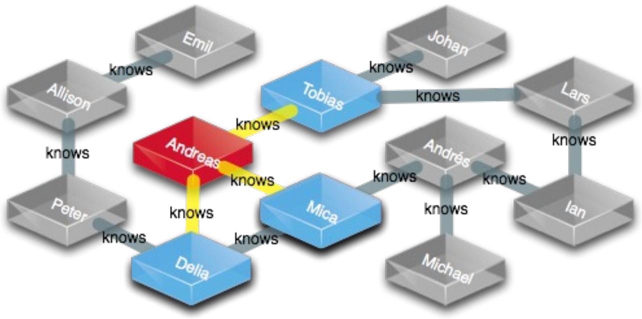

a.
POISSON
Number of nodes with k links
Most nodes have the same number of links
No highly connected nodes
Number of links (k)
b.
c.
POWER LAW
Number of nodes with k links
Many nodes with only a few links
A few hubs with large number of links
Number of links (k)
d.

Network Science. By Albert-László Barabási. http://networksciencebook.com/

# Graphs: a unique Data Model

- While the relational model is flexible to represent a large , specialized models tends to gives huge execution benefits



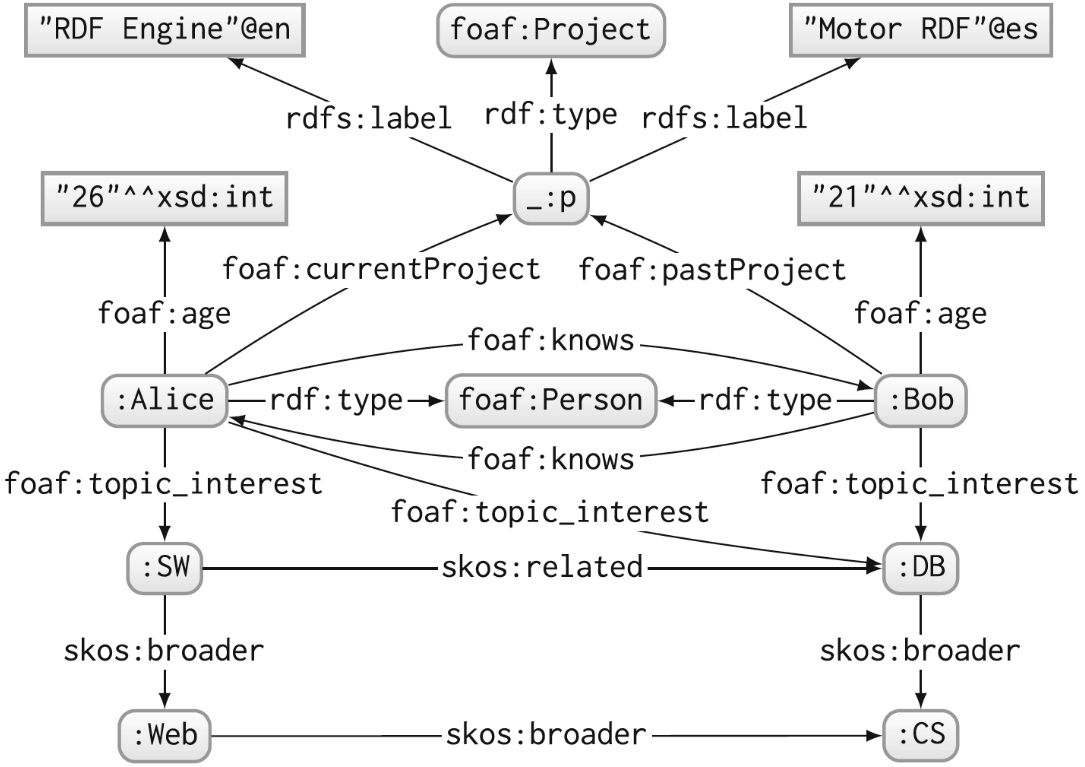


# Graph Database

- Optimized for the connections between records
  - Fast querying across records
- Transactional with CRUD operations (create/read/update/delete)
- Difference in the use case:
  - A relational database may tell you how many books your read last year
  - A graph database will tell you *which books you and the friends of your friends have read in common*
- Query Languages:
  - Many specialized and/or proprietary languages
  - SPARQL standard for RDF data

# Triple Pattern and Basic Graph Pattern

- Recall triples: s (subject), p (predicate), o (object).
  - The unit information in a property graph



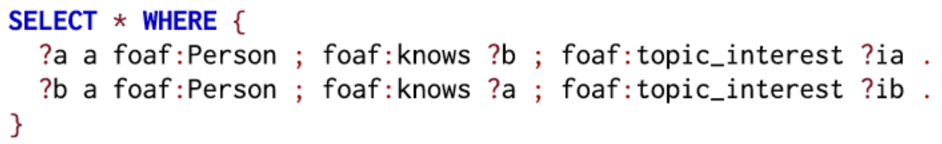
| Subject | Predicate | Object |
|---|---|---|
| :Alice | rdf:type | foaf:Person |
| :Alice | foaf:age | "26"^^xsd:int |
| :Alice | foaf:topic_interest | :DB |
| :Alice | foaf:topic_interest | :SW |
| :Alice | foaf:knows | :Bob |
| :Alice | foaf:currentProject | _:p |
| :Bob | rdf:type | foaf:Person |
| :Bob | foaf:age | "21"^^xsd:int |
| :Bob | foaf:topic_interest | :DB |
| :Bob | foaf:knows | :Alice |
| :Bob | foaf:pastProject | _:p |
| _:p | rdf:type | foaf:Project |
| _:p | rdfs:label | "RDF Engine"@en |
| _:p | rdfs:label | "Motor RDF"@es |
| :SW | skos:broader | :Web |
| :SW | skos:related | :DB |
| :Web | skos:broader | :CS |
| :DB | skos:broader | :CS |

*Ali, Waqas, et al. "A survey of RDF stores & SPARQL engines for querying knowledge graphs." The VLDB Journal (2022): 1-26.*

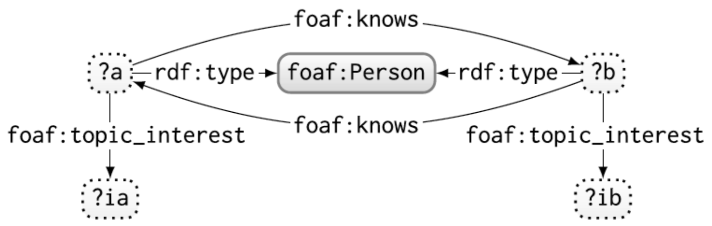
# Triple Pattern and Basic Graph Pattern

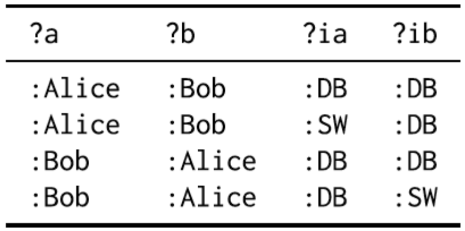
- **Triple patterns** are used to select sets of triples:
  - Example: ?y :rdf-type foaf:Person
  - This is an (*, p, o) pattern, ie., matches all subjects with the provided predicate and object
  - Binds all subjects to the variable **?y**

- A basic pattern is a set of triple patterns.
  - Returns triples that match all patterns (intersection)



```
SELECT * WHERE {
  ?a a foaf:Person ; foaf:knows ?b ; foaf:topic_interest ?ia .
  ?b a foaf:Person ; foaf:knows ?a ; foaf:topic_interest ?ib .
}
```





| ?a | ?b | ?ia | ?ib |
|---|---|---|---|
| :Alice | :Bob | :DB | :DB |
| :Alice | :Bob | :SW | :DB |
| :Bob | :Alice | :DB | :DB |
| :Bob | :Alice | :DB | :SW |

# GRAPH DATABASES

STORAGE

# Graph Data Structures

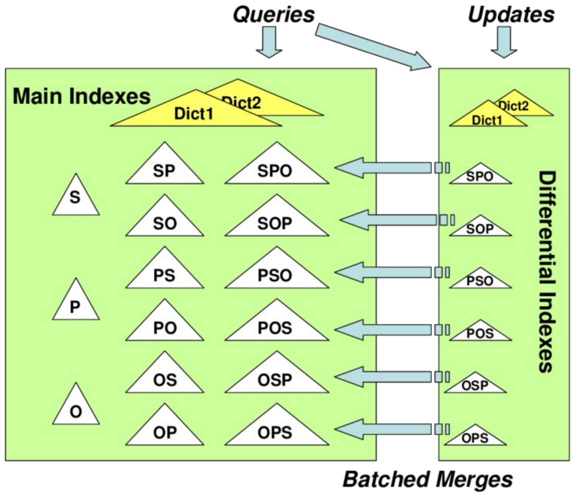
- Adjacency Matrix
  - Can use matrix operations / Linear algebra
  - Array stores
  - Inneficient storage (high sparsity)

- Adjacency lists
  - Very good for graph search
  - Document stores
  - Redundancy

- List of triples
  - Can use relational models (Triple Table, Property tables, Vertical partitioning)
  - Inner joins!

# Index-based Triple Stores

- SIX indexes (B+ trees):
  - All permutations of S, P, O
  - Single index scan
- Pros
  - Highly compressed
- Cons
  - Complex query execution engines



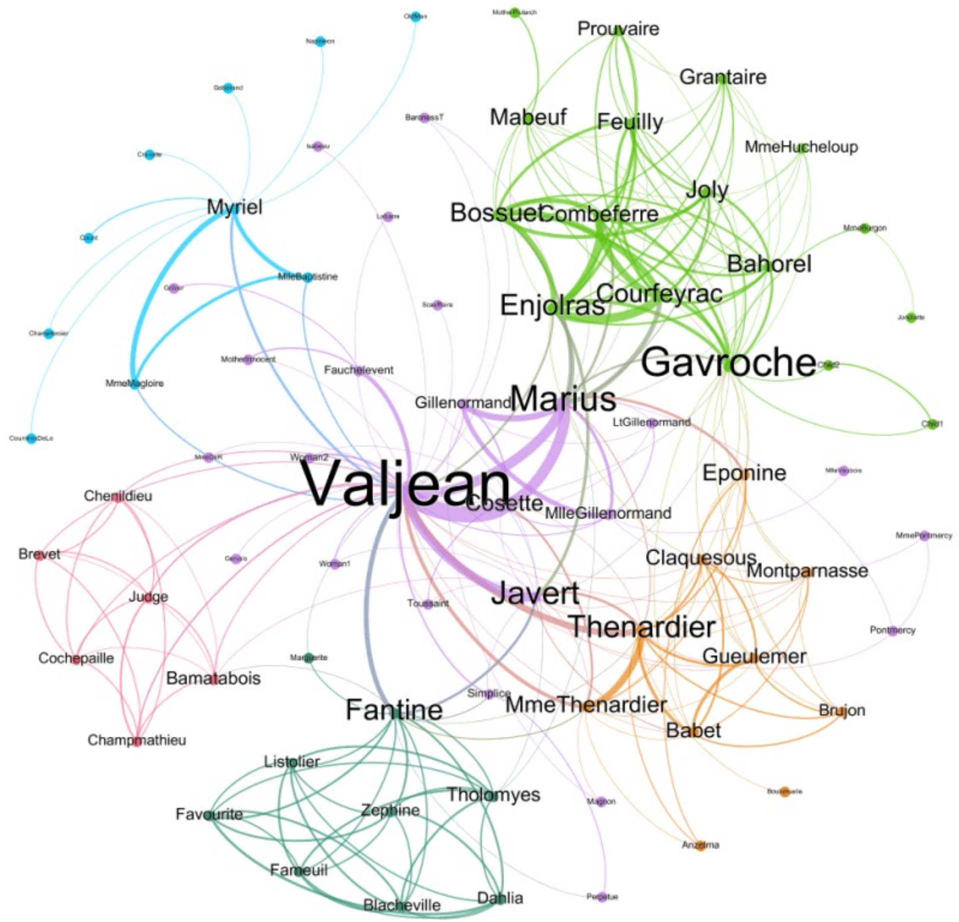
*RDF-3X*

- Systems:
  - **RDF-3X: "**The RDF-3X engine for scalable management of RDF data"
  - **Hexastore: "**Sextuple Indexing for Semantic Web Data Management"

# GRAPH DATABASES

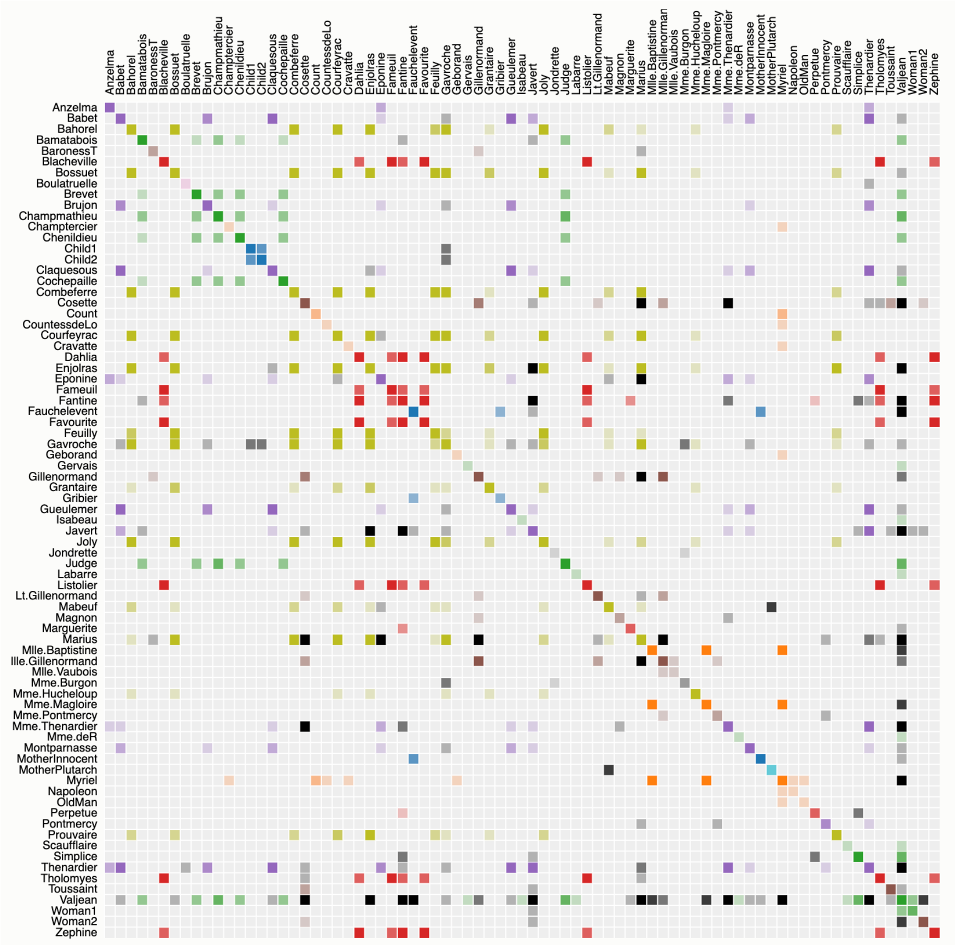
## PARTITIONING

# Partitioning



*Les Misérables* Character Co-occurrence Graph

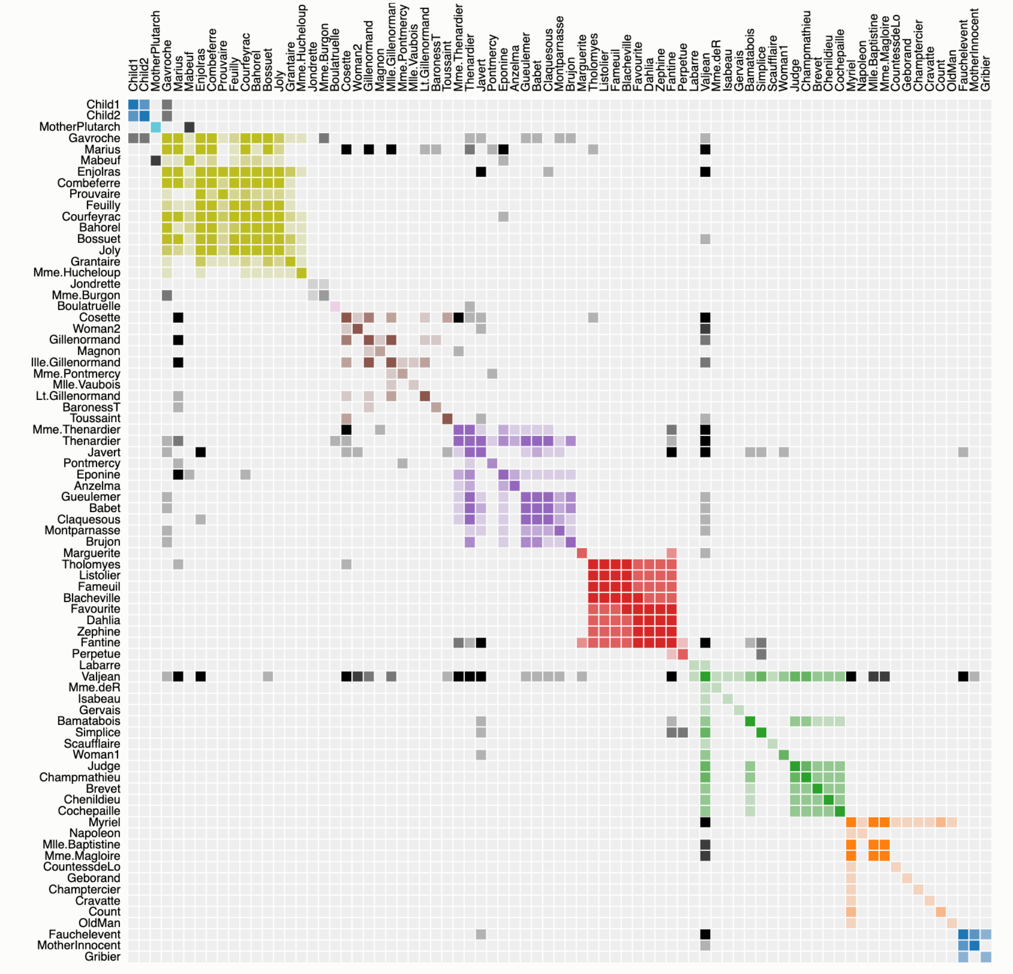
# Adjacency Matrix



Visualization Link

*Les Misérables* Character Co-occurrence Graph

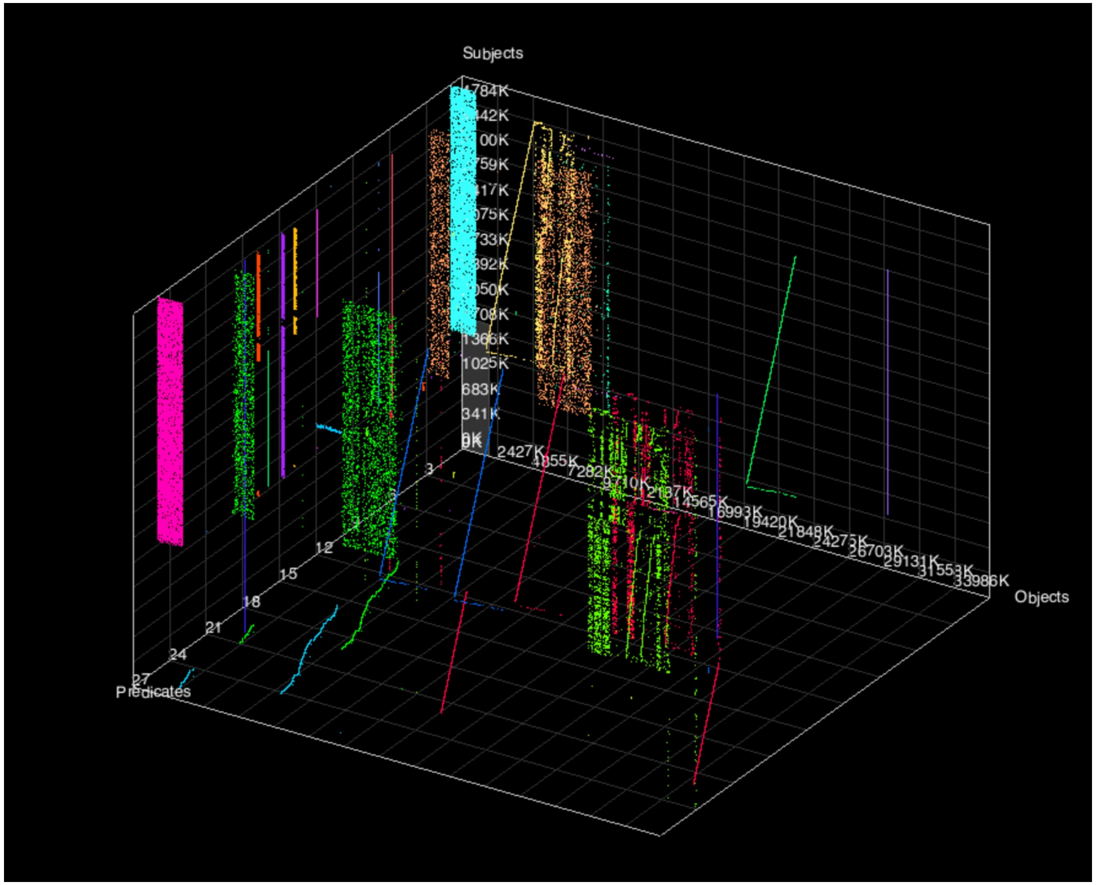
# Adjacency Matrix



*Les Misérables* Character Co-occurrence Graph

Visualization Link

# 3D Matrix



RDF HDT

# Agenda

- Graph DBMS
  - Introduction
  - Storage and Indexing
  - Partitioning

- Array DBMS
  - Introduction
  - Physical Array Storage
  - Space filling curves

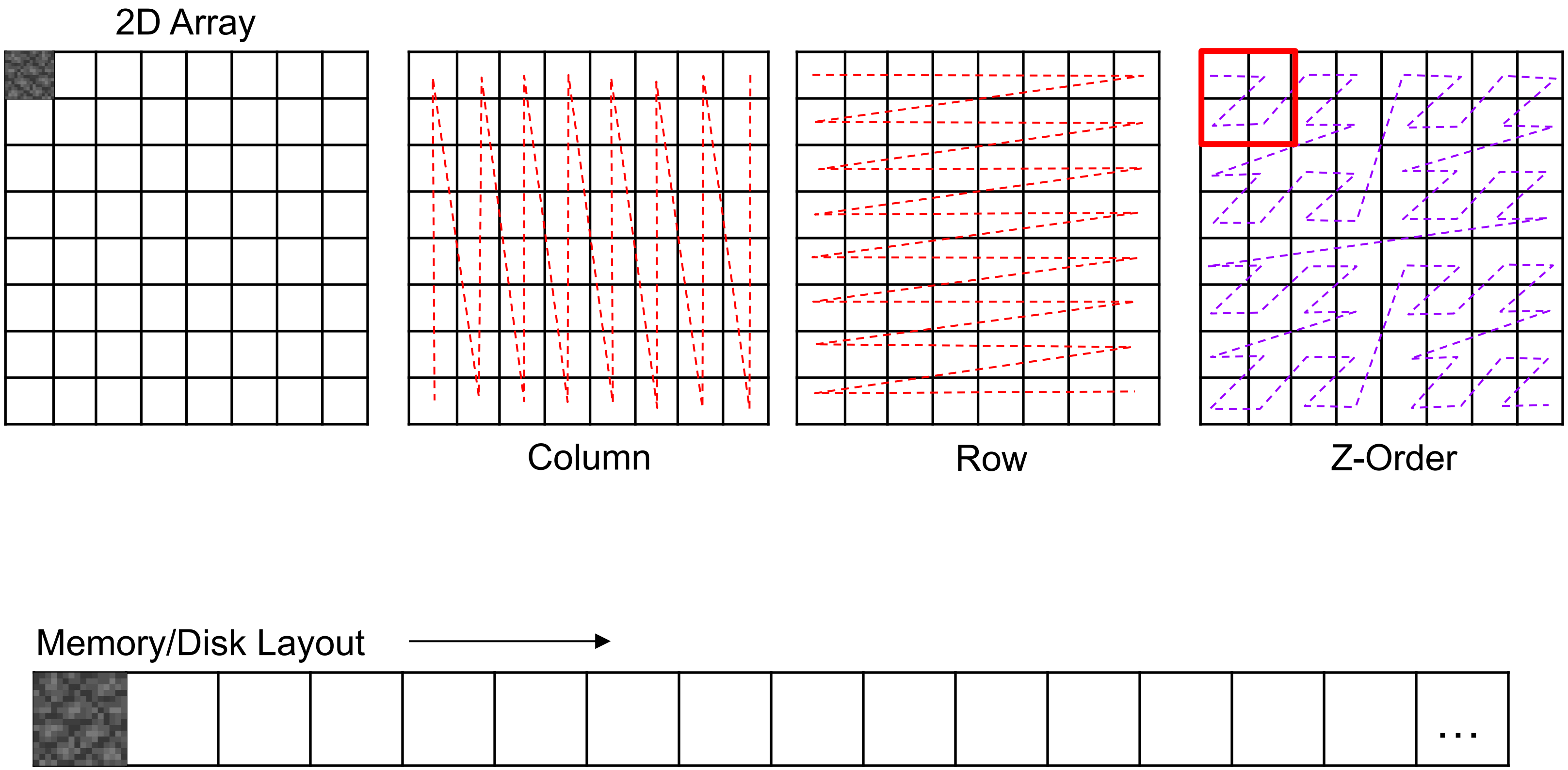
# Multi-dimensional Arrays

- Scientific data is often in the form of multi-dimensional arrays (matrices).
  - Arrays of images, video frames, or various scientific data

- Typical workload
  - Mostly analytical: slice, dice, aggregate
  - Algorithms: anomaly detection, object detection, etc.

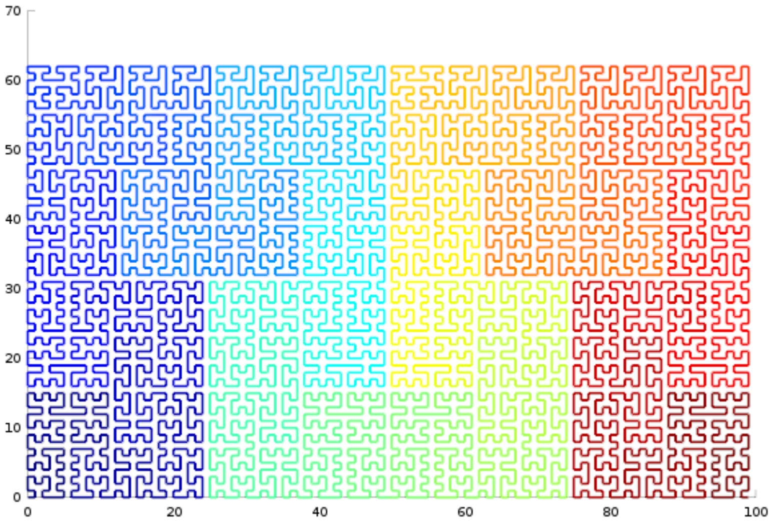
- Industry data file format: HDF5 (Hierarchal Data Format)

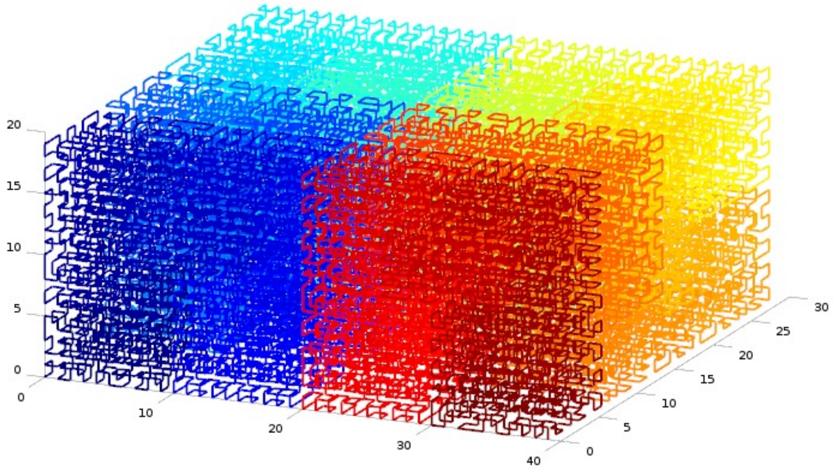
# Physical organization of an Array



# Space Filling Curves

- A space-filling curve is a mathematical curve that passes through every point in a 2D (or higher) dimensional space.
- Maps 1D line into a high-dimensional space (and vice versa)
  - Using mathematical formulae
- Examples: Hilbert-curve (images below*) / Z-Order / Peano-curve





*https://github.com/jakubcerveny/gilbert

# Agenda

- Graph DBMS
  - Introduction
  - Storage and Indexing
  - Partitioning

- Array DBMS
  - Introduction
  - Physical Array Storage
  - Space filling curves

- Q&A
- Midterm this Thursday during class time.