

Graph Databases

ANNA QUERALT, OSCAR ROMERO

(FACULTAT D'INFORMÀTICA DE BARCELONA)

Graph Databases

A (native) graph database:

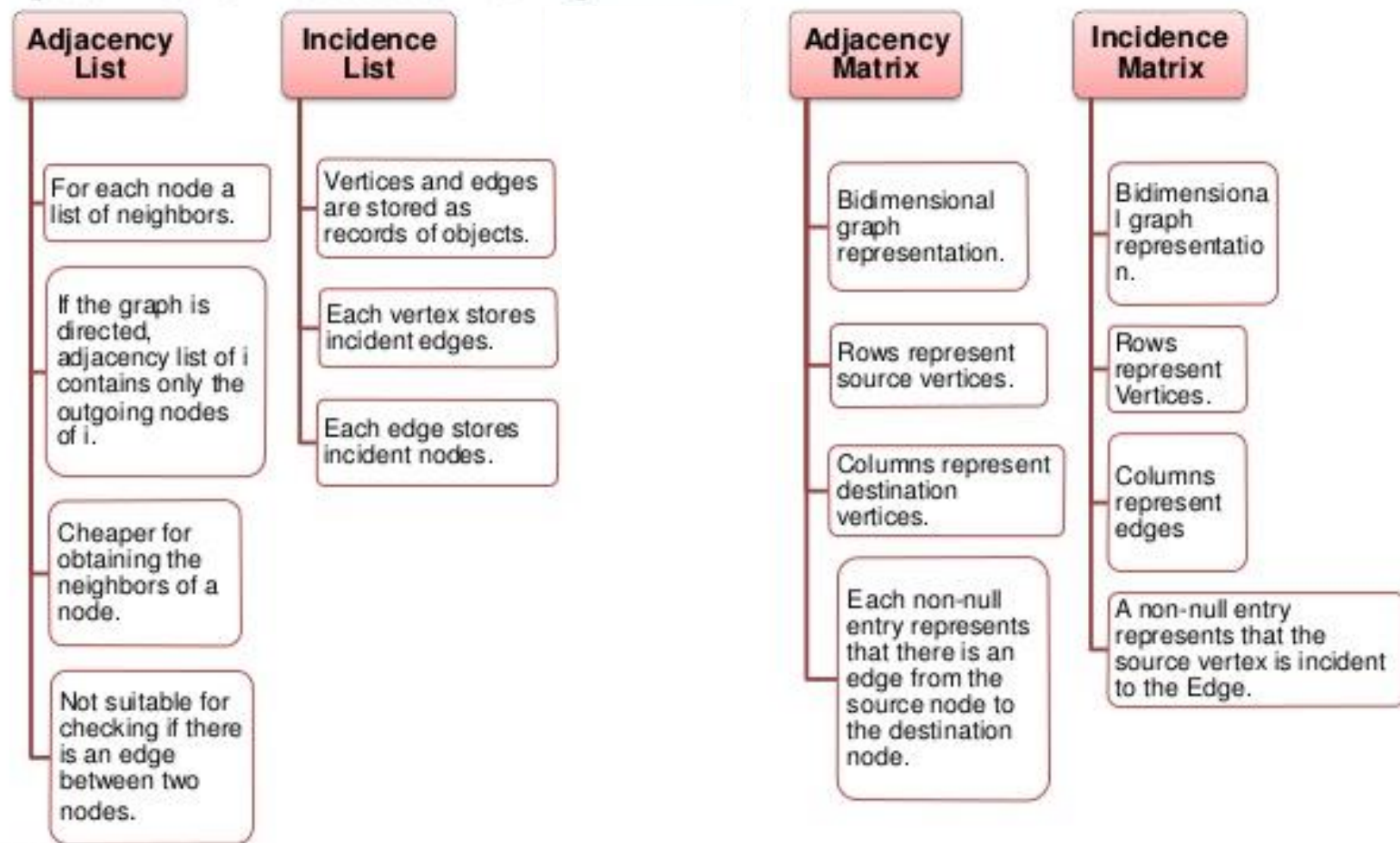
- Provides means to efficiently **process** graph data
 - **Index-free adjacency**
 - Note this does not diminish the possibility to plug a storage system with an external processing system
- Provides means to **store** graph data
 - Each having potentially different physical graph data models
- Examples: Neo4j, Titan

(Distributed) Graph frameworks typically refer to processing frameworks. Thus, like MapReduce or Spark, provide means to extract data from databases **BUT DO NOT STORE GRAPHS**

- Examples: Pregel (Google), Giraph (MapReduce), GraphX (Spark), ...

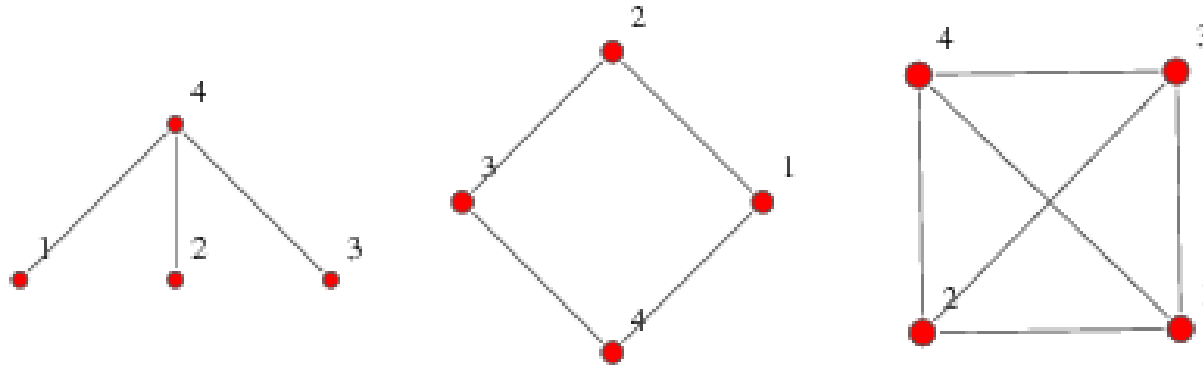
Implementation of Graphs

[Sakr and Pardede 2012]



Implementation of Graphs

Adjacency matrix (baseline)



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

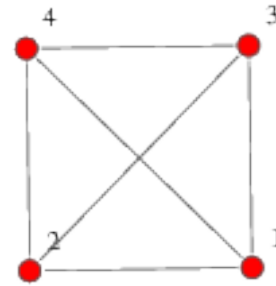
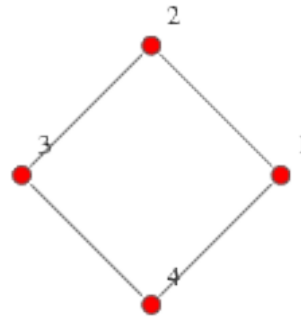
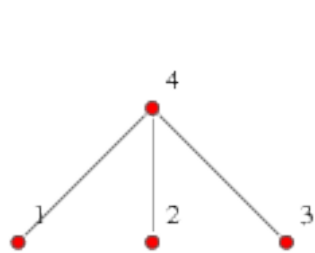
$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Activity

Objective: Understand the different structures needed to implement graphs following the main graph implementation strategies

- *Implement the following graphs as an adjacency list **AND** as an incidence list*



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Implementation of Graph Databases

NEO4J

Neo4J native graph storage

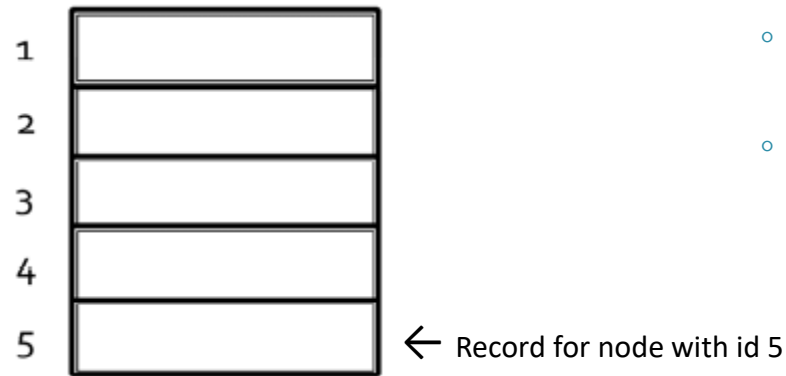
Based on **incidence lists**

- Implemented by means of singly and doubly linked list structures

Separate files for each different part of the graph

- Nodes
- Relationships
- Properties
- Labels
- Values

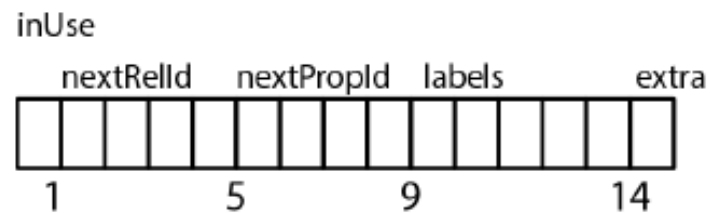
Incidence Lists – Neo4J



Nodes

- One physical file to store all nodes (in-memory, *Least Frequently Used* cache policy)
- Fixed-size** record: 15 bytes in length
 - Fast look-up: $O(1)$

Node (15 bytes)



- Each record is as follows:
- Byte 1 (metadata; e.g., in-use?)
- Bytes 2-5: id first relationship
- Bytes 6-9: id first property
- Bytes 10-14: labels
- Byte 15: extra information

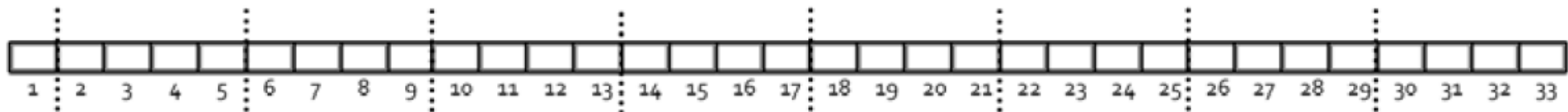
Incidence Lists – Neo4J

Relationship and property files.

- Both contain records of **fixed size**
- Cache with *Least Frequently Used* policy

Relationship file

- Contents of each record:
 - Metadata
 - id starting node, id end node
 - id label
 - ids of the previous and following relationship of the starting node and of the ending node
 - id first property
- Doubly linked list



Incidence Lists – Neo4J

Relationship and property files.

- Both contain records of **fixed size**
- Cache with *Least Frequently Used* policy

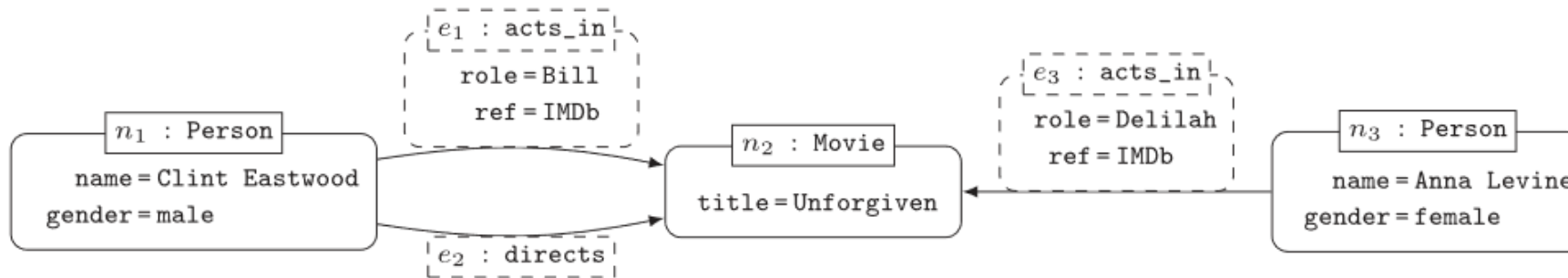
Property file

- A single file for all properties, regardless if they belong to nodes or to edges.
- Contents of each record:
 - Metadata (incl. a bit determining whether it belongs to an edge or node)
 - id node / edge
 - id of the following property of the node / edge
 - id property name
 - id property value
- Singly linked list

Activity

Objective: Understand how to implement a linked list to implement graphs

Consider the graph below. What would be the resulting data structures if you create such graph in Neo4J?



Types of graph databases

Types of Graph Databases

Some graph databases / processing frameworks are based on strong assumptions that are not explicit

- As a consequence of how they implement internal structures

Operational graphs:

- Map to the concept of a CRUD database
 - Nodes, edges can be deleted, updated, inserted and read
 - Example: Neo4j, Titan, OrientDB, Amazon Neptune, ...

Analytical graphs

- They are *snapshots* that cannot be modified by the final user
 - Equivalent to a data warehouse for graphs
 - Example: Sparksee, Giraph, GraphX, etc.

Summary

Unfortunately, there is no standard (yet) to implement graph databases

They all follow the same principles, but the way to implement it really affects graph processing

When choosing a graph database, consider:

- Operational vs. Analytical graph database
- Internal data structures
- Impact of the internal data structures on the required graph processing for your project