Graph Processing

ANNA QUERALT, OSCAR ROMERO

(FACULTAT D'INFORMÀTICA DE BARCELONA)

Basic Graph Operations

Adjacency $\text{Adjacency}(n) = \bar{N}$

$n_i \in \bar{N} \iff \exists e_1 \mid \rho(e_1) = (n_i, n) \vee \rho(e_1) = (n, n_i)$

Reachability $\text{Reachability}(n_{or}, n_{dest})$ is **true** $\iff \exists \text{Walk}(n_{or}, n_{dest})$

$\text{Walk}(n_{or}, n_{dest}) = (e_1 \dots e_m) \mid \exists n_1 \dots n_{m-1}, \rho(e1) = (n_{or}, n_1), \rho(e2) = (n_1, n_2)$

$\dots \rho(e_m) = (n_{m-1}, n_{dest})$

- Label-constrained reachability:
  - *x*, *y* can be variables, nodes, or a mix $x \xrightarrow{\alpha} y$
  - α is a regular expression over *Lab*

Pattern Matching

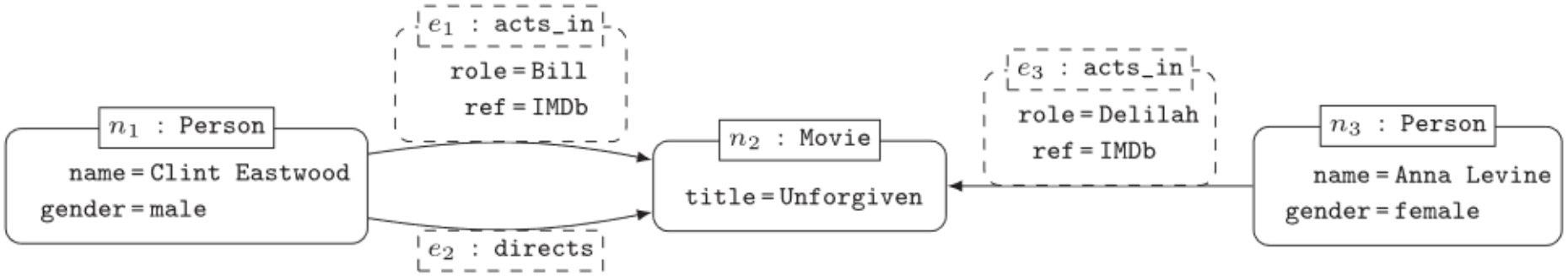
- Based on Basic Graph Patterns (bgps)
  - A bgp is a property graph where variables can appear in place of any constant

Activity

Objective: Understand the relationships between the basic graph operations

Answer the following questions:

- Is adjacency subsumed by reachability?
- Is adjacency subsumed by pattern matching?
- Is reachability subsumed by pattern matching?



Cost of Graph Operations

Note that the operations presented are conceptual, i.e. **agnostic of the graph db implementation**

Theoretical complexity(*):

- Adjacency:
  - **O(|V|)** (linear, #edges to visit)

- Label-constrained reachability:
  - ~ **O(|V|)** for a single pair
  - ~ **O(|V|²)** for all pairs

- Pattern matching: **NP-complete**
  - Navigational pattern matching: **NP-complete**
    - $O(|V|^3)$ with bounded simulation algorithms

(*) See additional materials in LearnSQL for details

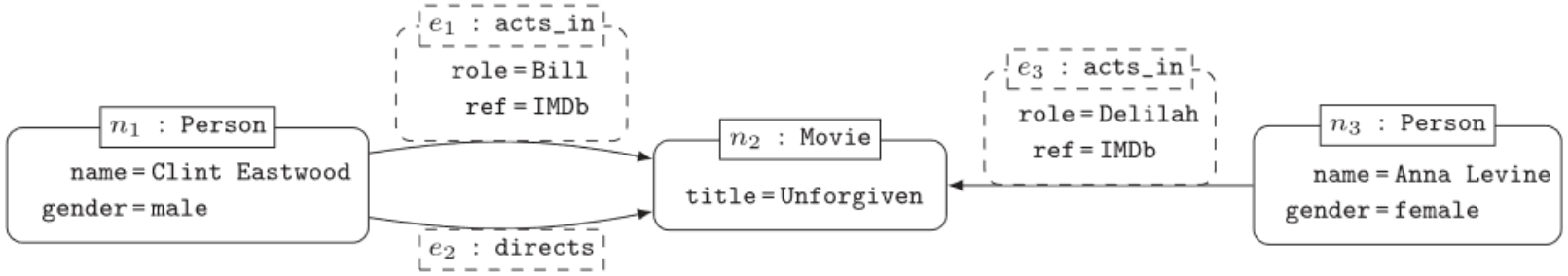
Activity

Objective: Identify the most efficient algorithm to solve a given query

Assume a graph containing relationships and nodes like the ones shown below (the graph may contain more nodes/eges, possibly with different labels)

Define (in the form of a pattern):

- A query that should be solved as an Adjacency problem
- A query that should be solved as a Label-constrained Reachability problem
- A query that should be solved as a (Navigational) Pattern Matching problem



Taking advantage of adjacency: Example

EUDAT project

Collaborative EUDAT Data Infrastructure
Data shared and preserved across borders and disciplines

Goal: analyse collaborations between authors

Main types of nodes:

- Data objects, Authors, Disciplines, Institutions

Setting:

- 3 graphs of different sizes: 7k, 10k, 440k Data objects
- Up to 630k nodes and 5.4M edges

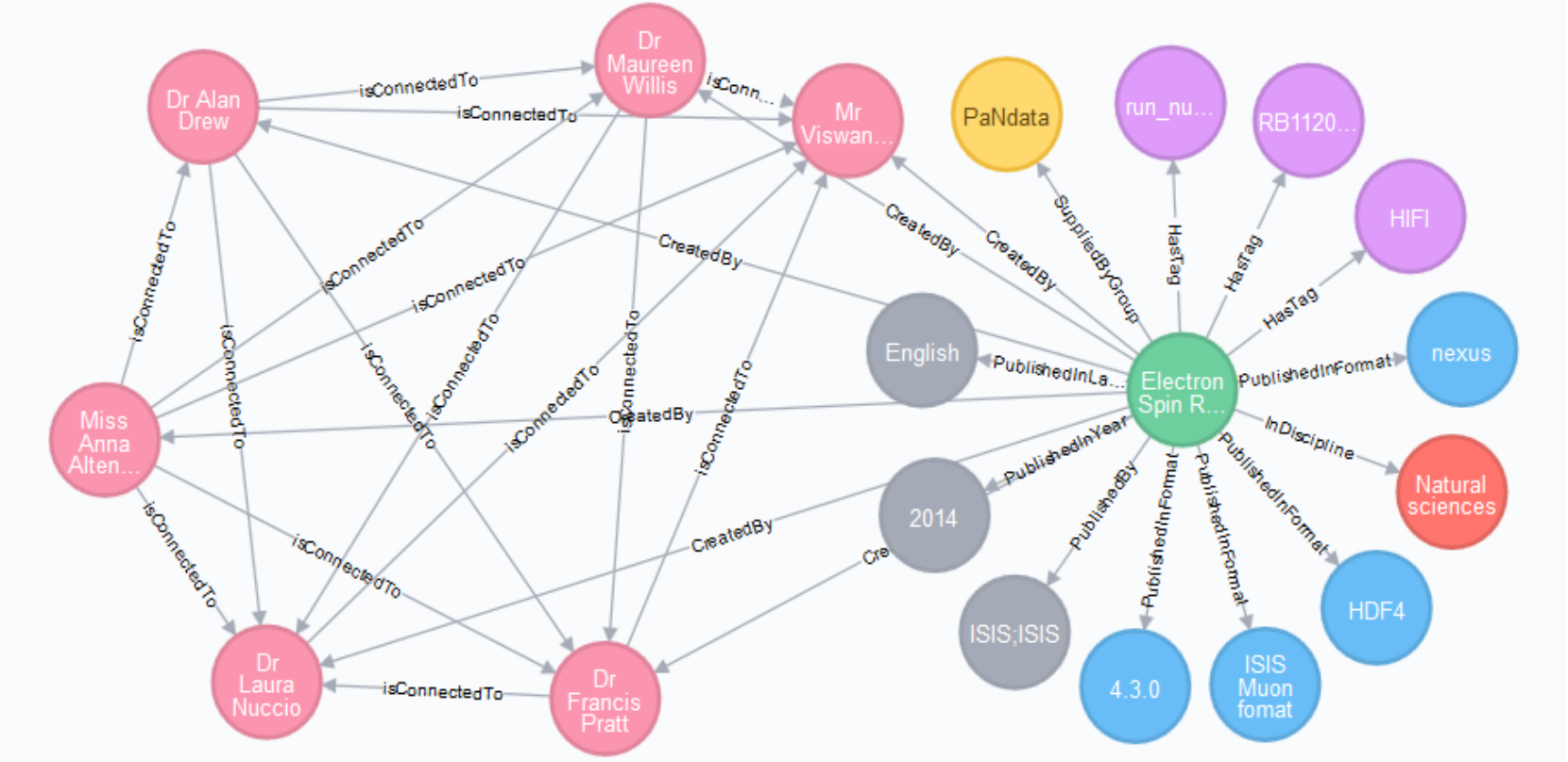
2 main types of queries:

- Local: “my collaborators” (*adjacency*), “how can I contact Person X” (*reachability*)
- Global: “strong collaborators” *(pattern matching),* “avg. number of collaborators” (*graph metrics*)

Taking advantage of adjacency: Example

Adding inferred “isConnectedTo” relationships (authors at distance 3 or less)

- 5.14M edges added



5.14 M edges added

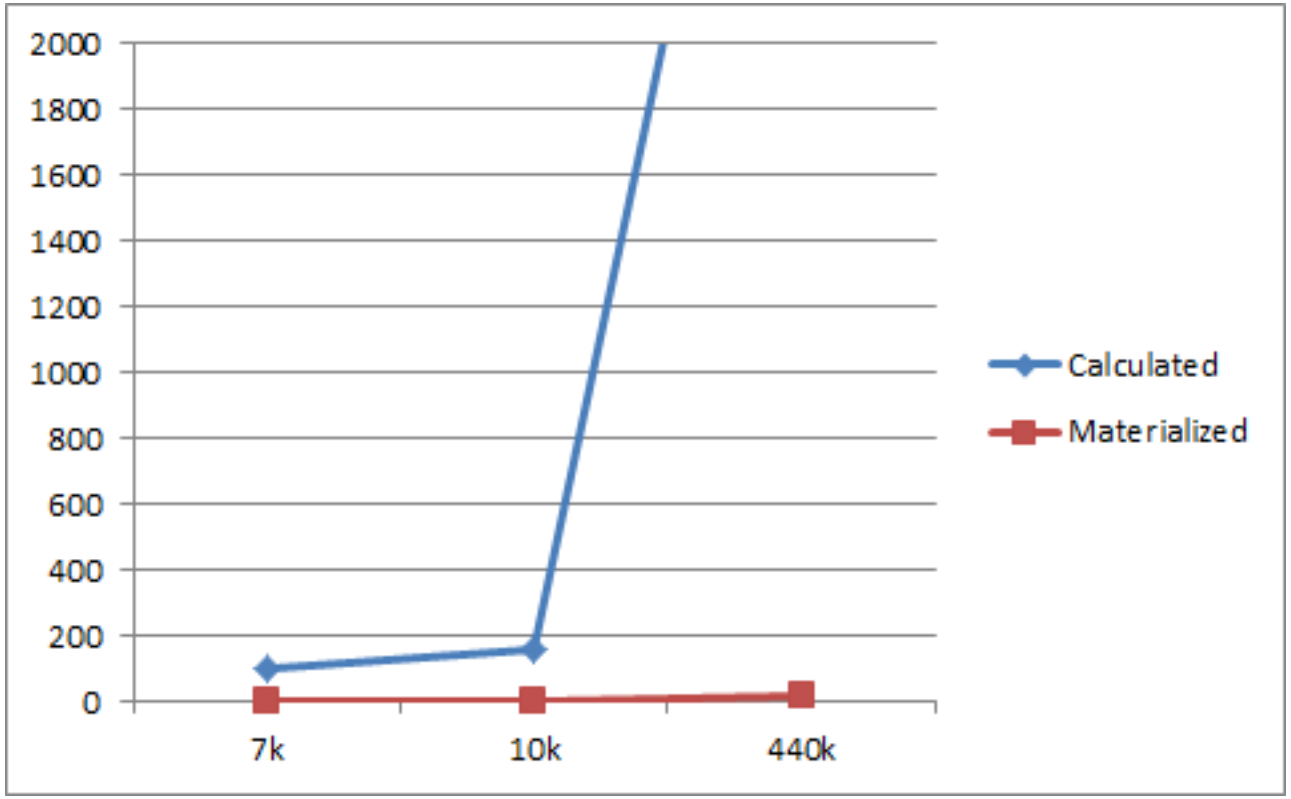
Creation in 2 steps:

- Add “isCoAuthorOf”
- Add “isConnectedTo”

Taking advantage of adjacency: Example

Benefits vs costs of the materialization:

- Finding collaborators at distance <=3 becomes feasible
- Implies maintaining the inferred edges when a new data object is added (1-2 seconds)



Complex graph processing

Graph Metrics

They can be defined as combinations of adjacency, reachability, pattern matching and complex graph patterns

- Given their relevance, they are typically provided as built-in functions

Computational cost: The cost depends on the specific metric

Examples:

- the min / max degree in the graph
- the graph diameter
- the graph density / sparsity
- betwenness of a node
- the pageRank of a node
- …

Graph Metrics

In this course, we will focus on the following algorithms:

- Centrality algorithms: determine the importance of nodes in a graph
  - Page Rank
  - Betweenness
  - Closeness
- Community detection algorithms: evaluate how groups of nodes are clustered or partitioned, as well as their tendency to strengthen or break apart
  - Triangle counting
  - Louvain
  - (Strongly) connected components
- Similarity algorithms: compute the similarity of pairs of nodes based on their neighborhoods or their properties
  - Node similarity
- Path finding algorithms: find the path between two or more nodes
  - Dijkstra's shortest path

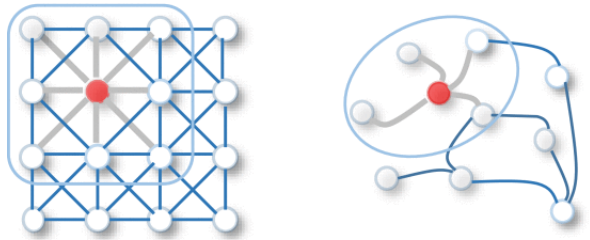
Complex Graph Processing

Different approaches (depending on the goal):

- **Graph Processing Pipelines,** consisting of:
  - Basic operations: Adjacency, Reachability, Pattern Matching
  - Complex graph patterns: Projection, Distinct, Grouping, Aggregations, Union
  - Graph Metrics: pre-defined algorithms

- **Graph Embeddings**:
  - Provide a vector representation of graphs that can be later processed by ML algorithms
  - Applications: Node classification, Node clustering, Link prediction, …

- **Graph Neural Networks:**
  - A particular case of neural networks for processing graph data
  - Applications: Recommendations, drug discovery, categorization, …



Z. Wu, et al. "A Comprehensive Survey on Graph Neural Networks," in *IEEE Trans. on Neural Networks and Learning Systems* 32(1), 2021.

Summary

Basic graph operations are not disjoint

- Adjacency and reachability can be expressed in terms of (navigational) pattern matching
- Pattern matching is a NP-complete problem

There are efficient algorithms that implement adjacency and reachability

- Could be used by the graph database to optimize queries

Different approaches for complex graph processing:

- Graph processing pipelines
- Graph embeddings
- GNNs