Graph Query Languages

Foundations of Graph Query Languages

- Declarative languages to query the graph
  - Typically, it matches to an extended version of pattern matching
  - For pattern matching, every current graph database engine chooses a fix semantics for it (i.e., either homomorphism or one of the three isomorphism-based interpretations)
- APIs providing implementation of graph metrics or (label-constrained) shortest-path
  - Depending on the metric or algorithm, it maps to adjacency, reachability or pattern matching

Types of Queries

- Graph databases distinguish certain types of queries, since each of them map to a different access plan:
  - Adjacency queries
    - Neighbourhood queries require accessing the basic data structure and navigate it (i.e., find a node and follow its edges)
  - Regular path queries (or navigational graph paterns)
    - Combine pattern matching and reachability: require specific graph-oriented algorithms
    - **It is still equivalent to conjunctive queries**
  - Complex graph patterns
    - Add further expressivity beyond conjunctive queries
    - Grouping / aggregations, set operations (union, difference, etc.) and joins (through attribute values)

Adjacency Queries

- Depend on the database structure
  - Time to find a node or an edge depends on how the graph data structures are implemented (thus, different performance for each database)
    - See the graph databases session
  - Once a node / edge is found, time to find its adjacent / incident neighbours

Navigational Graph Patterns

- *Navigational graph patterns (**NGP**s)* or *regular path queries (**RPQ**s)* refer to an extended algorithm typically implemented in graph databases that mixes pattern matching and reachability
- RPQs **extend** the bgp definition by allowing regular expressions on edges to describe path queries as part of the pattern. A path is described as: $x \xrightarrow{\alpha} y$ over G
  - X and y are nodes in G
  - α a regular expression over *Lab* (the set of labels in G)

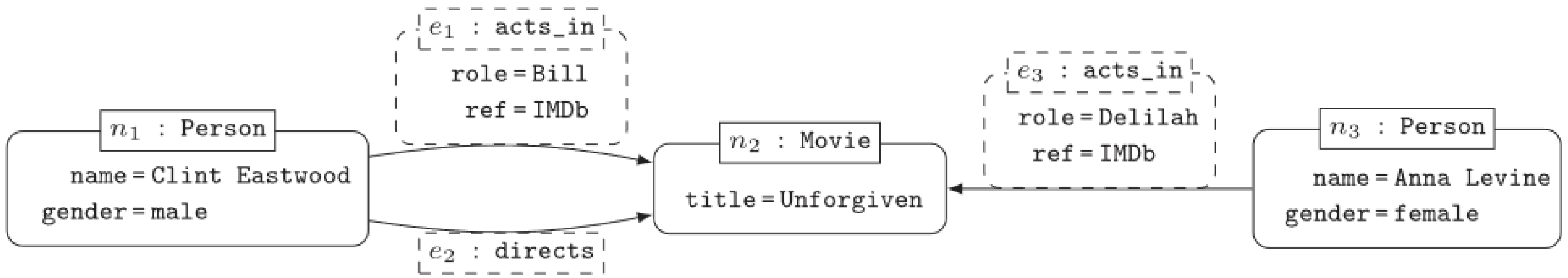
Path Queries

- The regular expressions evaluated differ from language to language. The most usual ones are:
  - (*) Kleene star and ($^+$) Kleene plus
  - ($\circ$) Concatenation
  - ($^-$) Inverse
  - (|) Union

  … and combinations of them

Activity

- *Objective: Understand how RPQs extend pattern matching*
- Assume a graph containing relationships and nodes like the ones shown below
  - Define a bgp including path queries from the previous slide to find *all co-actors of all actors*
  - On top of that, think of how to retrieve *all actors, starting from Clint Eastwood, you can reach by (transitivetely) following the co-actoring relationship, at least once*



Complex Graph Patterns

- RPQs are equivalent to conjunctive queries without projections (i.e., joins and equality selections)
- However, database languages (typically based on the relational algebra) are richer than that
- **GraphQL** was the first graph algebra extending RPQs with relational-like operators
  - [Projection] – subset of the variables
  - Union
  - Difference
  - Left-outer join / Optional
  - [Selection / Filter] – considering selections on properties
- GraphQL was the first formal graph language presented (2008) and included RPQs and complex graph patterns
  - They did so by introducing the concept of **graph motif**

He et al. Graphs-at-a-time: Query Language and Access Methods for Graph Databases. SIGMOD'08 (https://sites.fas.harvard.edu/~cs265/papers/he-2008.pdf)

Relevance of Complex Graph Patterns

- Querying a graph is reducible to querying a relational database (i.e., **non-recursive GraphQL is equivalent to the relational algebra**)
  - It opens the door to incorporate other analysis techniques on graphs. For example:
    - Graph warehousing: OLAP operations are known to be reducible to the relational algebra. Thus, it is possible to apply OLAP on top of graphs
    - Advanced data techniques that were reducible to the relational algebra (e.g., skyline queries)
    - Apply the data integration operations we saw in the graph data model session. For example:
      - Matching a subgraph to another subgraph
      - Link two independent graphs
      - Etc.

Navigational graph patterns

MOST POPULAR LANGUAGES

Cypher

- Created by Neo4j
  - Nowadays, standard de facto adopted by other graph databases (OpenCypher)
- High-level, declarative language
  - It is both a DDL and a DML
- Allows navigational graph patterns
  - However, it is quite limited when expressing regular path expressions
- It applies pattern matching under **no-repeated-edge isomorphism semantics**

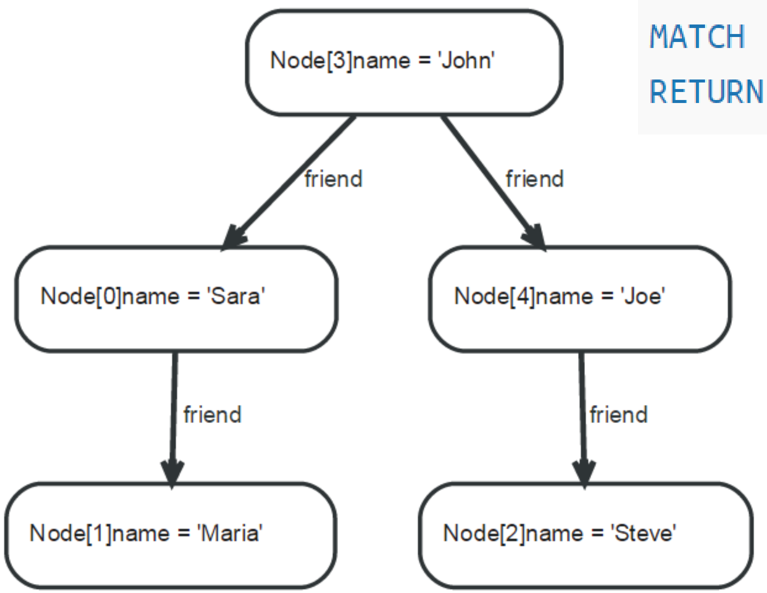
Neo4j Clauses

- Clauses:
  - DML:
    - MATCH: The graph pattern (bgp / ngp) to match
    - WHERE: Filtering criteria
    - WITH: Divides a query into multiple, distinct parts
    - RETURN: What to return
  - DDL:
    - CREATE: Creates nodes and relationships
    - DELETE: Removes nodes, relationships and properties
    - SET: Set values to properties
    - FOREACH: Performs updating actions once per element in a list

https://neo4j.com/docs/developer-manual/current/cypher/

Cypher DML

- Cypher applies a data pipeline, where each stage is a MATCH-WHERE-WITH/RETURN
  - It allows the definition of aliases to be passed between stages



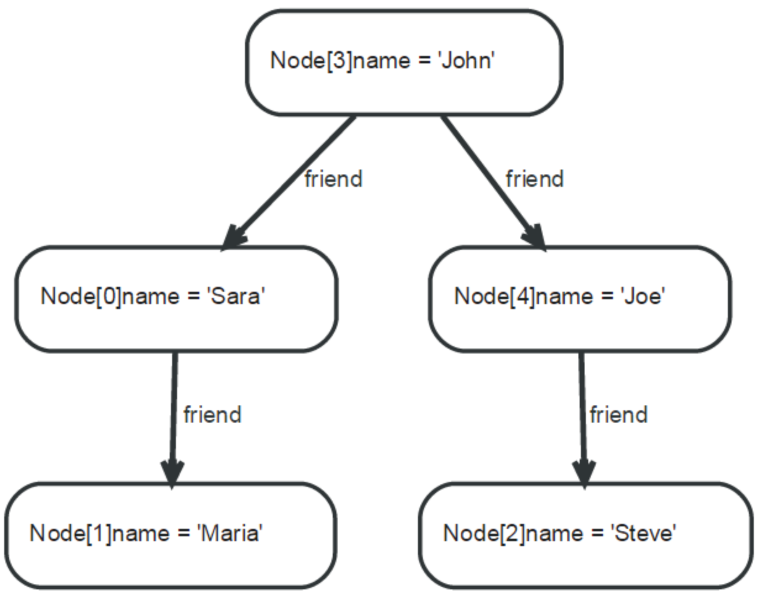

```
MATCH (john {name: 'John'})-[:friend]->()-[:friend]->(fof)
RETURN john, fof
```

john	fof
Node[3]{name:"John"}	Node[1]{name:"Maria"}
Node[3]{name:"John"}	Node[2]{name:"Steve"}

2 rows

Activity

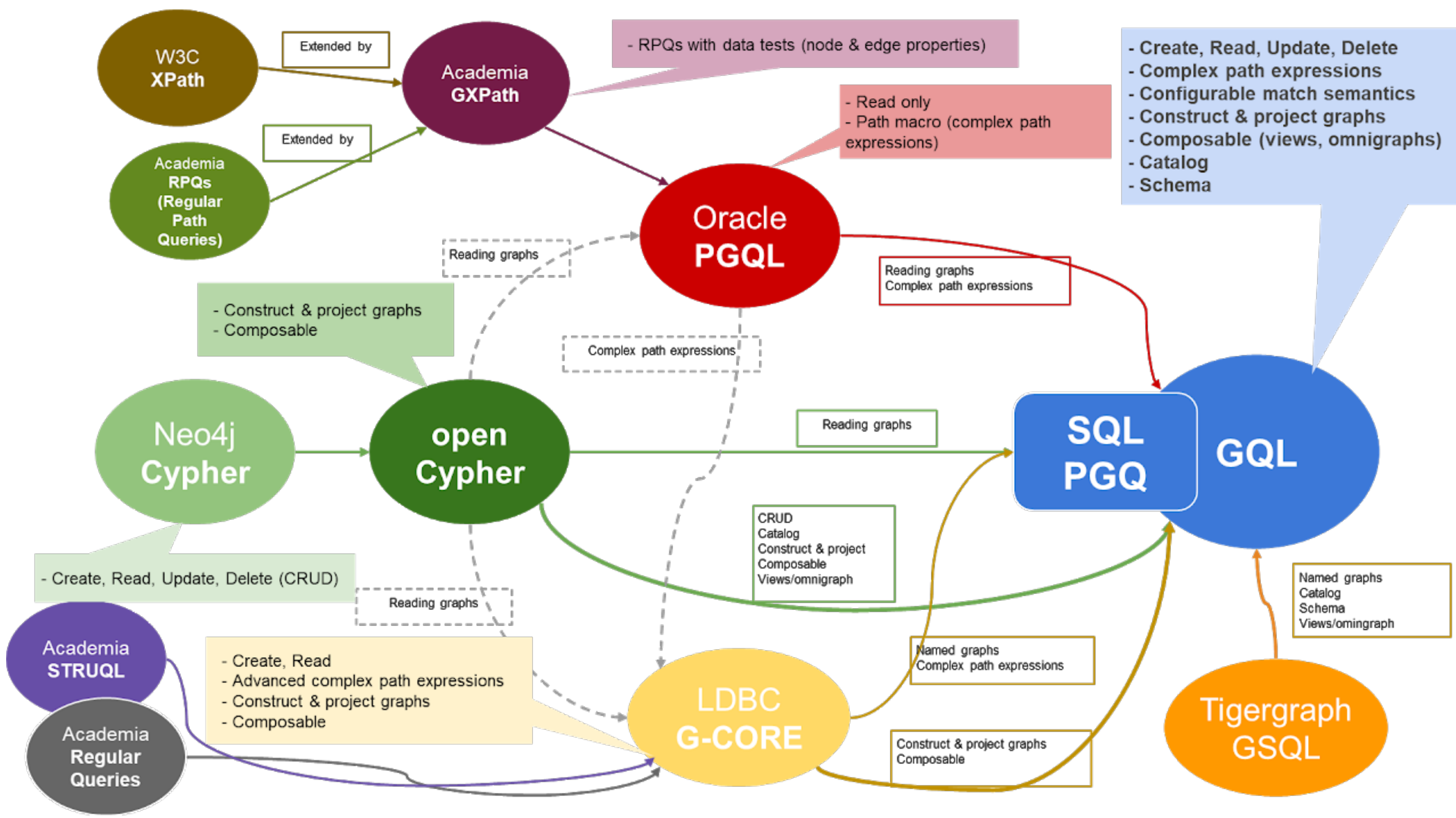
- *Objective: Basics on Cypher*
- Given the following graph, write the Cypher query for the next statements:



1) Return all nodes
2) Return all edges
3) Return all neighbour nodes of 'John'
4) Return the incident nodes of all edges

GQL

- There is currently a big effort towards standardization
- Graph Query Language: https://www.gqlstandards.org/



***RPQ stands for regular path queries

Summary

- Graph languages have been strongly formalized
  - Computational complexity deeply studied
- Navigational pattern matching as keystone
  - Pattern matching
  - Reachability
- Complex pattern matching
  - Formalized based on graph motifs
  - Extends navigational pattern matching with relational-like operators
- Complex pattern matching is necessary to unleash the power of graphs for data integration, OLAP or advanced data analytics
- Most popular languages
  - Cypher, Gremlin
  - Unfortunately, no standard for complex pattern matching yet