Advanced Data Management (CSCI 680/490)

Graph Databases

Dr. David Koop

Recent History in Databases

- Early 2000s: Commercial DBs dominated, Open-source DBs missing features
- Mid 2000s: MySQL adopted by web companies
- Late 2000s: NoSQL dos scale horizontally out of the box
- Early 2010s: New DBMSs that can scale across multiple machines natively and provide ACID guarantees

[A. Pavlo]

NewSQL

ScaleBase
Tokutek™
dbShards
ScaleArc
Akiban
ScaleDB
Schooner
JustOneDB
NuoDB
SQLFire
TransLattice
VoltDB
GenieDB
H-Store
xeround
Clustrix

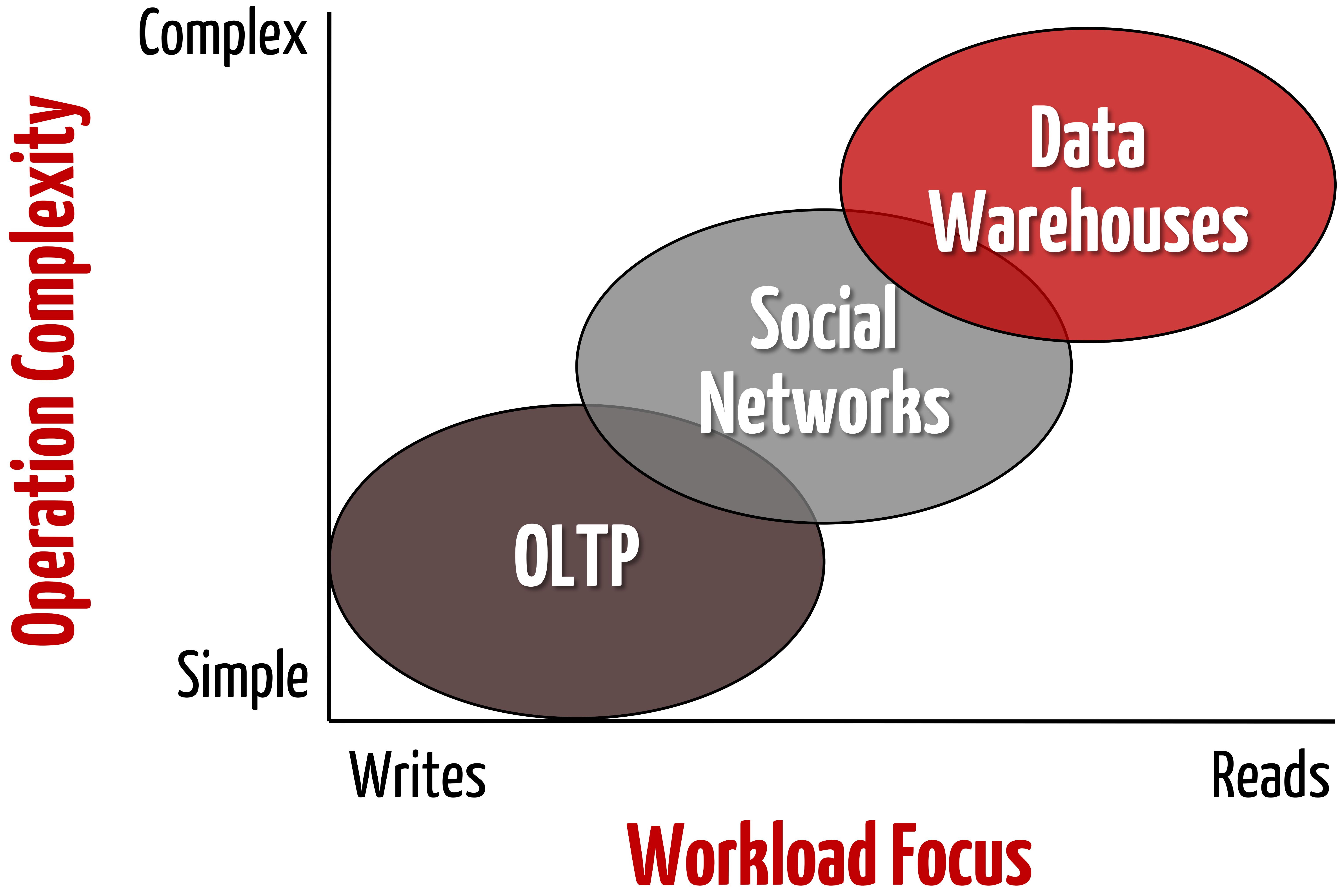
[A. Pavlo]

NewSQL

- 451 Group's Definition:
  - A DBMS that delivers the scalability and flexibility promised by NoSQL while retaining the support for SQL queries and/or ACID, or to improve performance for appropriate workloads.
- Stonebraker's Definition:
  - SQL as the primary interface
  - ACID support for transactions
  - Non-locking concurrency control
  - High per-node performance
  - Parallel, shared-nothing architecture

[A. Pavlo]

OLTP Workload



[A. Pavlo]

Ideal OLTP System

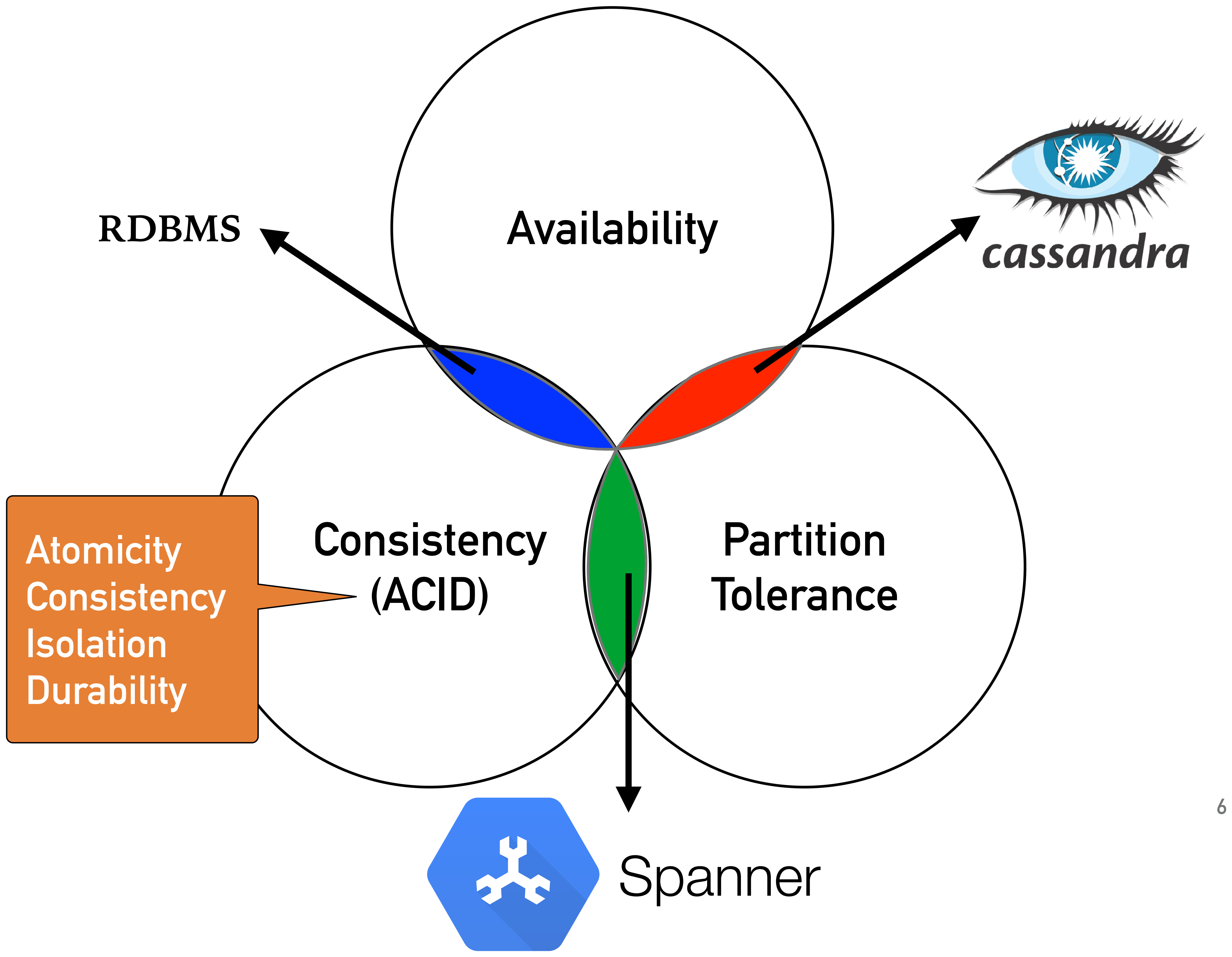
- Main Memory Only
- No Multi-processor Overhead
- High Scalability
- High Availability
- Autonomic Configuration

[A. Pavlo]

Spanner Overview

- Focus on scaling databases focused on OLTP (not OLAP)
- Since OLTP, focus is on sharding **rows**
- Tries to satisfy CAP (which is impossible per CAP Theorem) by not worrying about 100% availability
- External consistency using multi-version concurrency control through timestamps
- ACID is important
- Structured: universe with zones with zone masters and then spans with span masters
- SQL-like (updates allow SQL to be used with Spanner)

Spanner and the CAP Theorem



- Which type of system is Spanner?
  - C: consistency, which implies a single value for shared data
  - A: 100% availability, for both reads and updates
  - P: tolerance to network partitions
- Which two?
  - CA: close, but not totally available
  - So actually **CP**

External Consistency

- Traditional DB solution: **two-phase locking**—no writes while client reads
- "The system behaves as if all transactions were executed sequentially, even though Spanner actually runs them across multiple servers (and possibly in multiple datacenters) for higher performance and availability" [Google]
- Semantically indistinguishable from a single-machine database
- Uses multi-version concurrency control (MVCC) using **timestamps**
- Spanner uses **TrueTime** to generate monotonically increasing timestamps across all nodes of the system

Google Cloud Spanner: NewSQL

	CLOUD SPANNER	TRADITIONAL RELATIONAL	TRADITIONAL NON-RELATIONAL
Schema	✓ **Yes**	✓ Yes	✗ No
SQL	✓ **Yes**	✓ Yes	✗ No
Consistency	✓ **Strong**	✓ Strong	✗ Eventual
Availability	✓ **High**	✗ Failover	✓ High
Scalability	✓ **Horizontal**	✗ Vertical	✓ Horizontal
Replication	✓ **Automatic**	Configurable	Configurable

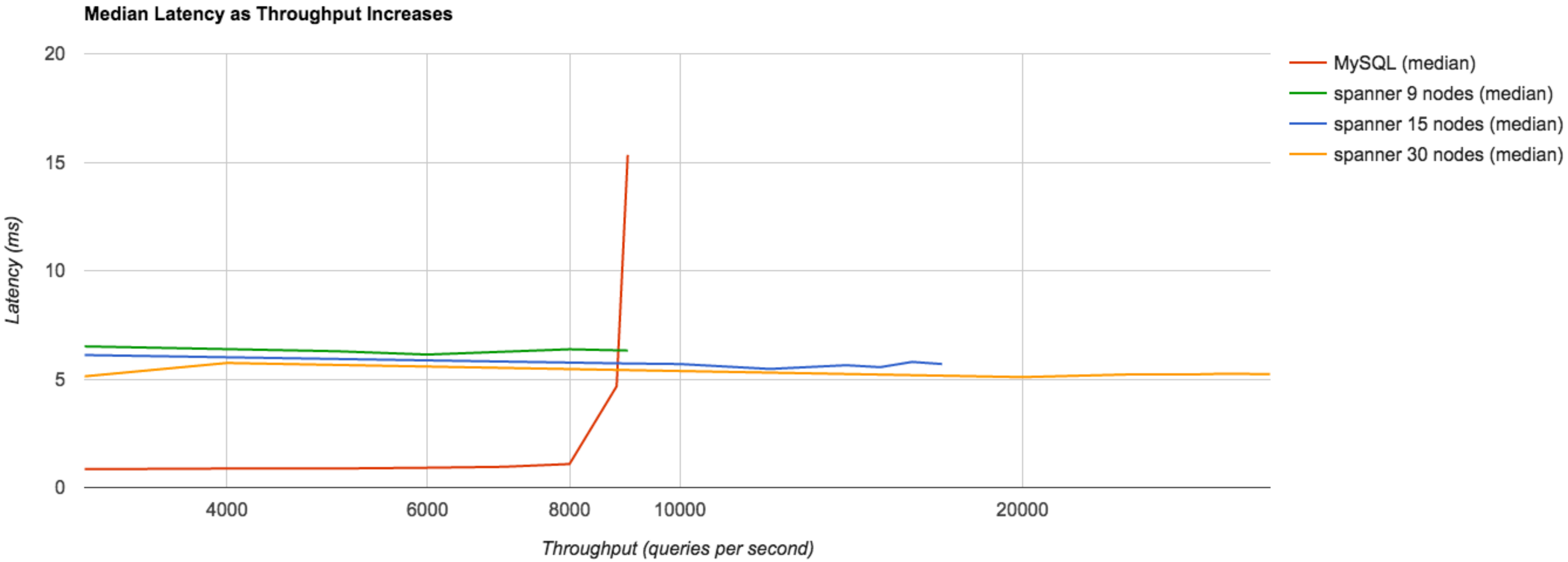
[https://cloud.google.com/spanner/]

Spanner as "Effectively CA"

- Criteria for being "effectively CA"
  1. At a minimum it must have very high availability in practice (so that users can ignore exceptions), and
  2. As this is about partitions it should also have a low fraction of those outages due to partitions.
- Spanner meets both of these criteria
- Spanner relies on Google's **network** (private links between data centers)
- TrueTime helps create **consistent snapshots**, sometimes have a commit wait

[E. Brewer, 2017]

Throughput: Spanner vs. MySQL



[P. Bakkum and D. Cepeda, 2017]

Assignment 4

- Work on Data Integration and Data Fusion
- Integrate artist datasets from different institutions (The Met, The Tate, Smithsonian, Carnegie Museum of Art)
  - Integrate information about names, places, nationality, etc.
- Record Matching:
  - Which artists are the same?
  - Which nationalities are the same? (British/English)
- Data Fusion:
  - Year of birth/death differences
  - Nationality differences

Test 2

- Wednesday, April 6
- Covers material from the beginning of course, emphasizing material since Test 1
- Similar Format to Test 1
- We have discussed more **papers** since Test 1

Specific Types of Data

Graphs: Social Networks

[P. Butler, 2010]

What is a Graph?

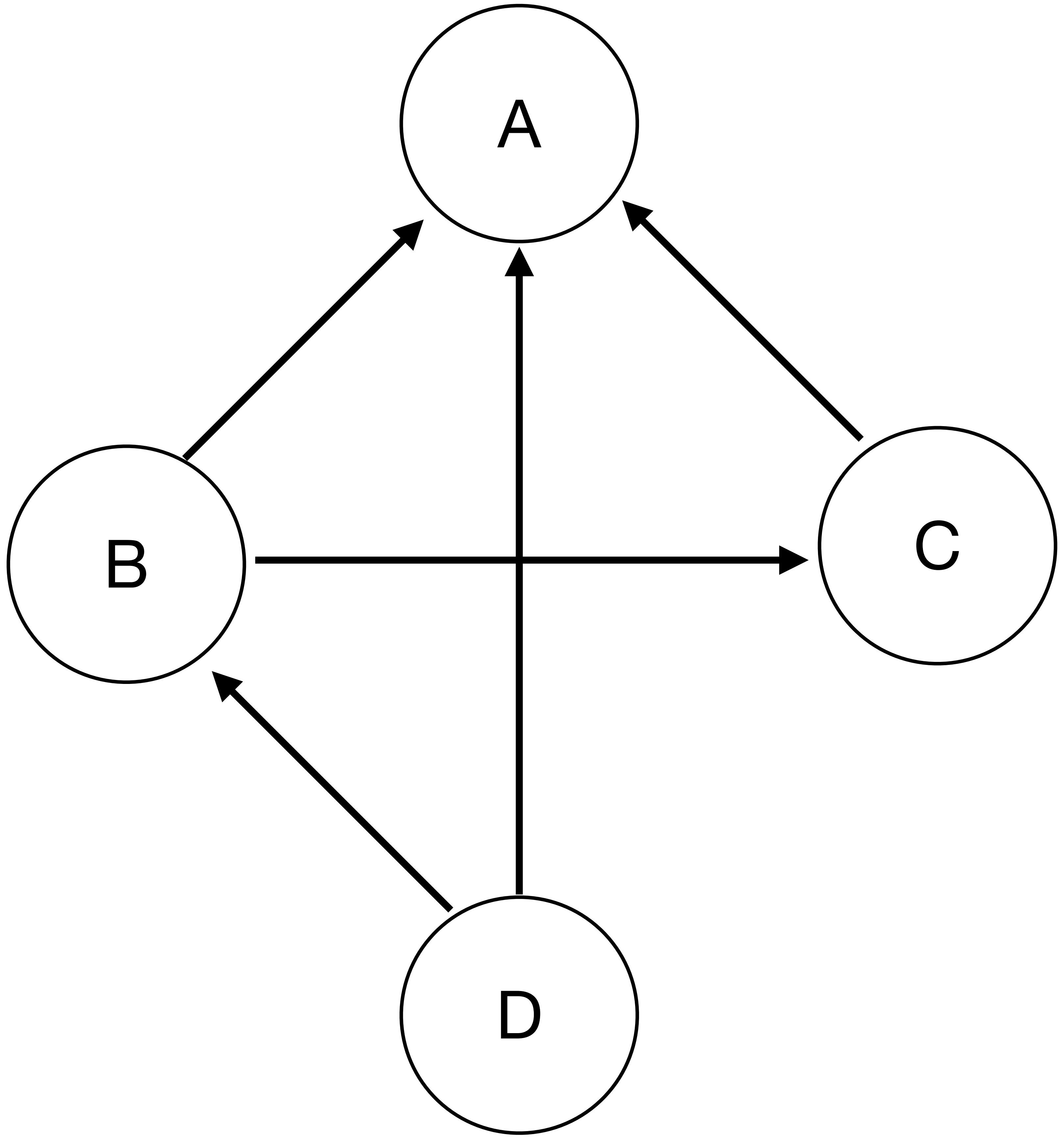
- An abstract representation of a set of objects where some pairs are connected by links.

Object (Vertex, Node)

Link (Edge, Arc, Relationship)

[M. De Marzi, 2012]

What is a Graph?

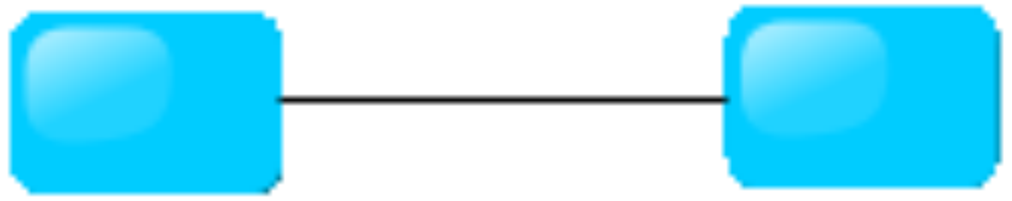


- In computing, a **graph** is an abstract **data structure** that represents set objects and their relationships as **vertices** and **edges/links**, and supports a number of graph-related **operations**
- Objects (nodes): `{A,B,C,D}`
- Relationships (edges): `{(D,B),(D,A),(B,C),(B,A),(C,A)}`
- Operation: shortest path from `D` to `A`

[K. Salama, 2016]

Different Kinds of Graphs

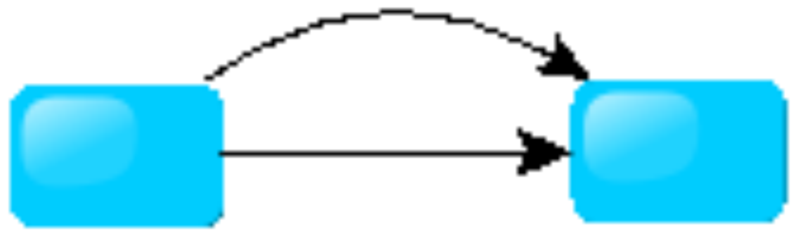
- Undirected Graph



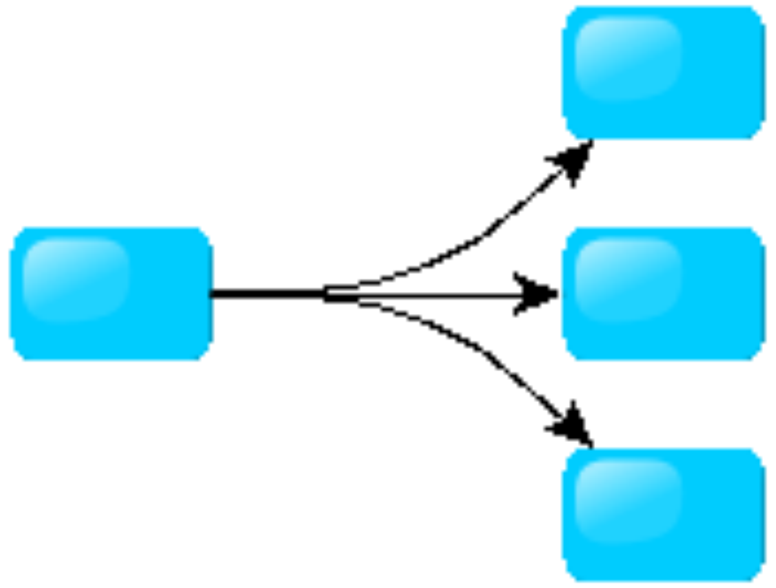
- Directed Graph

- Pseudo Graph

- Multi Graph



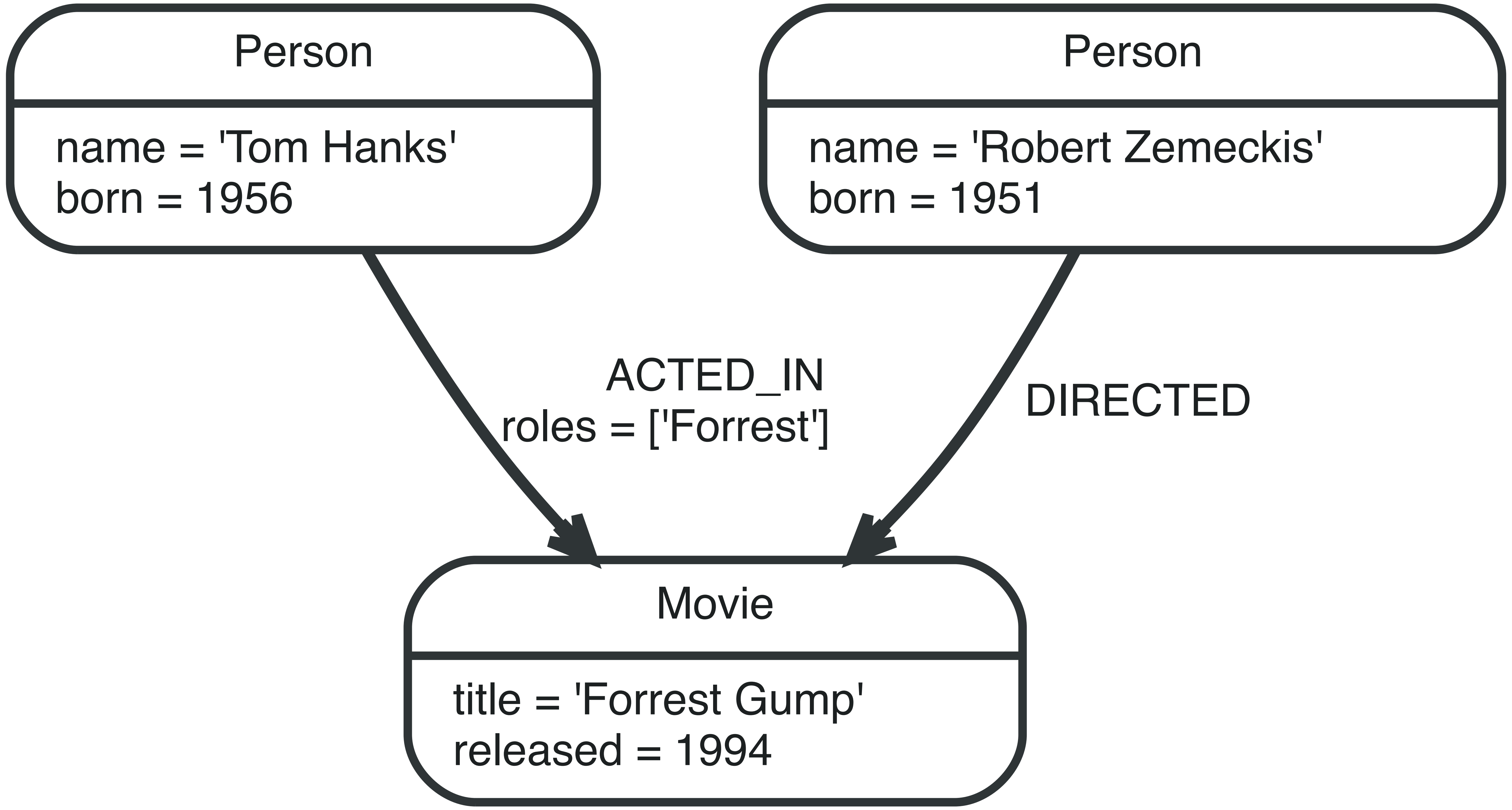
- Hyper Graph



[M. De Marzi, 2012]

Graphs with Properties

- Each vertex or edge may have properties associated with it
- May include identifiers or classes



[neo4j]

Types of Graph Operations

- Connectivity Operations:
  - number of vertices/edges, in- and out-degrees of vertices
  - histogram of degrees can be useful in comparing graphs
- Path Operations: cycles, reachability, shortest path, minimum spanning tree
- Community Operations: clusters (cohesion and separation)
- Centrality Operations: degree, vulnerability, PageRank
- Pattern Matching: subgraph isomorphism
  - can use properties
  - useful in fraud/threat detection, social network suggestions

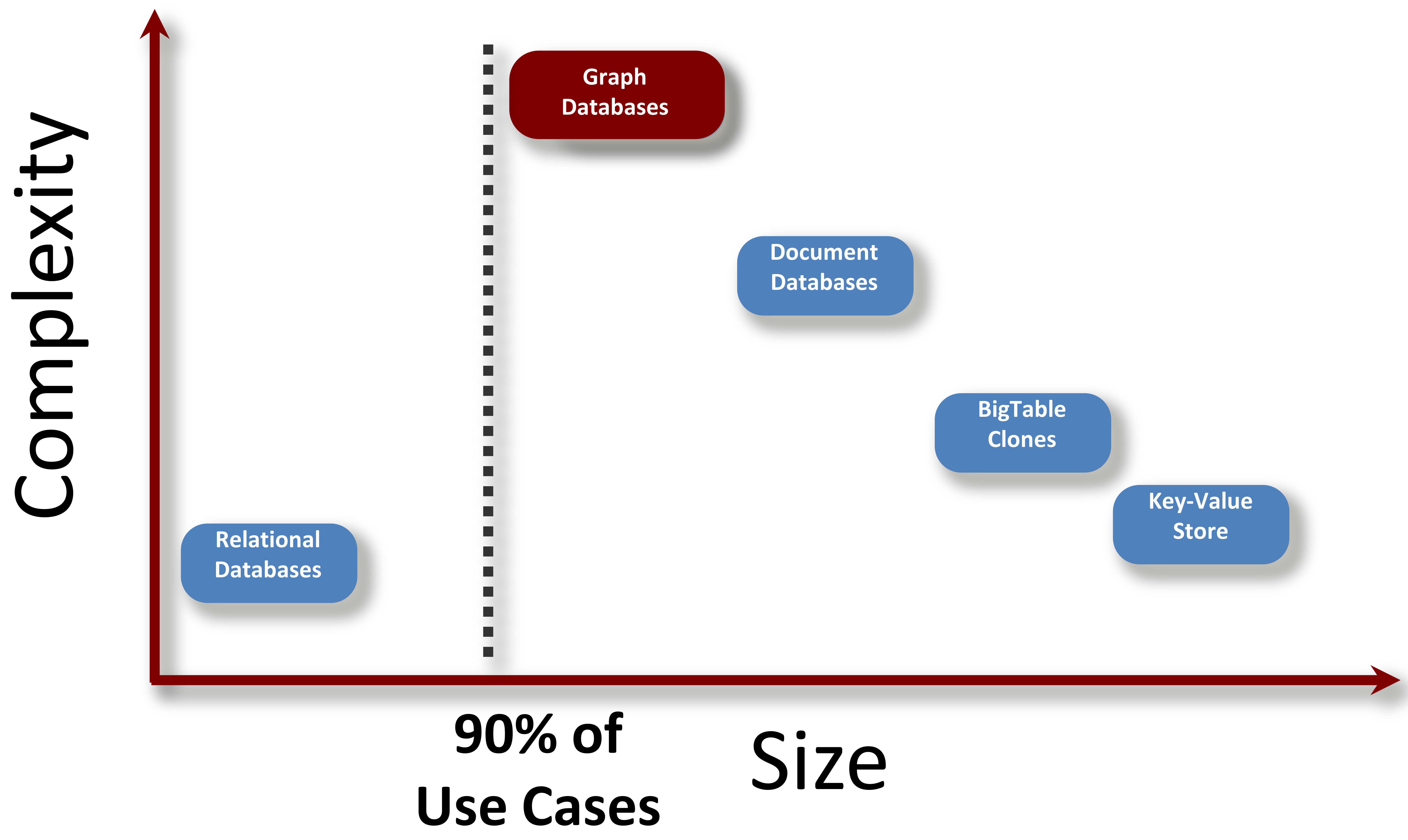
[K. Salama, 2016]

What is a Graph Database?

- A database with an explicit graph structure
- Each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Plus an Index for lookups

[M. De Marzi, 2012]

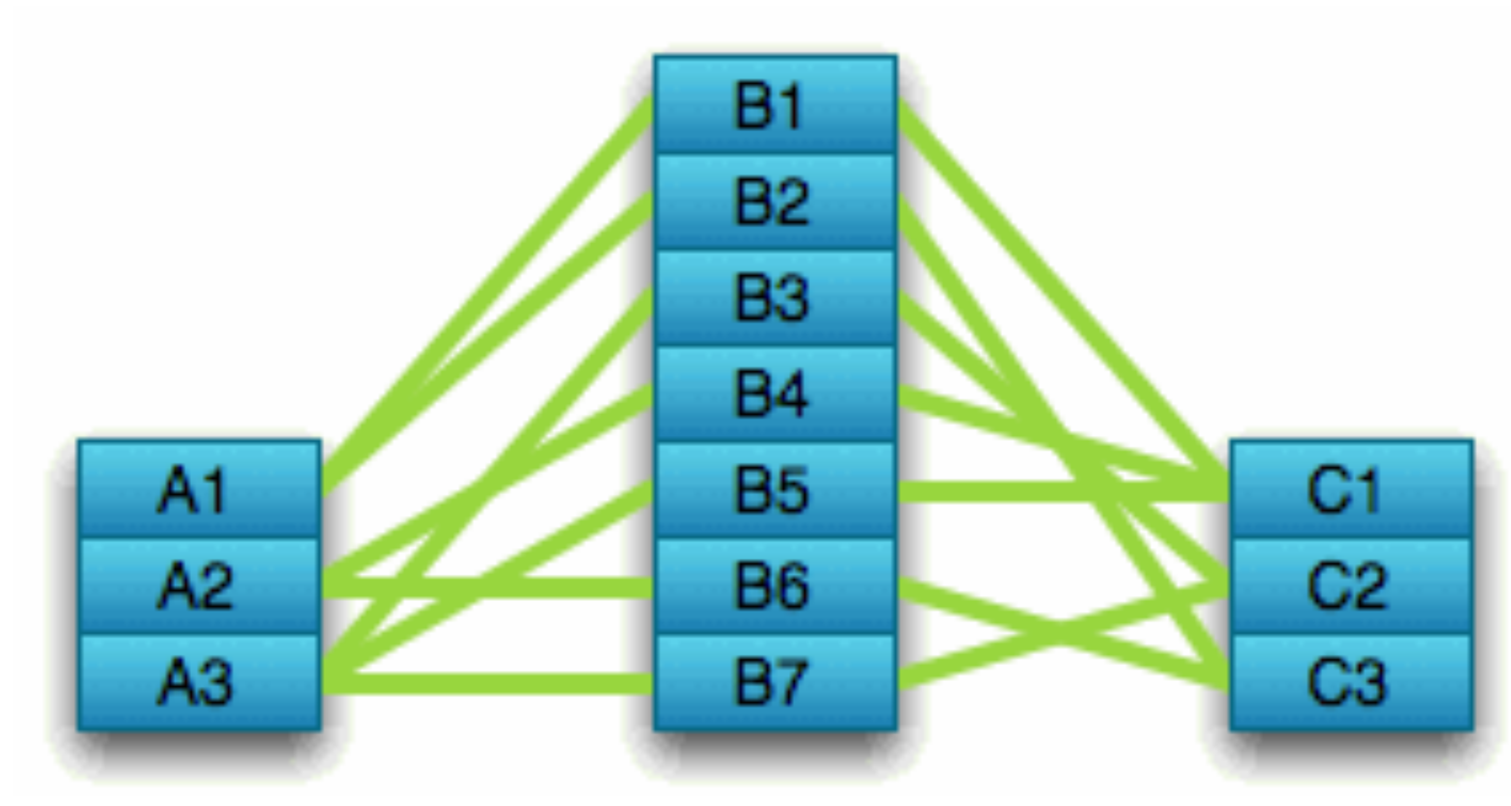
How do Graph Databases Compare?



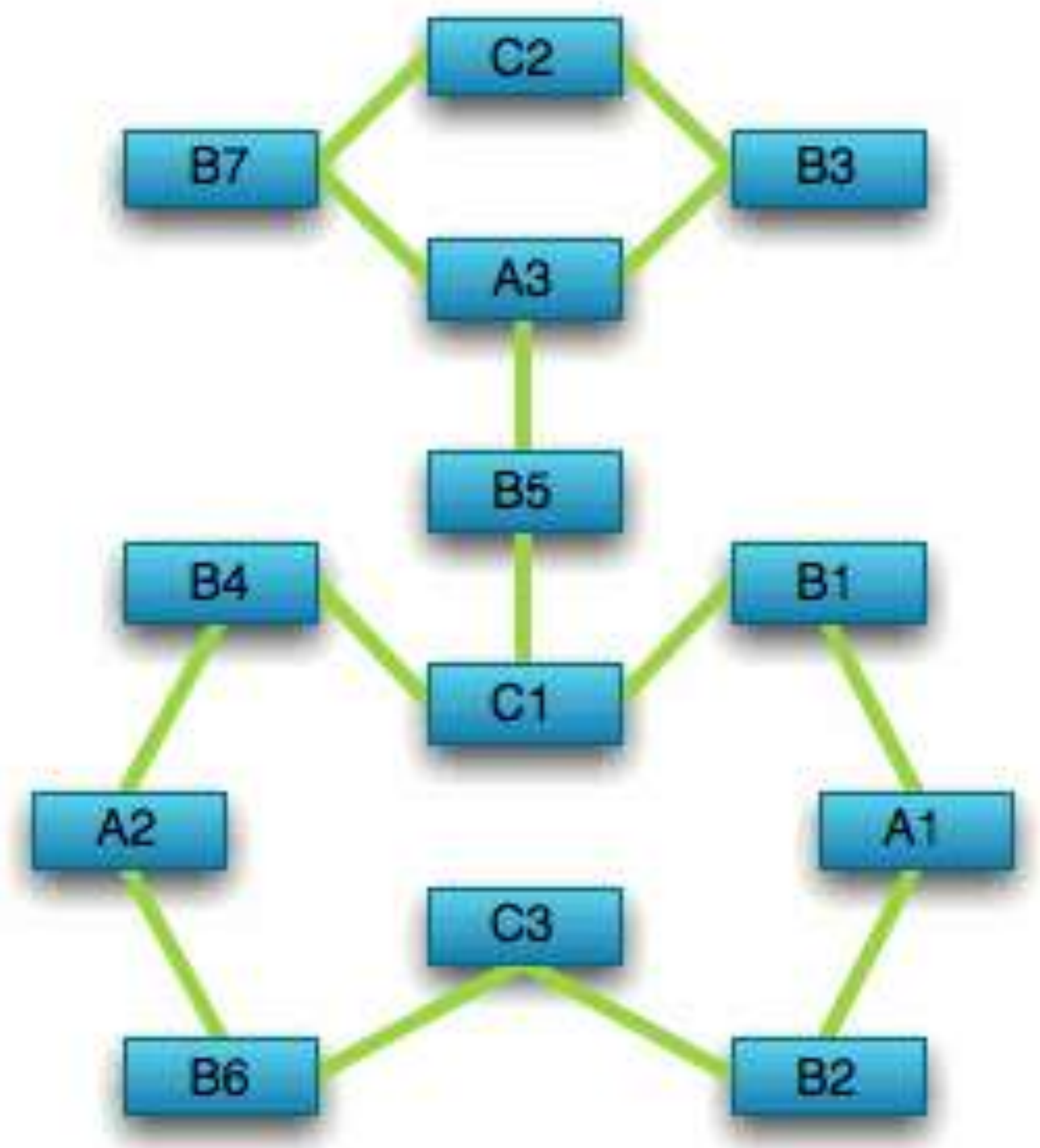
[M. De Marzi, 2012]

Graph Databases Compared to Relational Databases

Optimized for aggregation



Optimized for connections

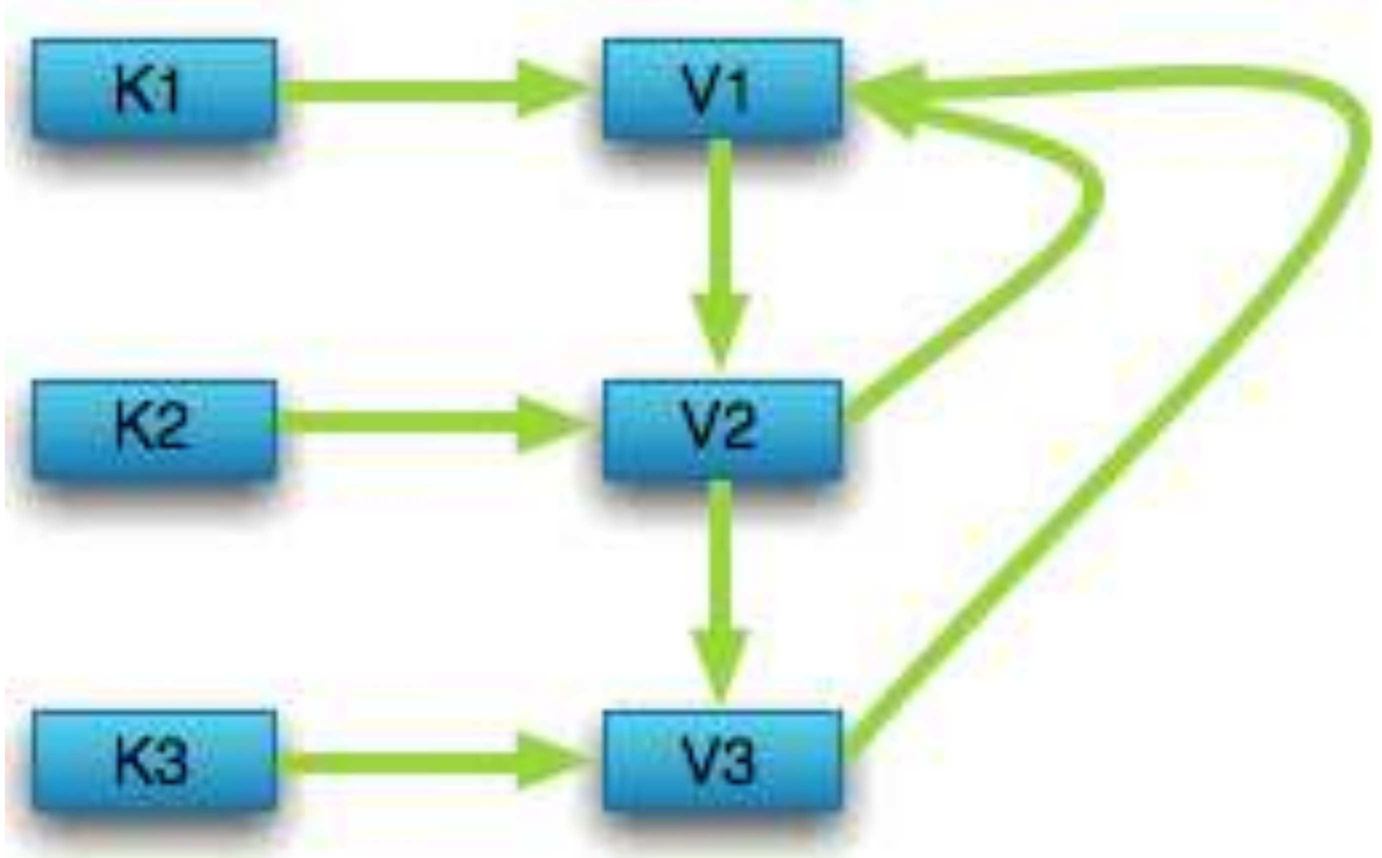


[M. De Marzi, 2012]

Graph Databases Compared to Key-Value Stores

Optimized for simple look-ups

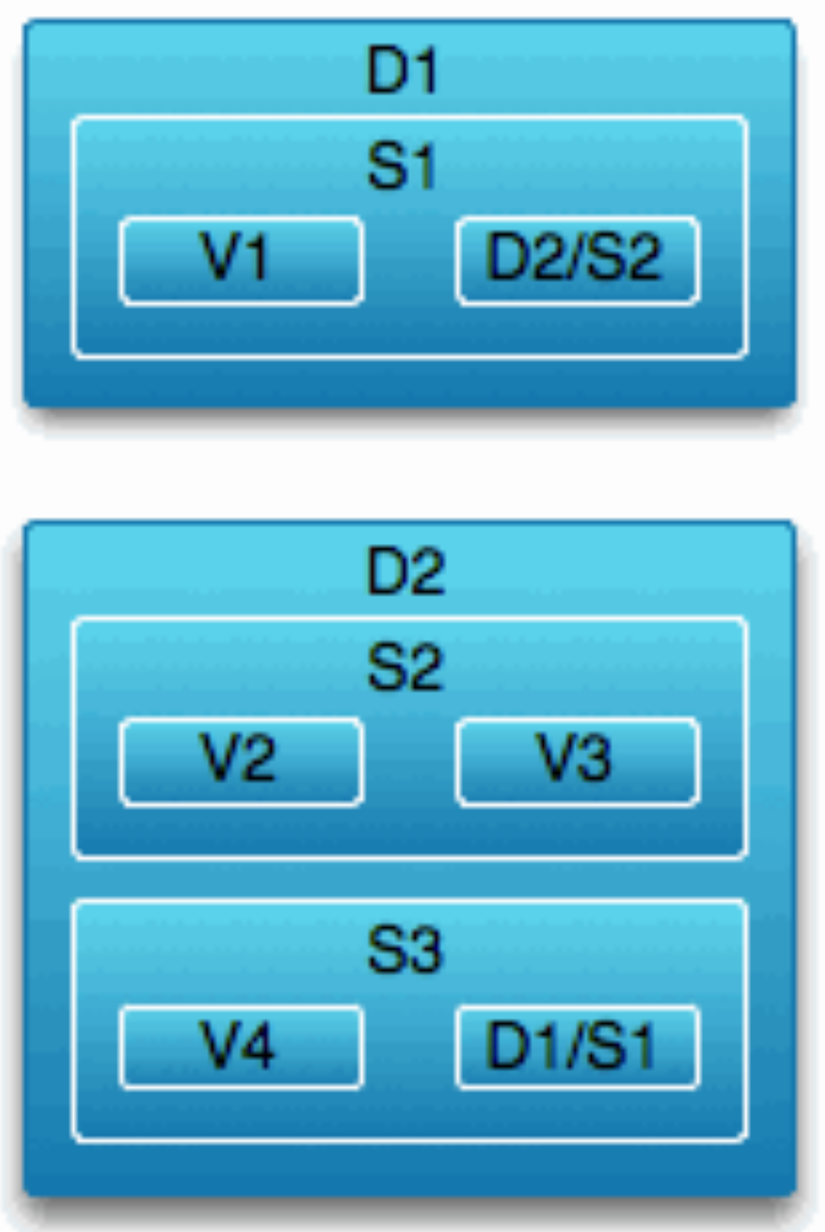
Optimized for traversing connected data



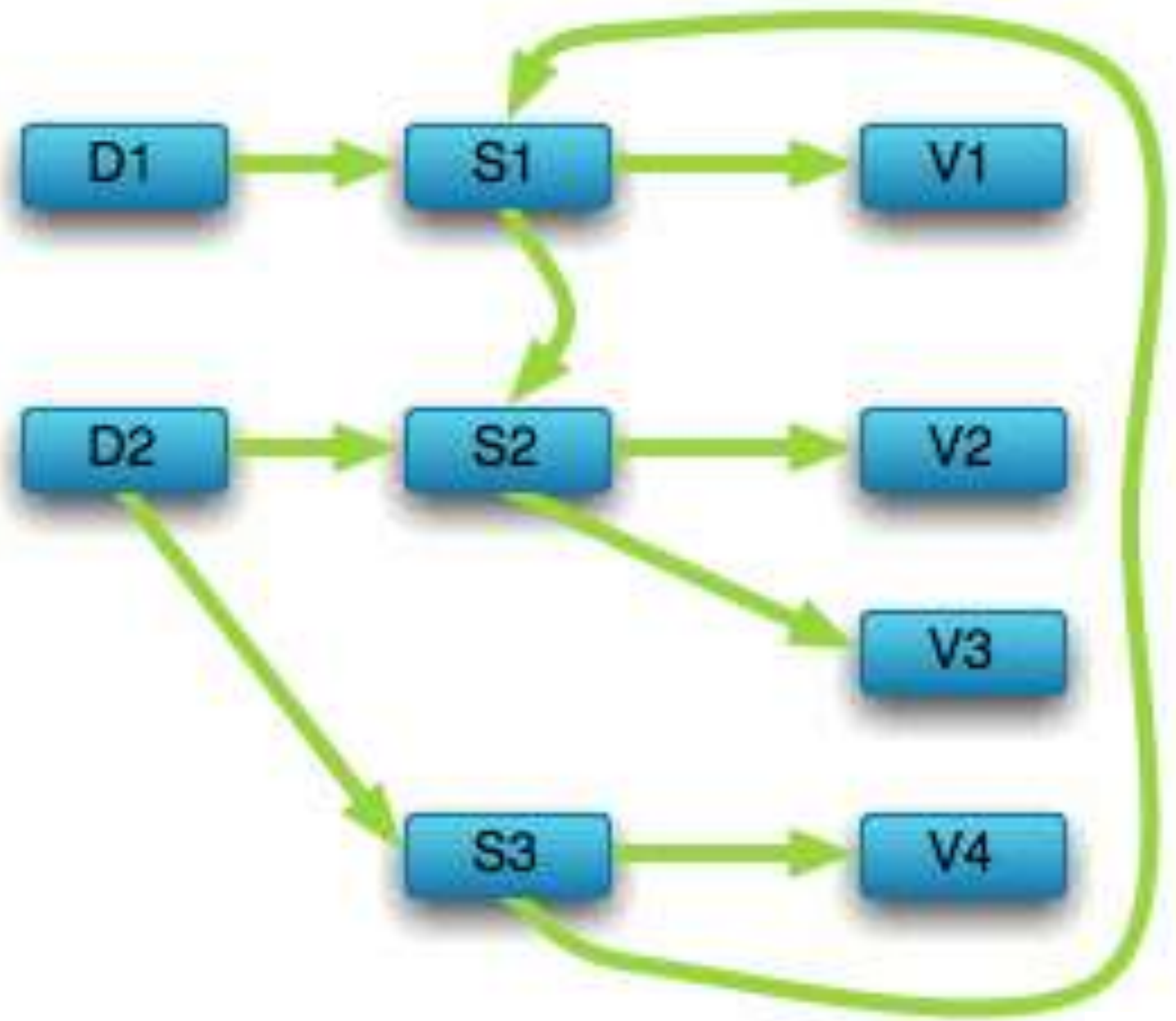
[M. De Marzi, 2012]

Graph Databases Compared to Document Stores

Optimized for “trees” of data



Optimized for seeing the forest and the trees, and the branches, and the trunks



[M. De Marzi, 2012]

The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing

S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu

The Future is Big Graphs

S. Sakr et al

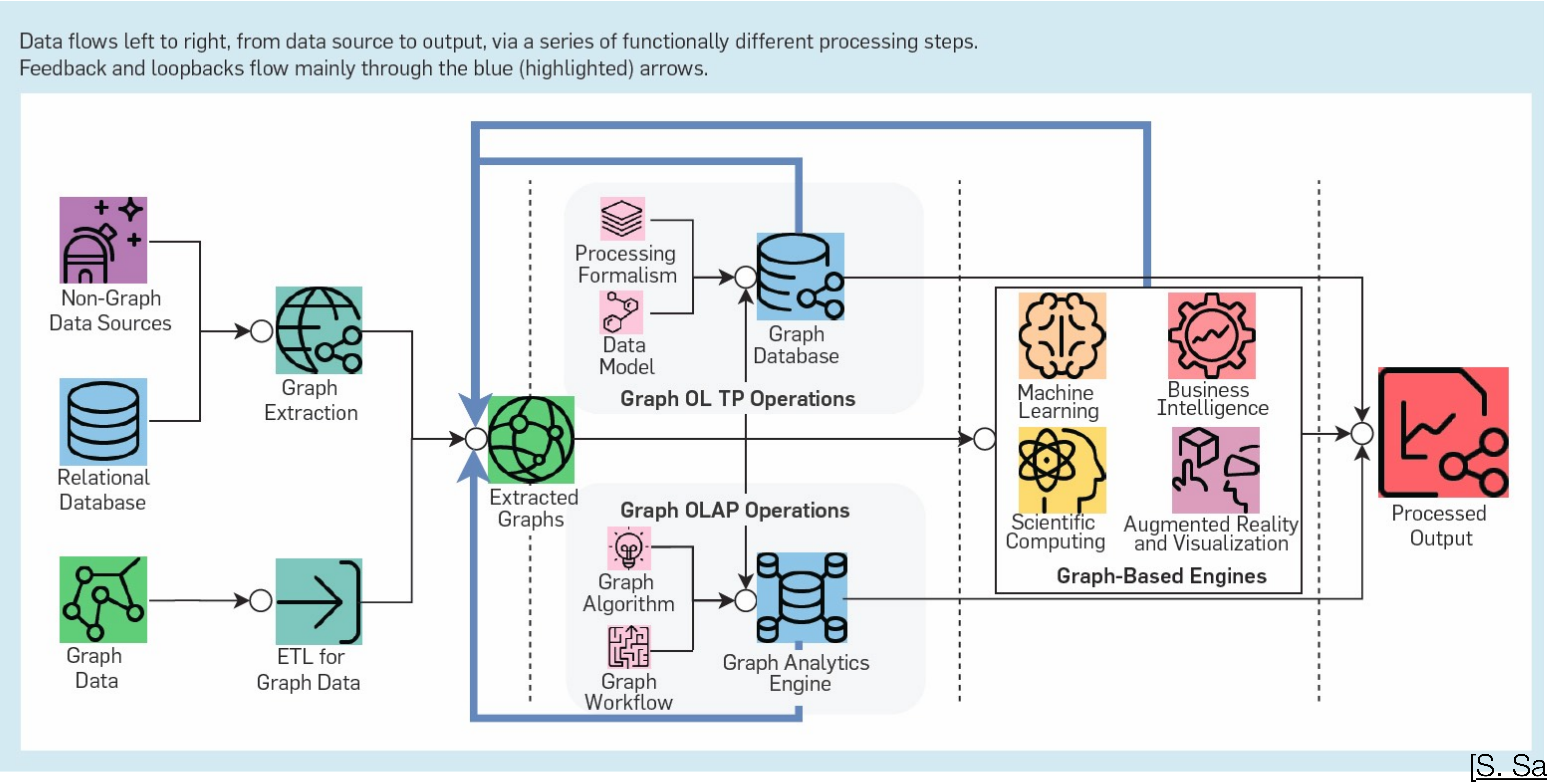
CACM

Insights for the Future of Graph Processing

- Graphs are ubiquitous abstractions enabling reusable computing tools for graph processing with applications in every domain.
- Diverse workloads, standard models and languages, algebraic frameworks, and suitable and reproducible performance metrics will be at the core of graph processing ecosystems in the next decade.

[S. Sakr et al.]

Pipeline for Graph Processing



[S. Sakr et al.]

Graph Databases

D. Lembo and R. Rosati

Why Graph Database Models?

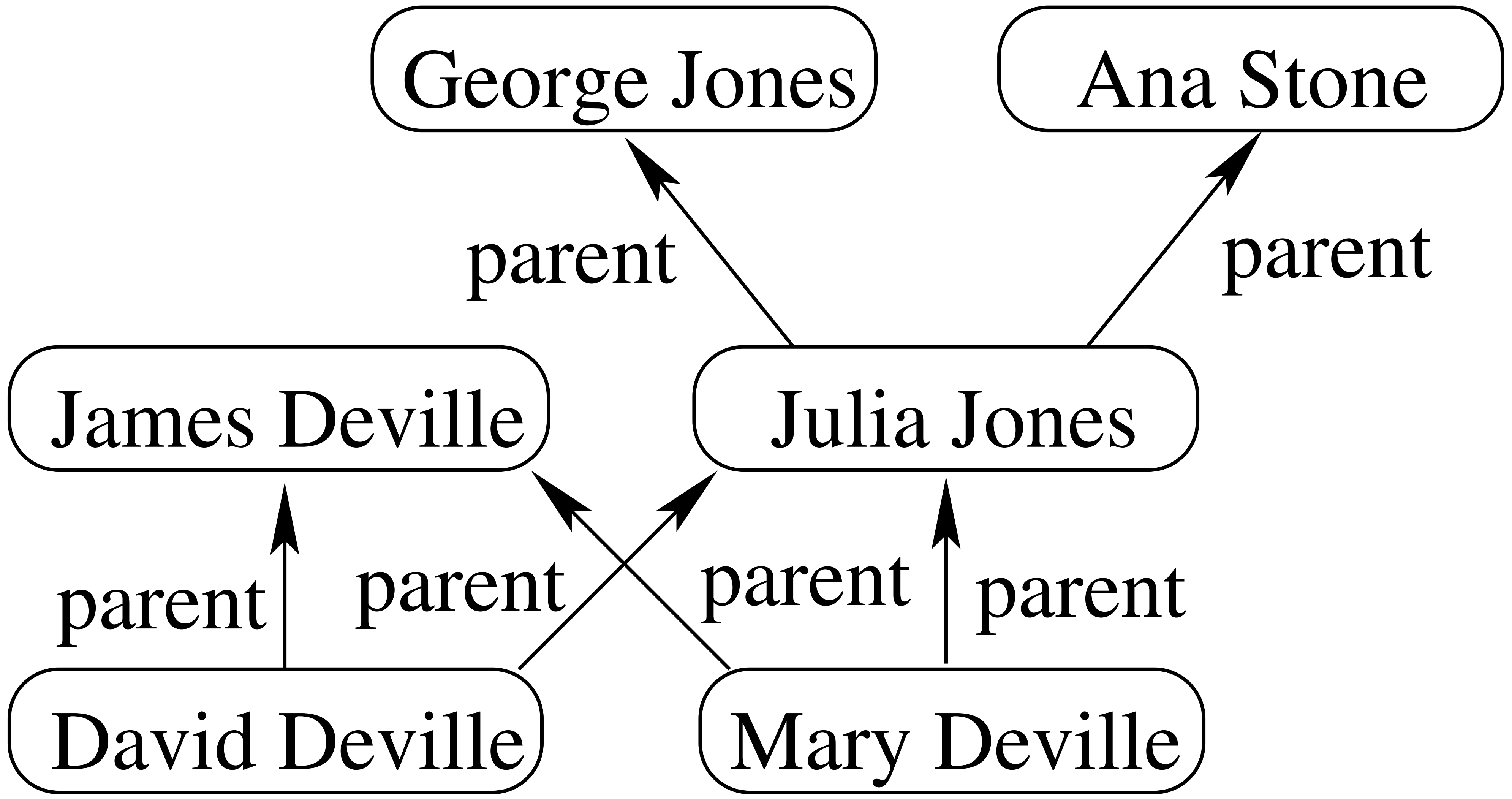
- Graphs has been long ago recognized as one of the most simple, natural and intuitive knowledge representation systems
- Graph data structures allow for a natural modeling when data has graph structure
- Queries can address direct and explicitly this graph structure
- Implementation-wise, graph databases may provide special graph storage structures, and take advantage of efficient graph algorithms available for implementing specific graph operations over the data

[R. Angles and C. Gutierrez, 2017]

Relational Model

NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

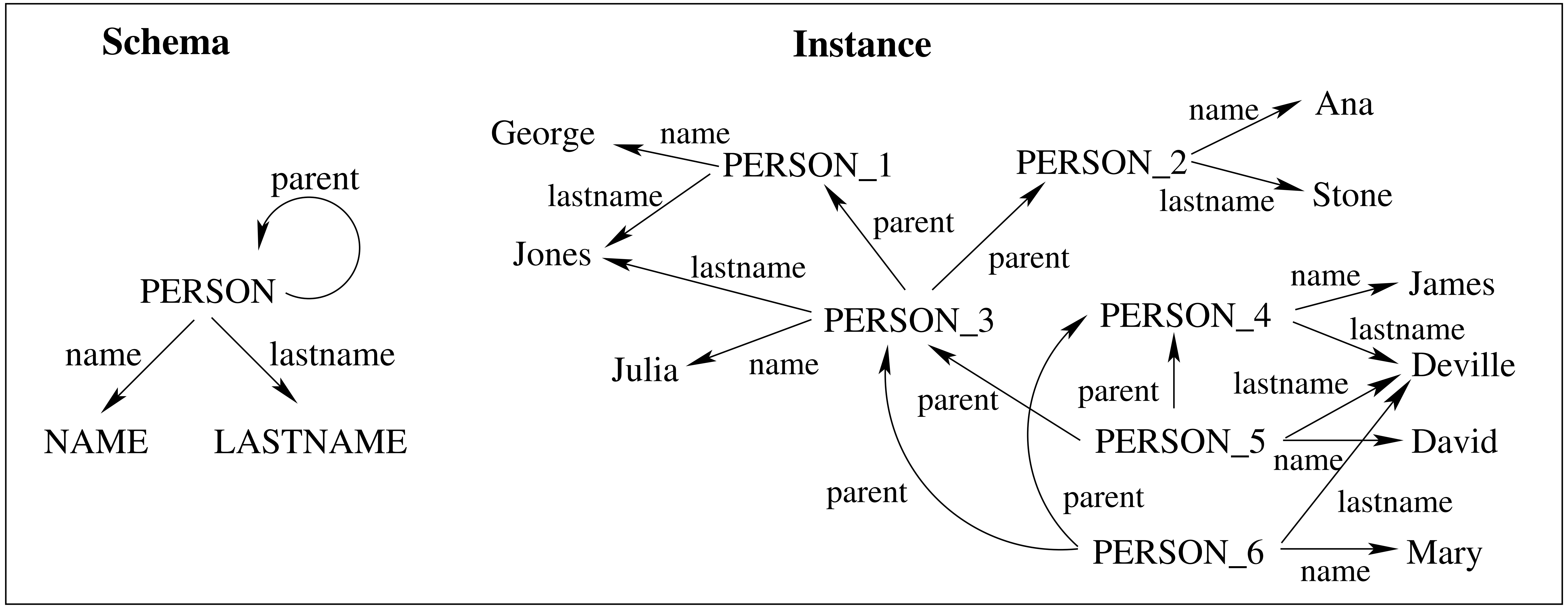
PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



[R. Angles and C. Gutierrez, 2017]

Basic Labeled Model (Gram)

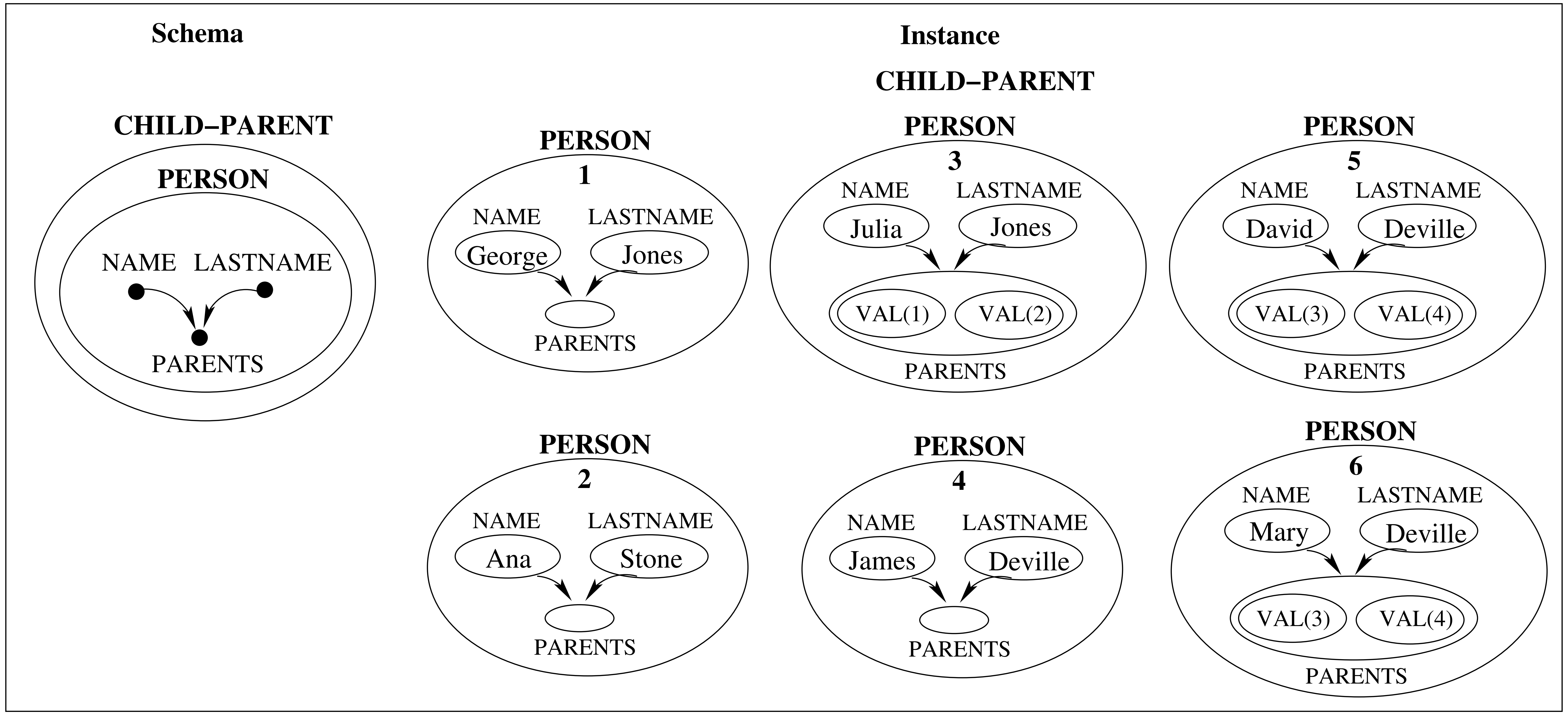
- Directed graph with nodes and edges labeled by some vocabulary
- Gram is a directed labeled multigraph
  - Each node is labeled with a symbol called a **type**
  - Each edge has assigned a label representing a **relation** between types



[R. Angles and C. Gutierrez, 2017]

Hypergraph Model (Groovy)

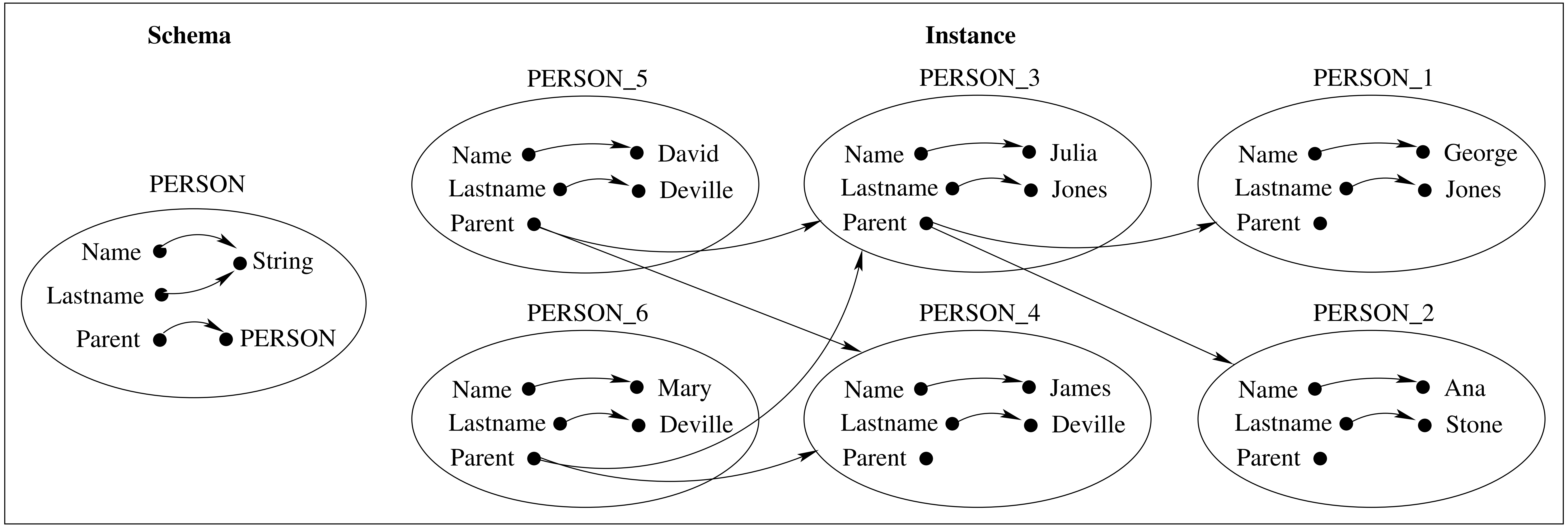
- Notion of edge is extended to **hyperedge**, which relates an arbitrary set of nodes
- Hypergraphs allow the definition of complex objects (undirected), functional dependencies (directed), object-ID and (multiple) structural inheritance



[R. Angles and C. Gutierrez, 2017]

Hypernode Model

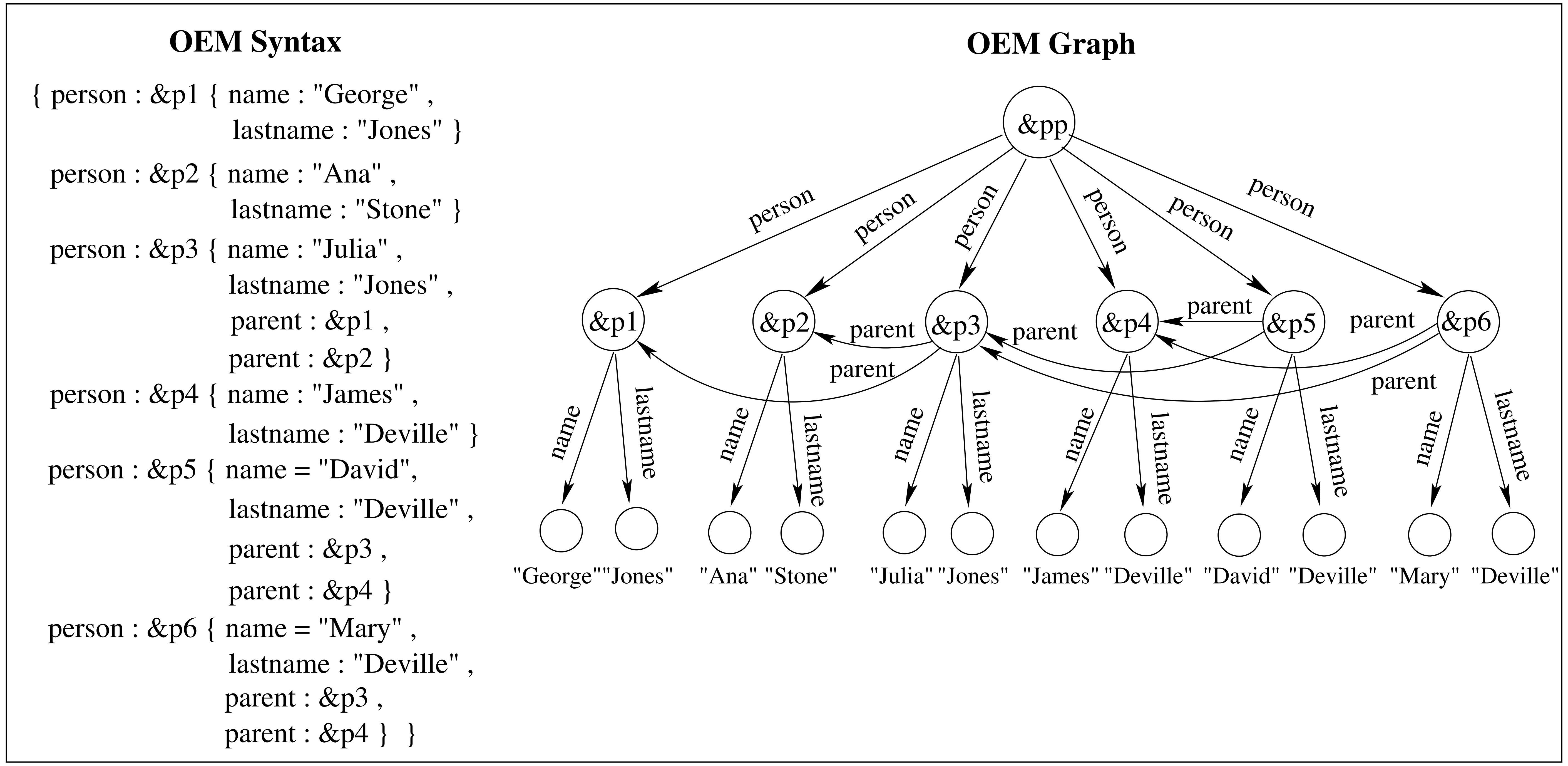
- Hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing **nesting** of graphs
- **Encapsulates** information



[R. Angles and C. Gutierrez, 2017]

Semistructured (Tree) Model: (OEM Graph)

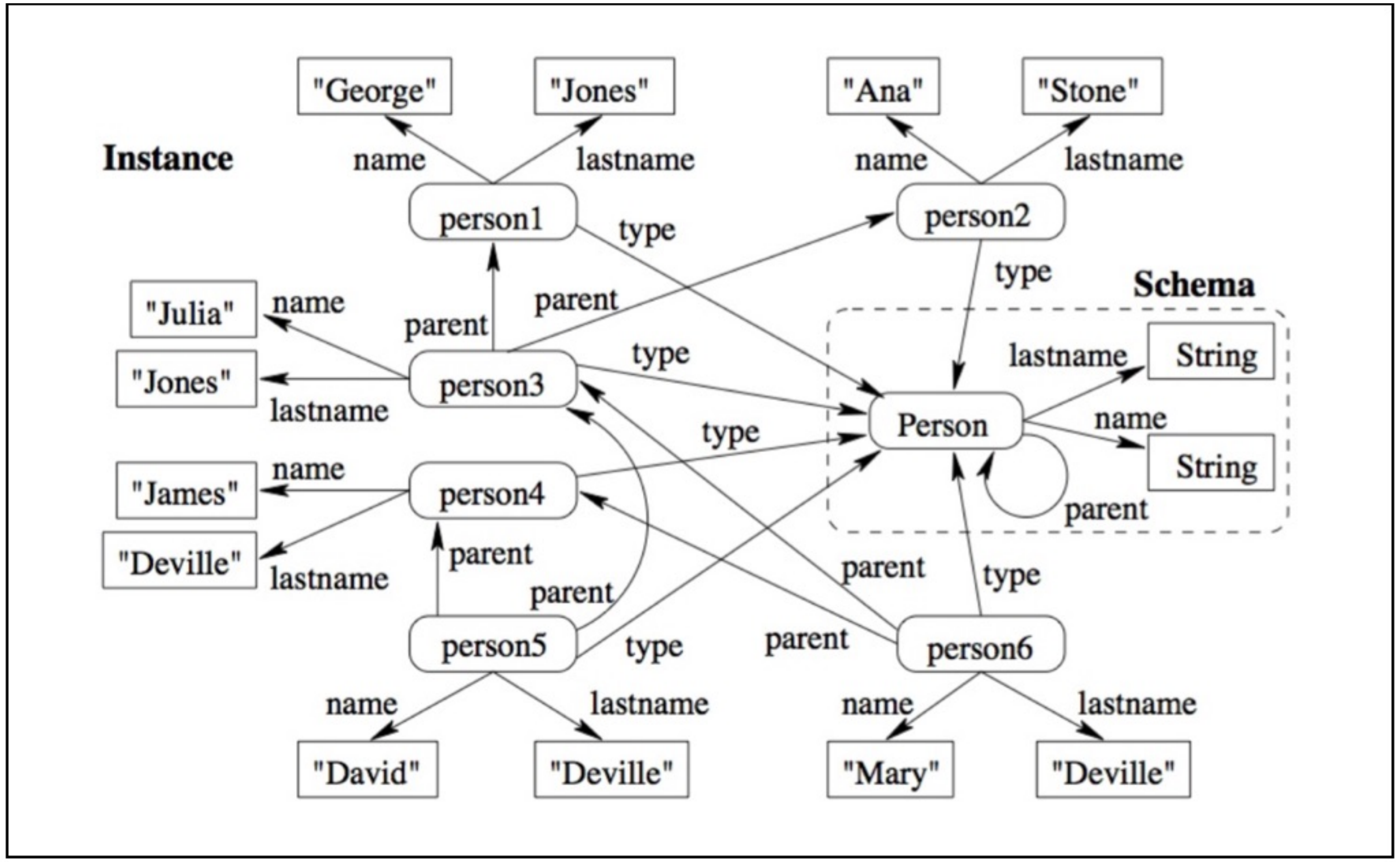
- "Self-describing" data like JSON and XML
- OEM uses pointers to data in the tree



[R. Angles and C. Gutierrez, 2017]

RDF (Triple) Model

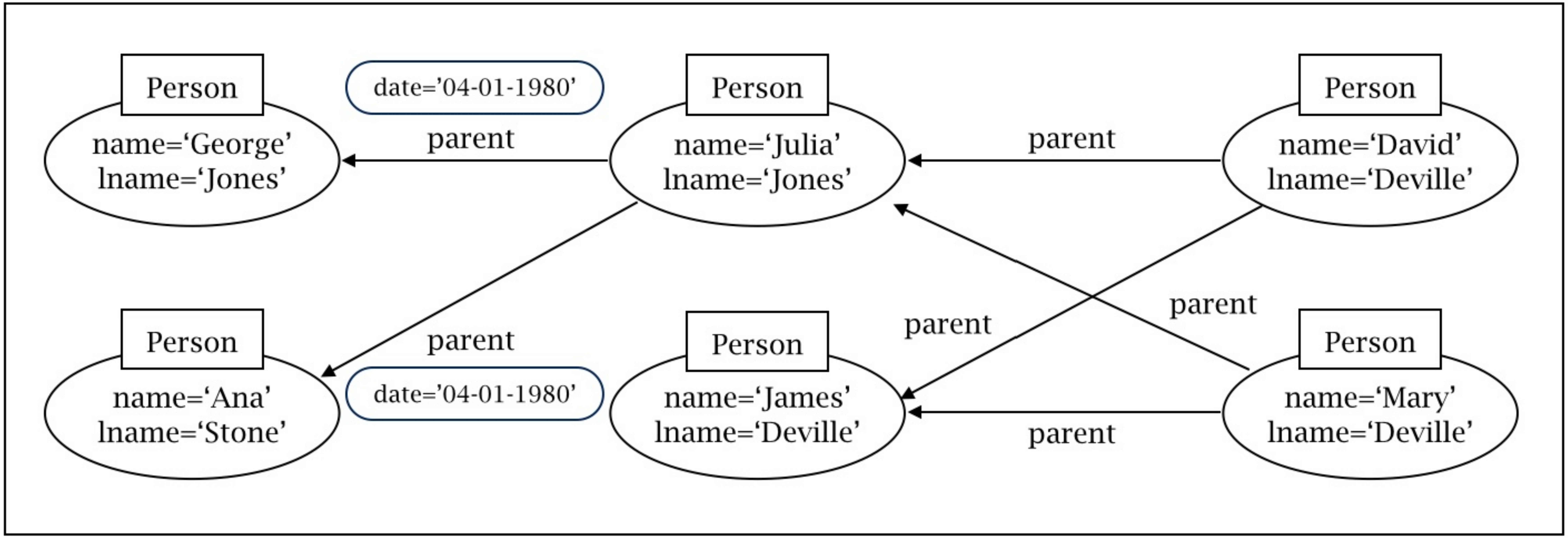
- Interconnect resources in an extensible way using graph-like structure for data
- Schema and instance are **mixed** together
- SPARQL to query
- Semantic web



[R. Angles and C. Gutierrez, 2017]

Property Graph Model (Cypher in neo4j)

- Directed, labelled, attributed multigraph
- Properties are **key/value pairs** that represent metadata for nodes and edges



[R. Angles and C. Gutierrez, 2017]

Types of Graph Queries

- Adjacency queries (neighbors or neighborhoods)
- Pattern matching queries (related to graph mining)
  - Graph patterns with structural extension or restrictions
  - Complex graph patterns
  - Semantic matching
  - Inexact matching
  - Approximate matching
- Reachability queries (connectivity)

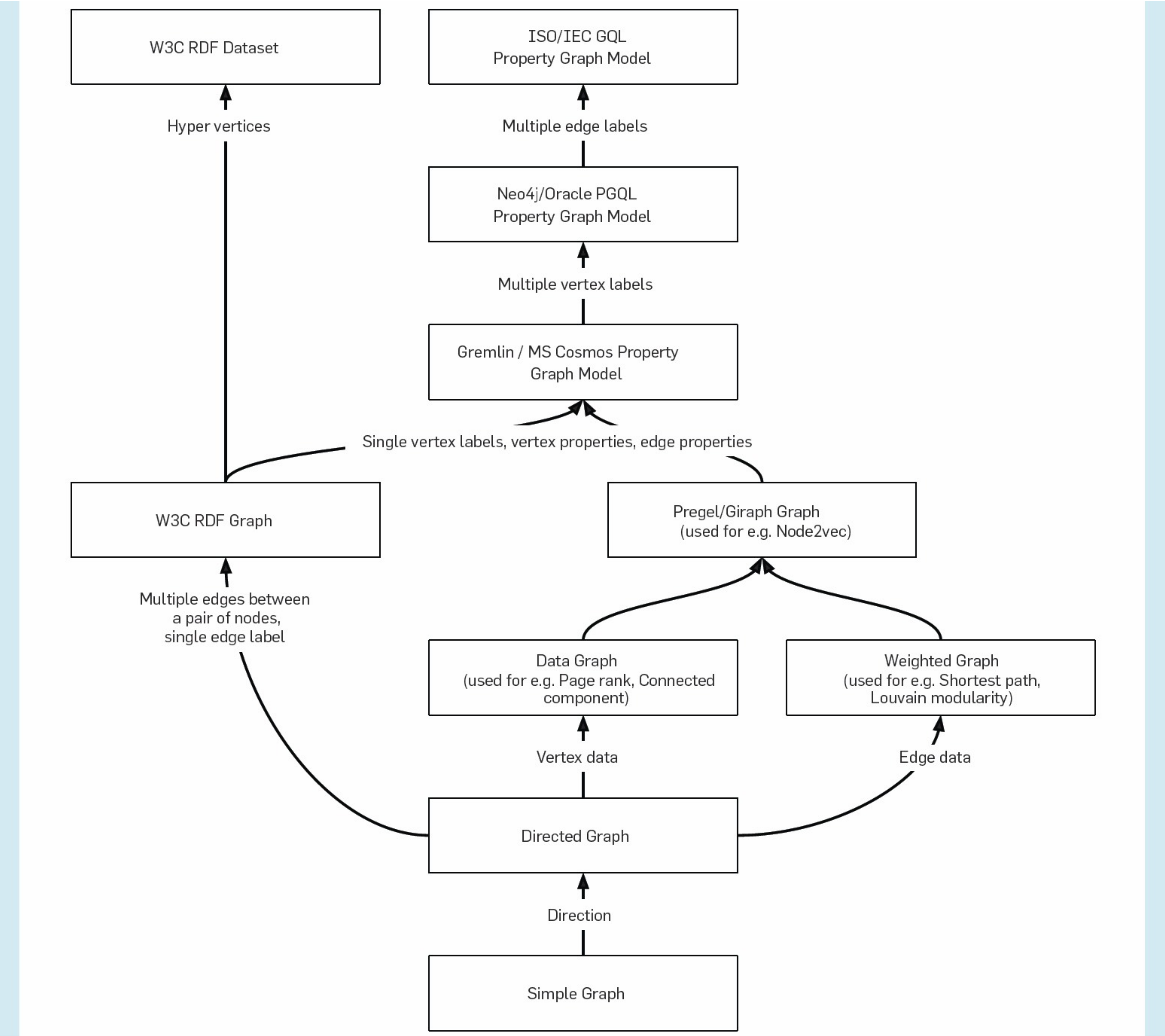
[R. Angles and C. Gutierrez, 2017]

Types of Graph Queries (continued)

- Analytical queries
  - Summarization queries
  - Complex analytical queries (PageRank, characteristic path length, connected components, community detection, clustering coefficient)

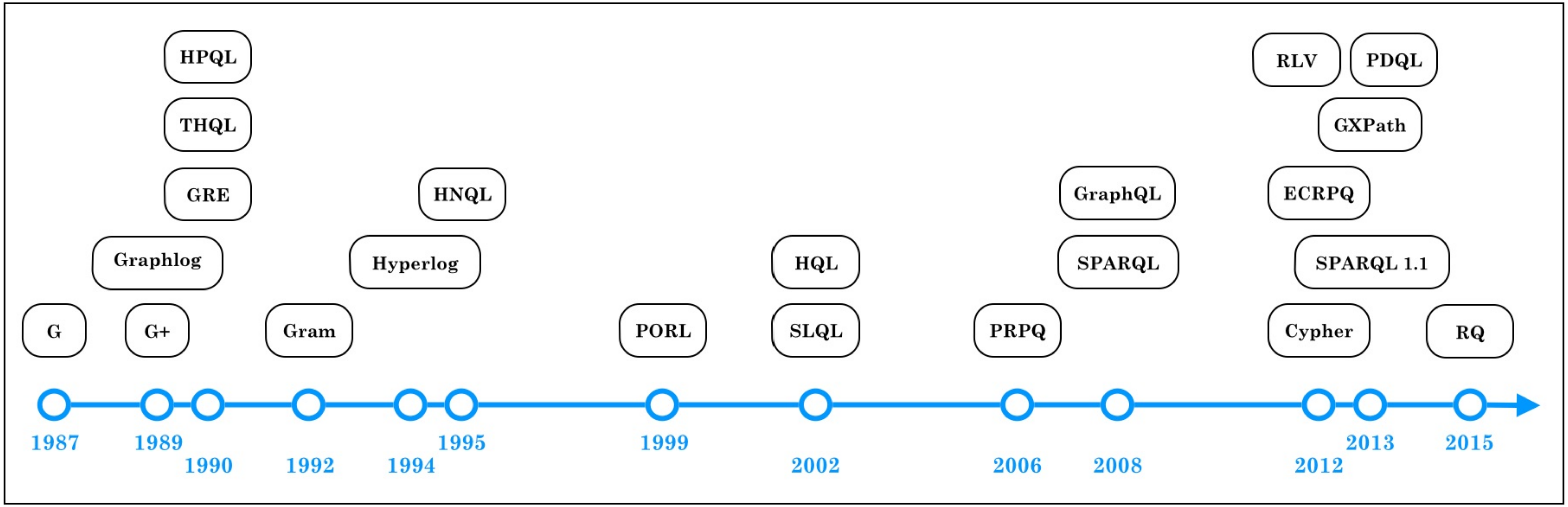
[R. Angles and C. Gutierrez, 2017]

Graph Structures



[S. Sakr et al.]

Graph Query Languages



[R. Angles and C. Gutierrez, 2017]

Cypher

- Implemented by neo4j system
- Expresses reachability queries via path expressions
  - `p = (a)-[:knows*]->(b)`: nodes from `a` to `b` following `knows` edges
- ```
  START x=node:person(name="John")
  MATCH (x)-[:friend]->(y)
  RETURN y.name
  ```

[R. Angles and C. Gutierrez, 2017]
  ```

SPARQL (RDF)

- Uses SELECT-FROM-WHERE pattern like SQL
- ```
  SELECT ?N
  FROM <http://example.org/data.rdf>
  WHERE { ?X rdf:type voc:Person . ?X voc:name ?N }
  ```

[R. Angles and C. Gutierrez, 2017]
  ```

Comparing Graph Database Systems: Features

Data Storage

Graph Database	Main memory	External memory	Backend Storage	Indexes
AllegroGraph	•	•		•
DEX	•	•		•
Filament	•		•	
G-Store		•		
HyperGraphDB	•	•	•	•
InfiniteGraph		•		•
Neo4j	•	•		•
Sones	•			•
vertexDB		•	•	

Operations/Manipulation

Graph Database	Data Definition Language	Data Manipulat. Language	Query Language	API	GUI
AllegroGraph	•	•	•	•	•
DEX				•	
Filament				•	
G-Store	•		•	•	
HyperGraphDB				•	
InfiniteGraph				•	
Neo4j				•	
Sones	•	•	•	•	•
vertexDB				•	

[R. Angles, 2012]

Comparing Graph Database Systems: Representation

Graph Data Structures

	Graphs				Nodes		Edges		
Graph Database	Simple graphs	Hypergraphs	Nested graphs	Attributed graphs	Node labeled	Node attribution	Directed	Edge labeled	Edge attribution
AllegroGraph	•				•		•	•	
DEX				•	•	•	•	•	•
Filament	•				•		•	•	
G-Store	•				•		•	•	
HyperGraphDB		•			•		•	•	
InfiniteGraph				•	•	•	•	•	•
Neo4j				•	•	•	•	•	•
Sones		•		•	•	•	•	•	•
vertexDB	•				•		•	•	

Entites & Relations

	Schema			Instance					
Graph Database	Node types	Property types	Relation types	Object nodes	Value nodes	Complex nodes	Object relations	Simple relations	Complex relations
AllegroGraph					•			•	
DEX	•		•	•	•		•	•	
Filament					•			•	
G-Store					•			•	
HyperGraphDB	•		•		•			•	•
InfiniteGraph	•		•	•	•		•	•	
Neo4j				•	•		•	•	
Sones					•			•	•
vertexDB					•			•	

[R. Angles, 2012]

Comparing Graph Database Systems: Queries

Query Support

	Type			Use		
Graph Database	Query Lang.	API	Graphical Q. L.	Retrieval	Reasoning	Analysis
AllegroGraph	○	●	●	●	●	●
DEX		●		●		●
Filament		●		●		
G-Store	●			●		
HyperGraphDB		●		●		
InfiniteGraph		●		●		
Neo4j	○	●		●		
Sones	●		●	●		●
vertexDB		●		●		

Types of Queries

	Adjacency		Reachability				
Graph Database	Node/edge adjacency	k-neighborhood	Fixed-length paths	Regular simple paths	Shortest path	Pattern matching	Summarization
Allegro	●		●			●	
DEX	●		●	●	●	●	
Filament	●		●			●	
G-Store	●		●	●	●	●	
HyperGraph	●					●	
Infinite	●		●	●	●	●	
Neo4j	●		●	●	●	●	
Sones	●					●	
vertexDB	●		●	●		●	

[R. Angles, 2012]

The (sorry) State of Graph Database Systems

Peter Boncz

Keynote, EDBT-ICDT 2022