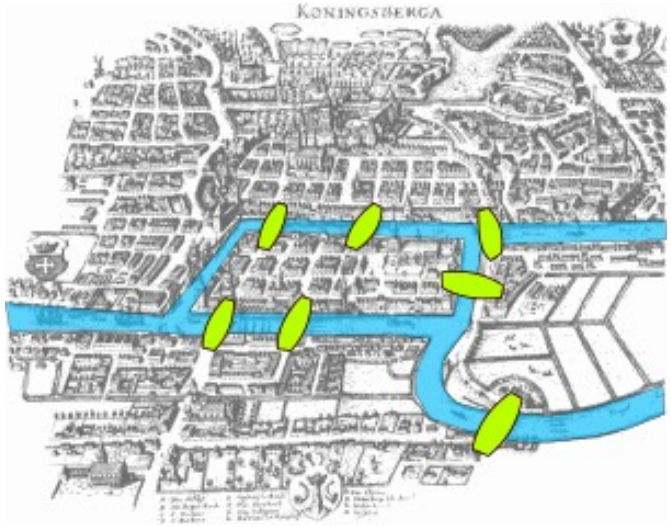
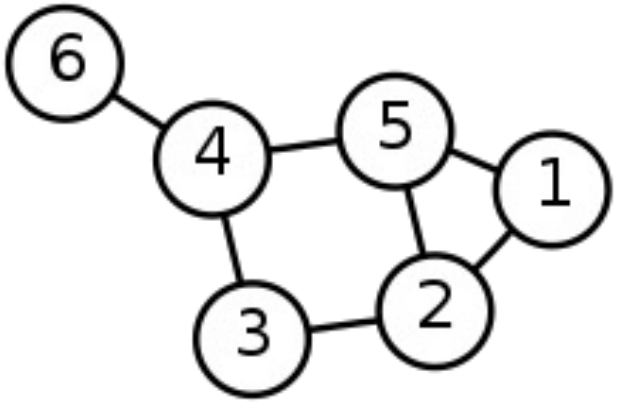
Graph Data Management


KONINGSBERGA


6
4
5
1
3
2

Graphs

Why Graphs?

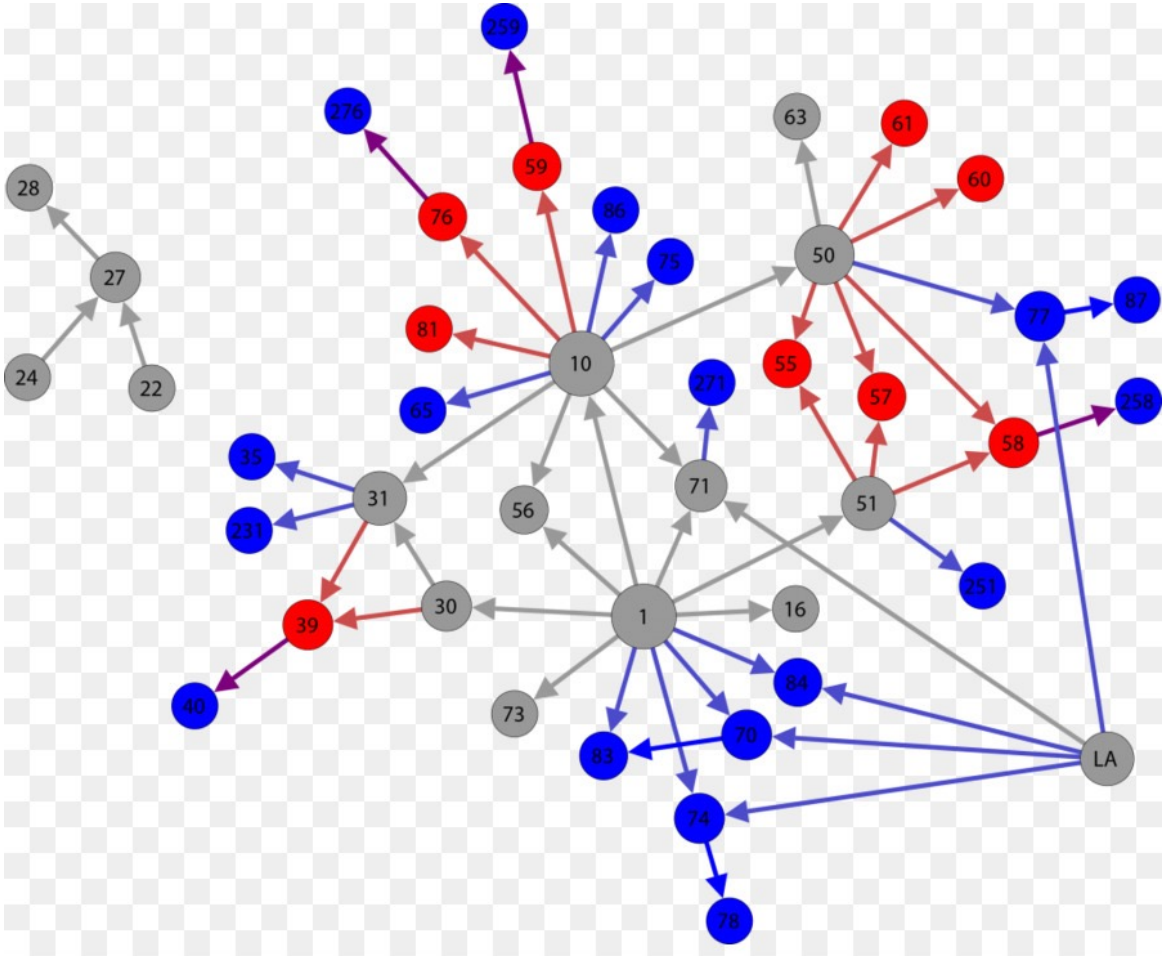
Why Graphs?

- Well-known generic data structure
- One way to address the impedence mismatch
  (objects in your Java don't match the structure of a DB)
- Maths and algorithms are well understood
- «blackboard friendly»

- Many use cases: data naturally modelled as graphs
  - Social networks
  - Web & Semantic Webs
  - Networks (roads, rails, telecommunication, …)
  - Biological networks …

Graphs

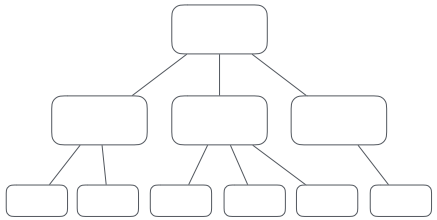
- Capture data consisting of complex relationships
- Focus on data inter-connection and topology



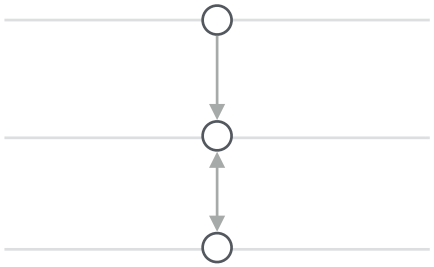
Why Graphs?

Business Processes

Hierarchies

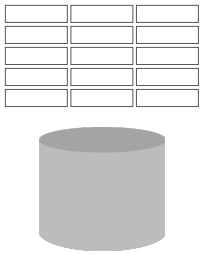


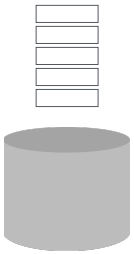
Traditional Supply Chain

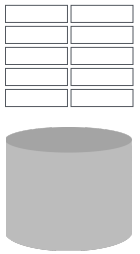


Information

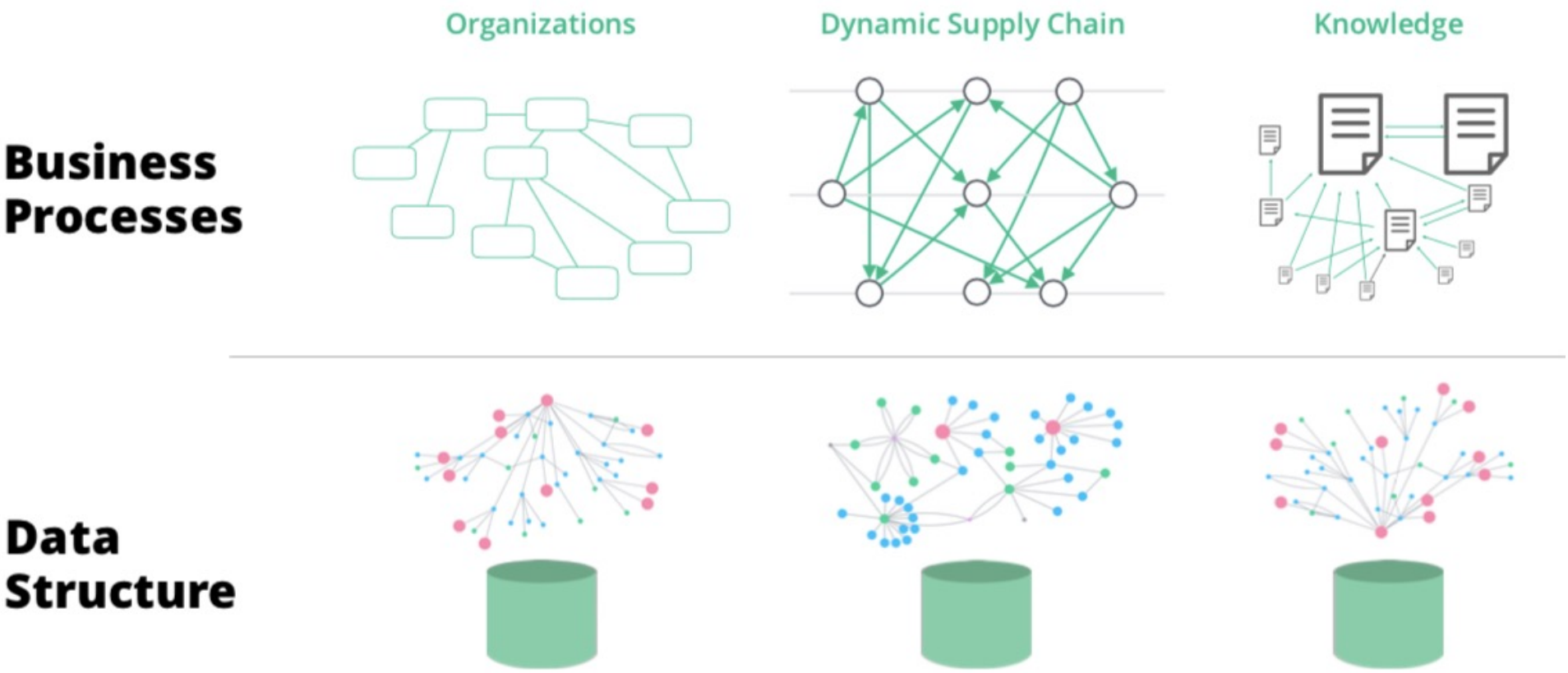
Data Structure







Why Graphs?


Organizations
Dynamic Supply Chain
Knowledge
Business Processes
Data Structure

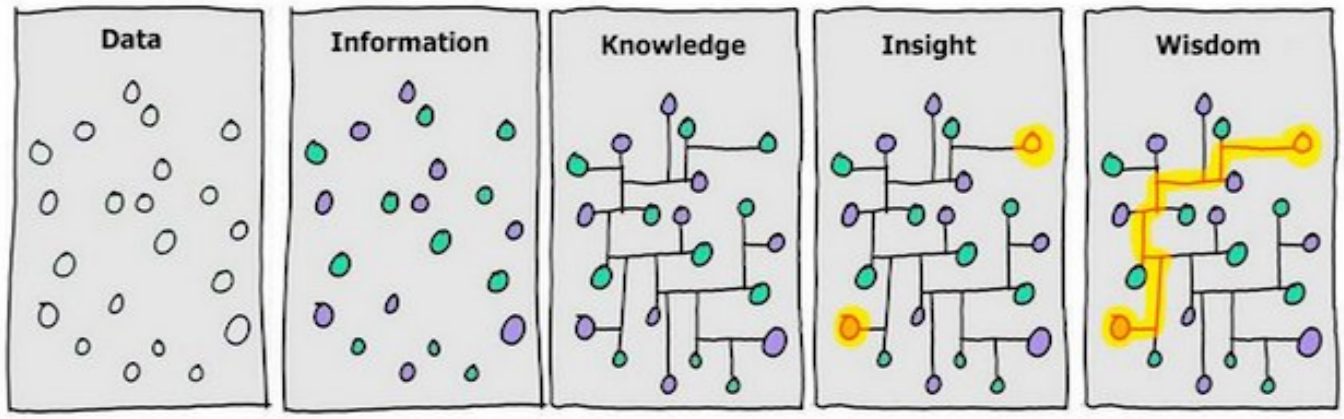
Graphs are everywhere!

Graphs provide a universal and simple blueprint for how to look at the world and make sense of it.

Everyone* uses graphs!

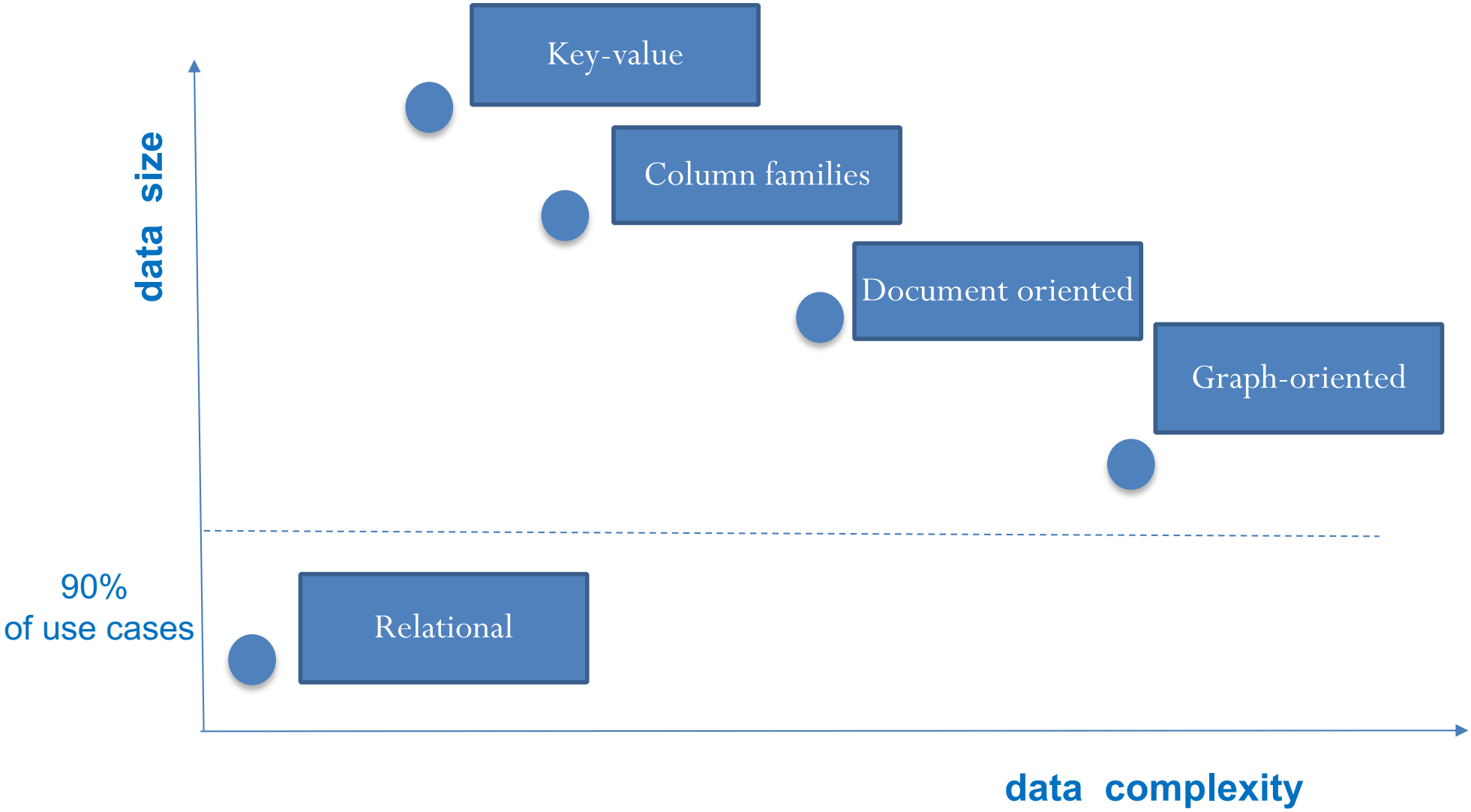
Tech-driving applications
= data science + multi-hop relationships

*not yet :-(



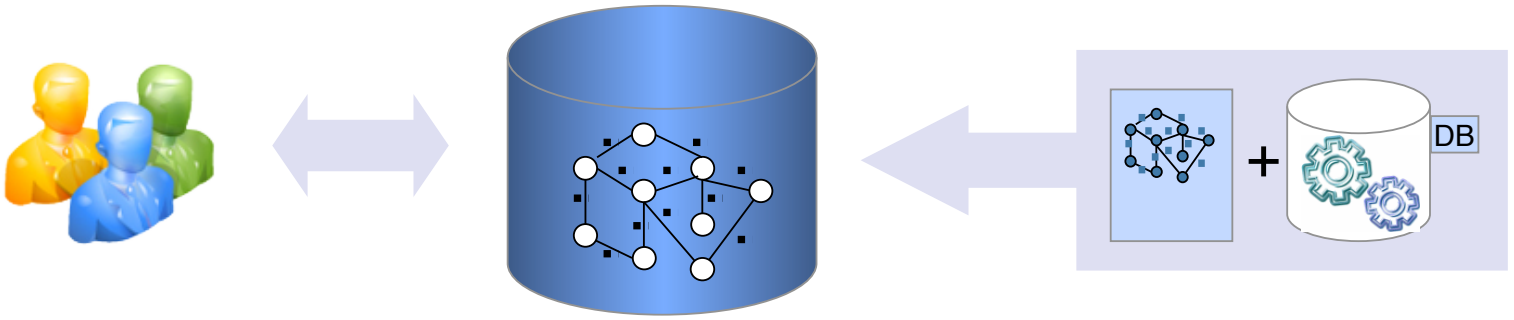
[Cartoon by David Somerville, based on a two pane version by Hugh McLeod.]

NoSQL data models

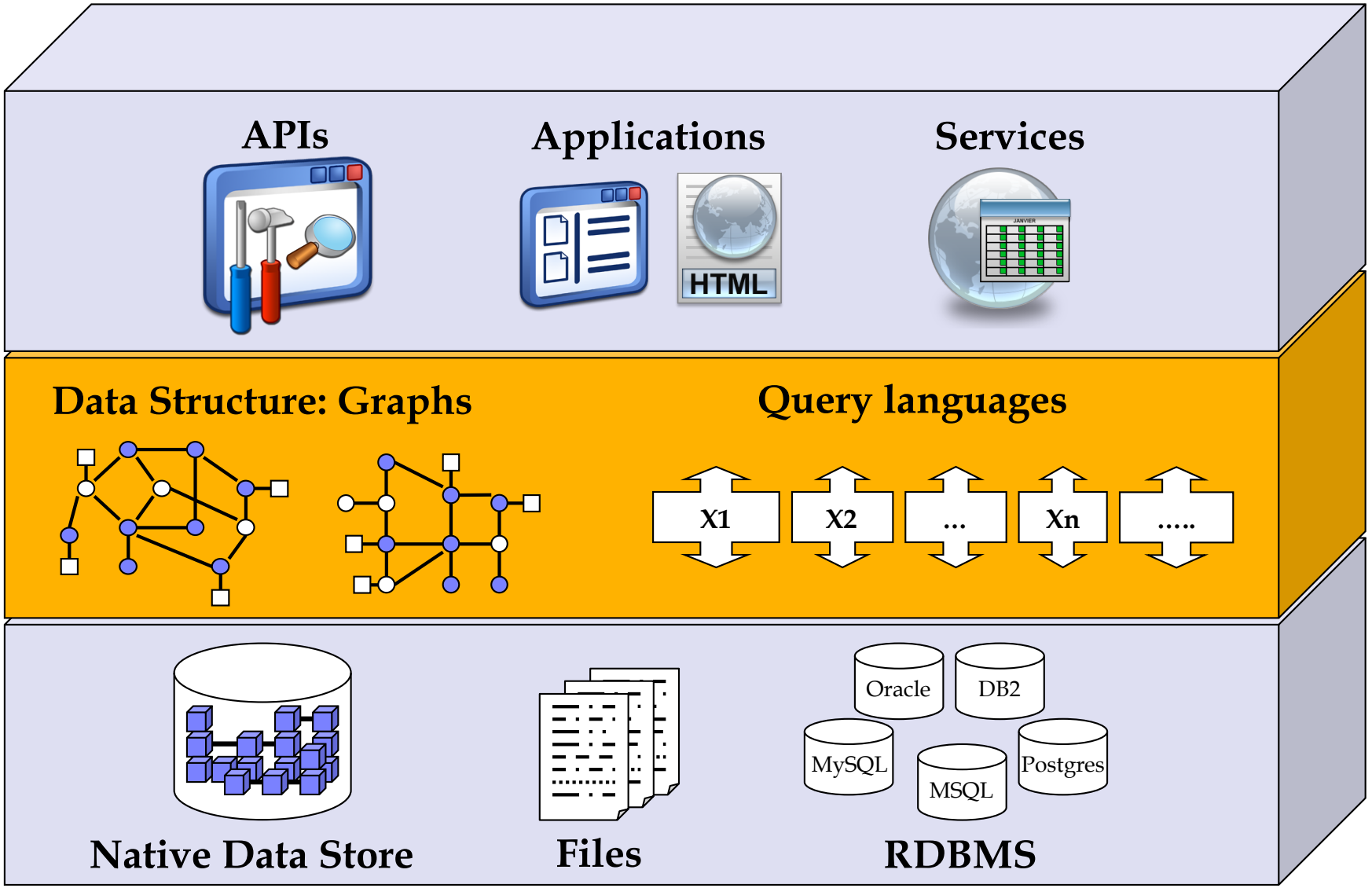


Data Management (database approach)

Manage huge volumes of data with logical precision
Separate modeling from implementation levels



The Database Modelling Level



Graph Data Model

Data structure

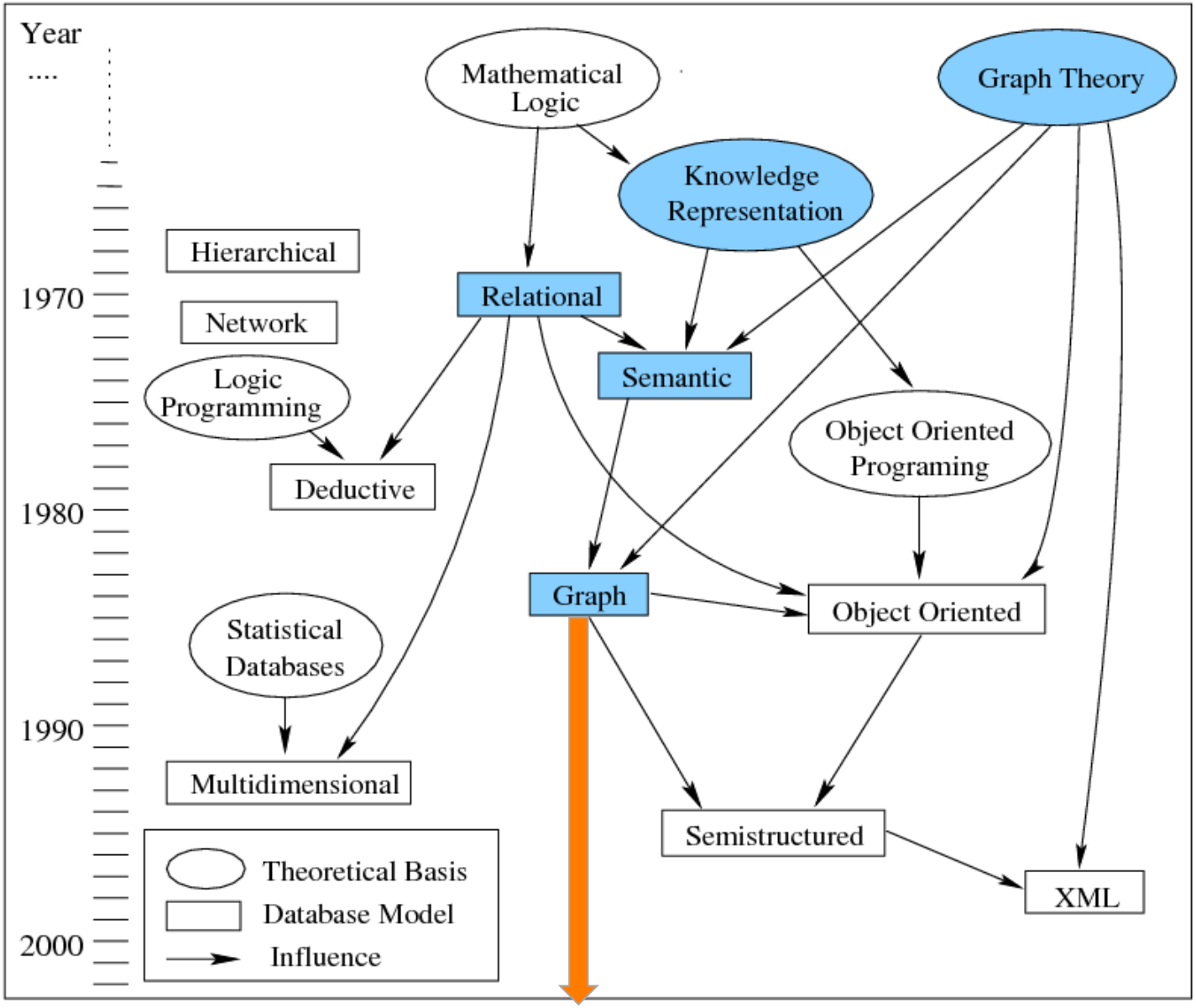
Data and/or the schema are represented by graphs, or by data structures generalizing the notion of graph (hypergraphs or hypernodes).

Graph Query Languages

Query Language

Data manipulation is expressed by graph transformations, or by operations whose main primitives are on graph features like paths, neighborhoods, subgraphs, graph patterns, connectivity, and graph statistics.

Historical Perspective – Data models



Y2k Change of Phase

Hardware able to process big graphs

As always, software behind hardware, and theory/models behind software

Use Cases

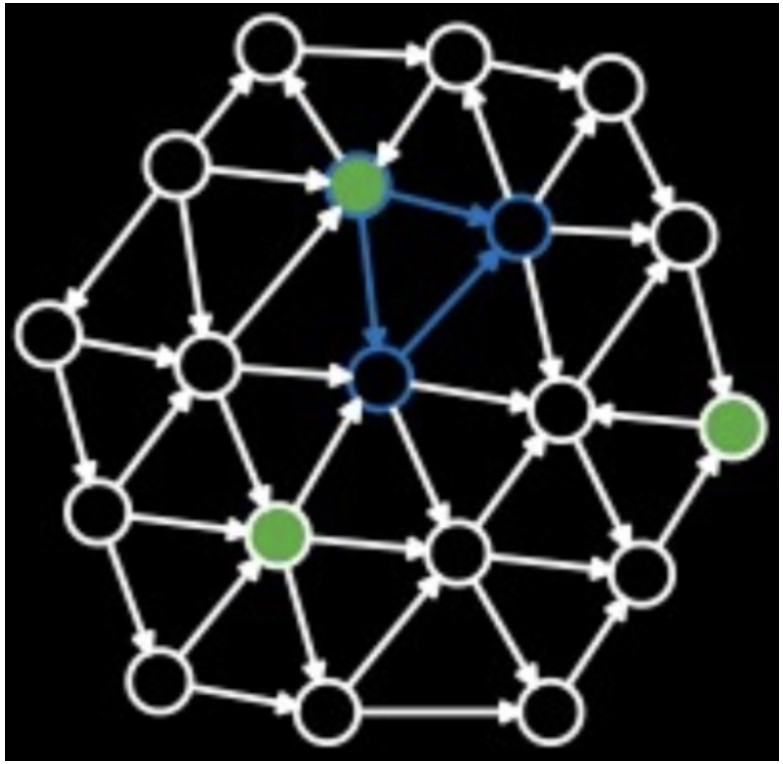
Social Network Domain

Social network domain – starting from Facebook

- Who likes me?
- Who likes something I've posted?
- Who likes something I like?
- Is any of my friend attending an event I am interested in?
- Is any of my old school mates friend any of my university colleagues?
- …

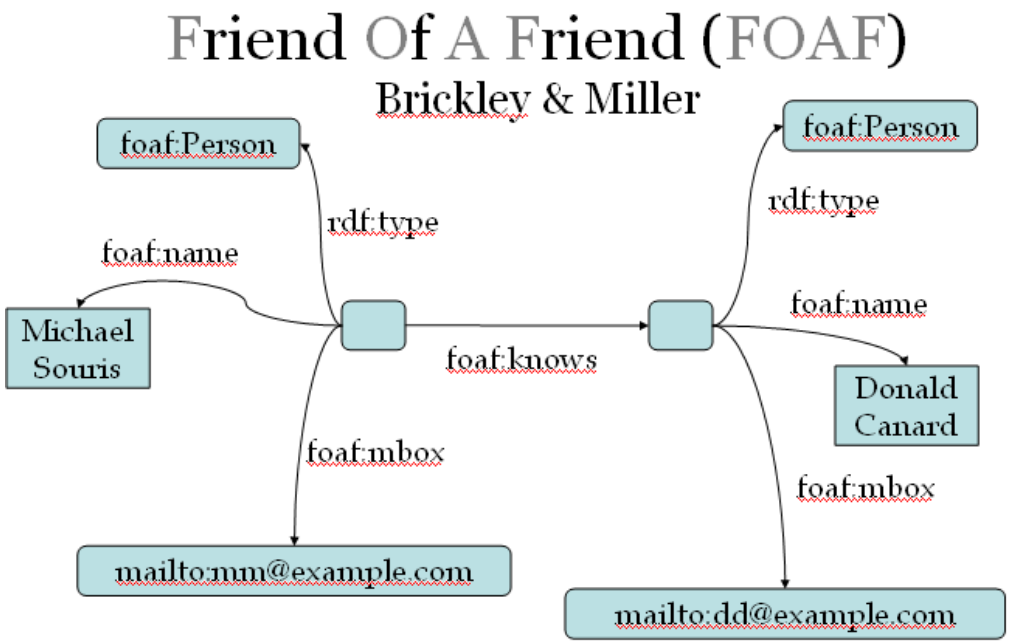
Query = Traversal

- "Who likes me"
  - traversal of depth 1 of the incoming nodes to me with the property "Like"
- "Is any of my old school mates friend of any of my university colleagues?"
  - traversal from me to all my friends, then from friend to friend

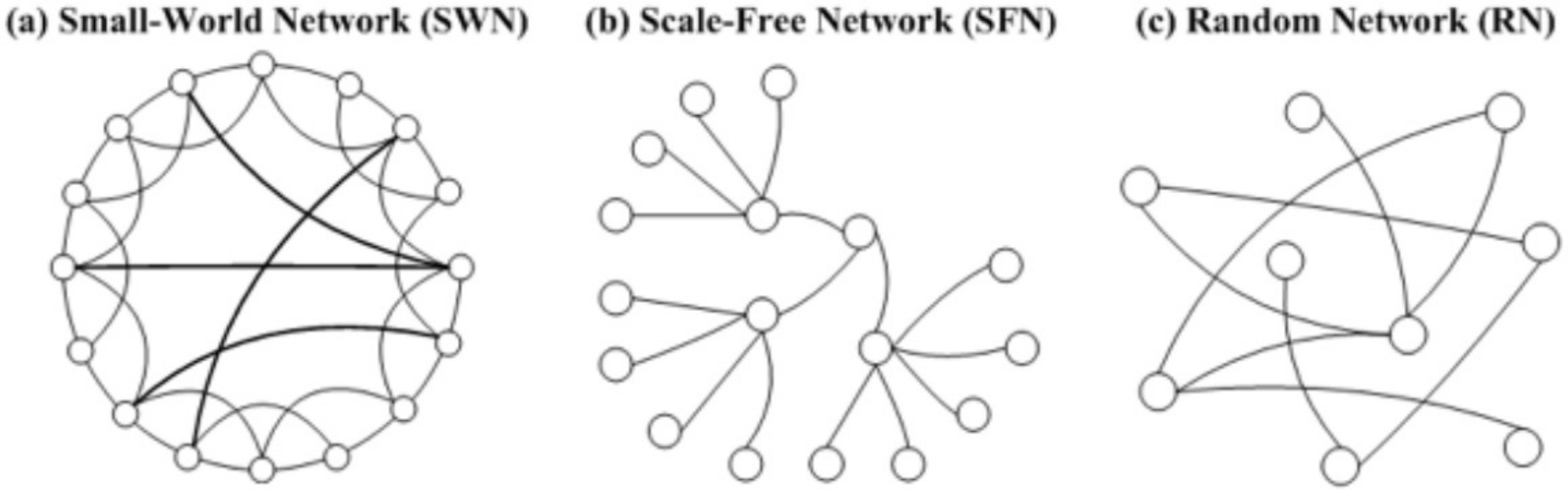


Use Cases - Web Domain

Use Case	Graph Query
What is/are the most cited paper/s?	Degree of a node
What is the influence of article D?	Paths
What is the Erdös distance between authos X and author Y?	Distance
Are suspects A and B related?	Paths
All relatives of degree one of Alice	Adjacency

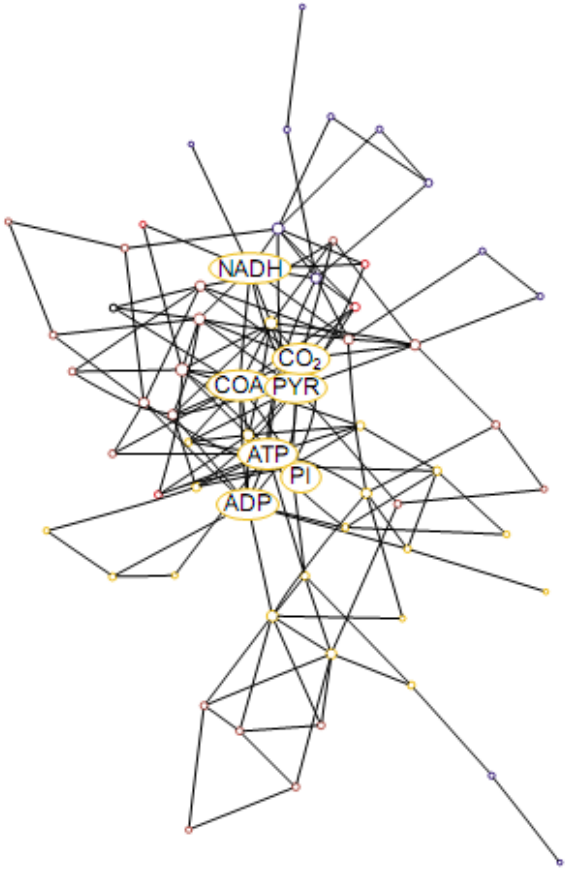


Use Cases – Social Network



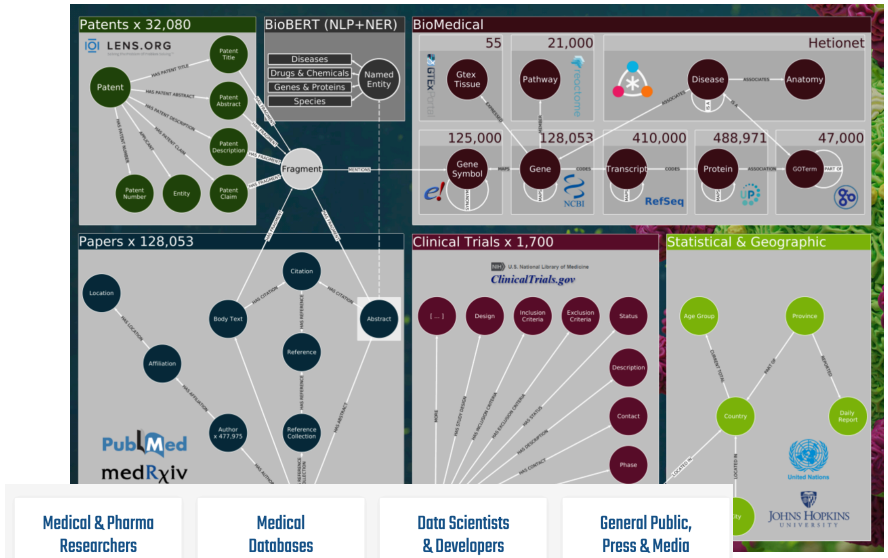
Use Cases - Biology Domain

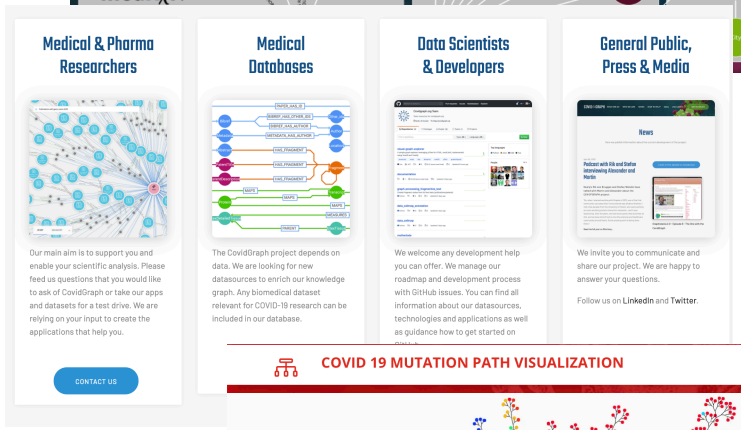
Use Case	Graph Query
Chemical structure associated with a node	Node matching
Find the difference in metabolisms between two microbes	Graph intersection, union, difference
To combine multiple protein interaction graphs	Majority graph query
To construct pathways from individual reactions	Graph composition
To connect pathways, metabolism of co-existing organisms	Graph composition
Identify “important” paths from nutrients to chemical outputs	Shortest path queries
Find all products ultimately derived from a particular reaction	Transitive Closure
Observe multiple products are co-regulated	Least common ancestor
To find biopathways graph motifs	Frequent subgraph recognition
Chemical info retrieval	Subgraph isomorphism
Kinaze enzyme	Subgraph homomorphism
Enzyme taxonomies	Subsumption testing
To find biopathways graph motifs	Frequent subgraph recognition

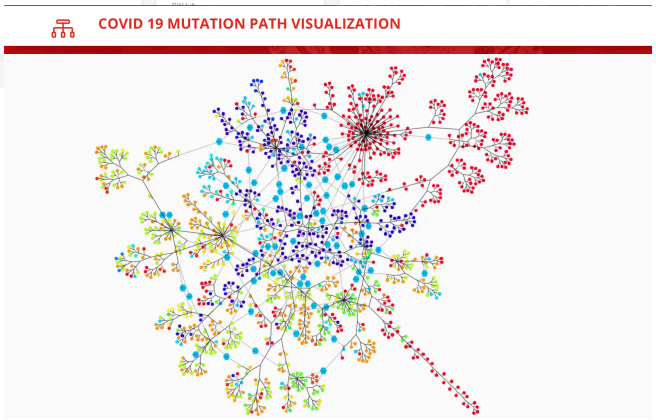


Use Cases – A plethora of domains

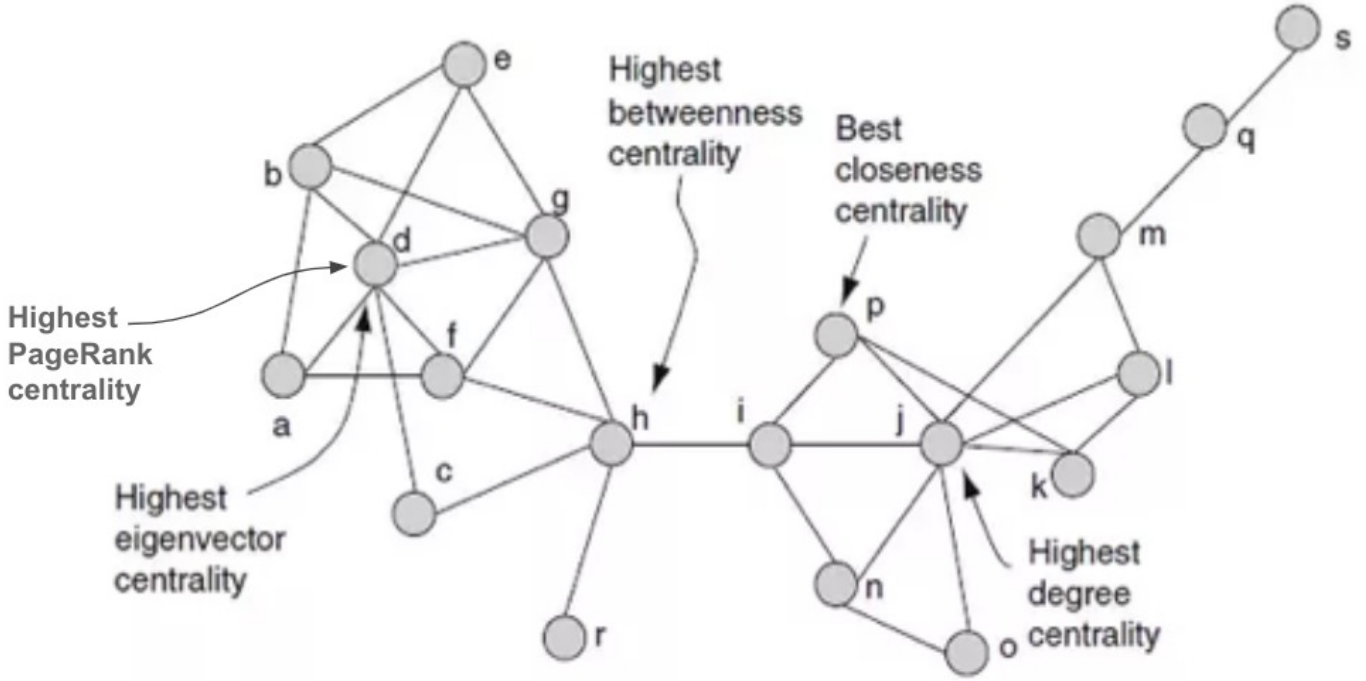
- Among which, the covidgraph.org initiative aiming at building the Covid19 knowledge graph:
  - Collecting patents, publications about the human coronaviruses
  - Biomedical data (genomics and omics)
  - Experimental data about clinical trials
  - Key demographic indicators







Use cases – A plethora of domains Centrality Measures



Different measures can be useful in different scenarios such web-ranking (page-rank), critical points detection (betweenness), transportation hubs (closeness)

Different types of graph data

- large set of small graphs
  - e.g., chemical compounds, biological pathways, …
  - searching for graphs that match the query
- few numbers of very large graphs
  - or one huge (not connected) graph
  - e.g., Web graph, social networks, …

Graph Data Models

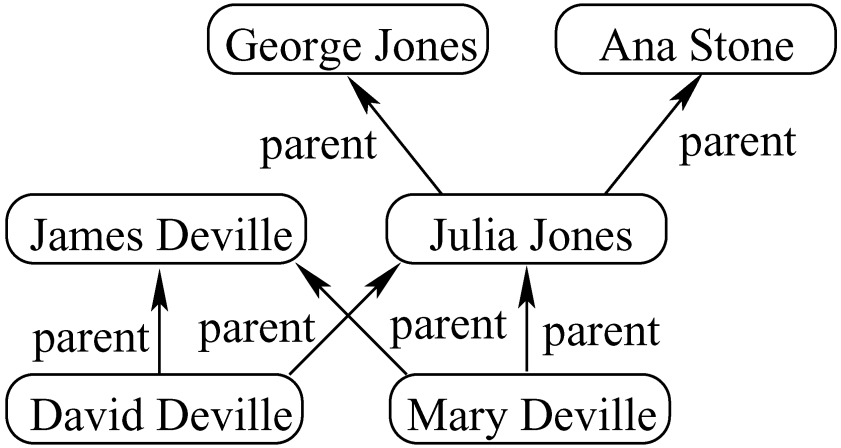
Graph data models

- How do humans conceptualize graphs?
- Interoperability issues (due to multiple heterogeneous data sources) are to be taken into account
- Balancing understandability and expressive power

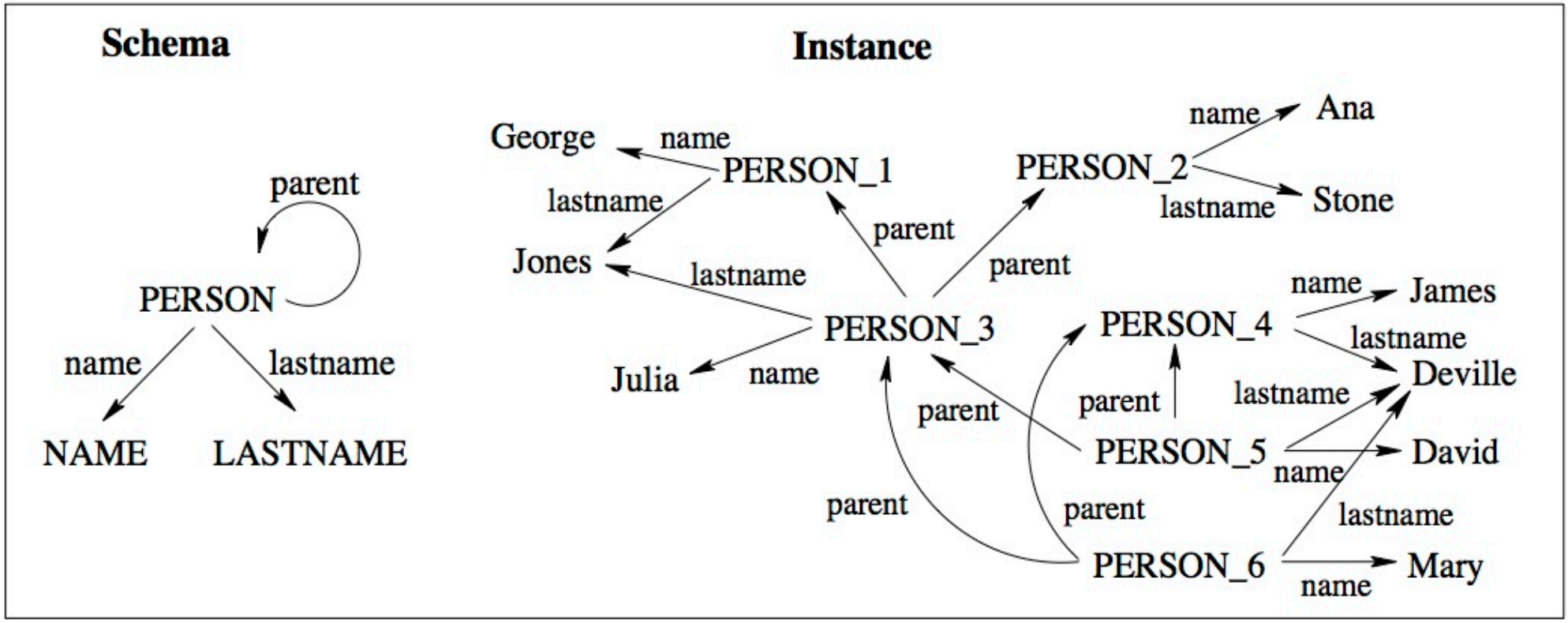
Example: Genealogy Data and Diagram

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

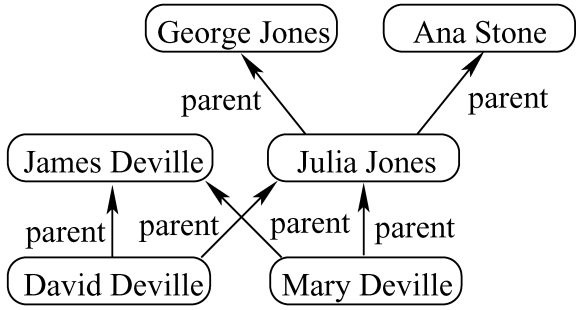


Simple Graph



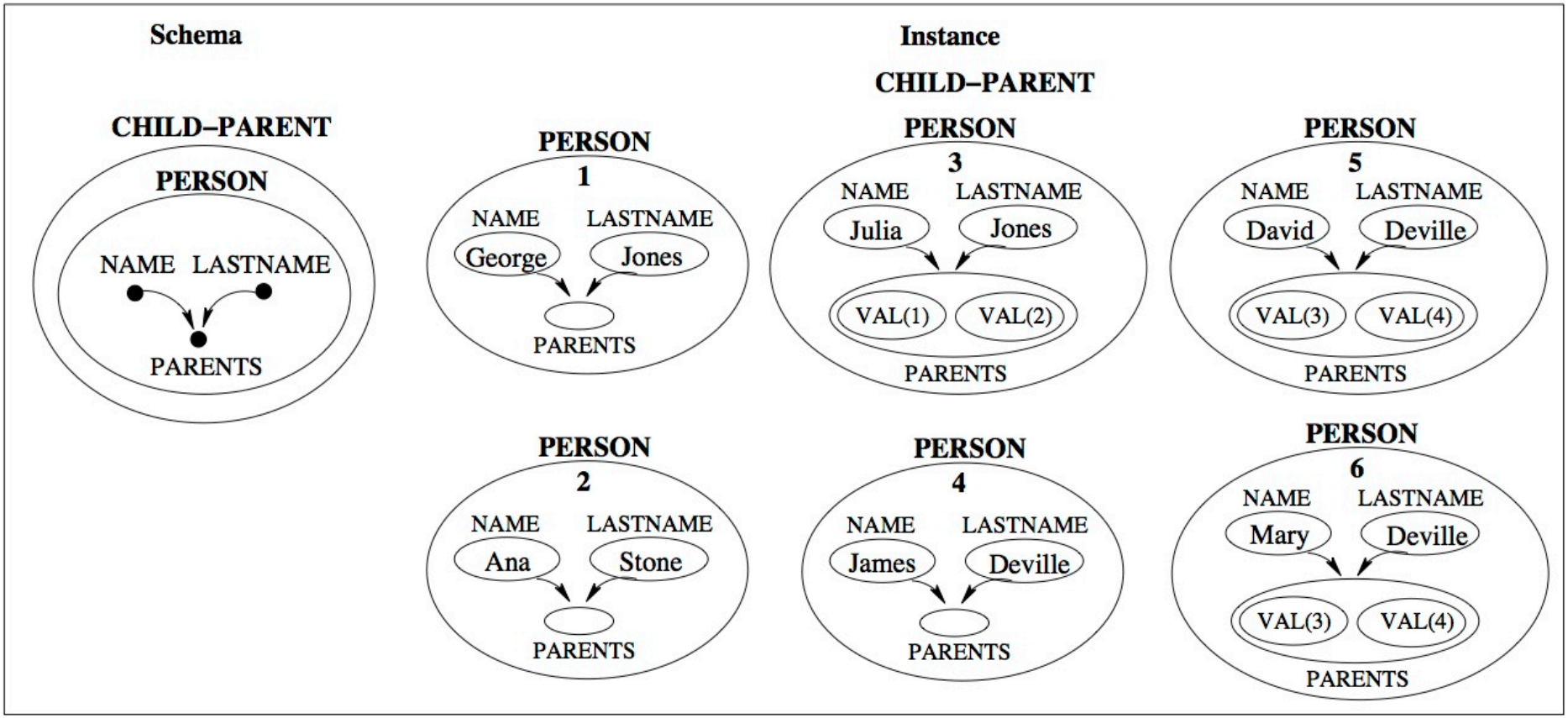
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

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



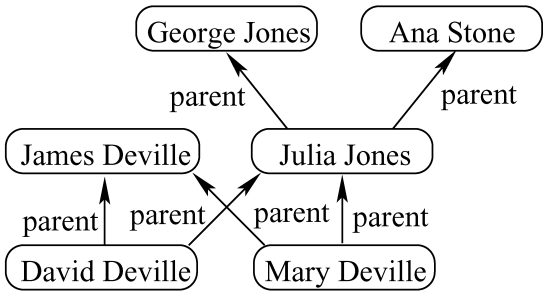
Hypergraph

A hyperedge relates an arbitrary set of nodes



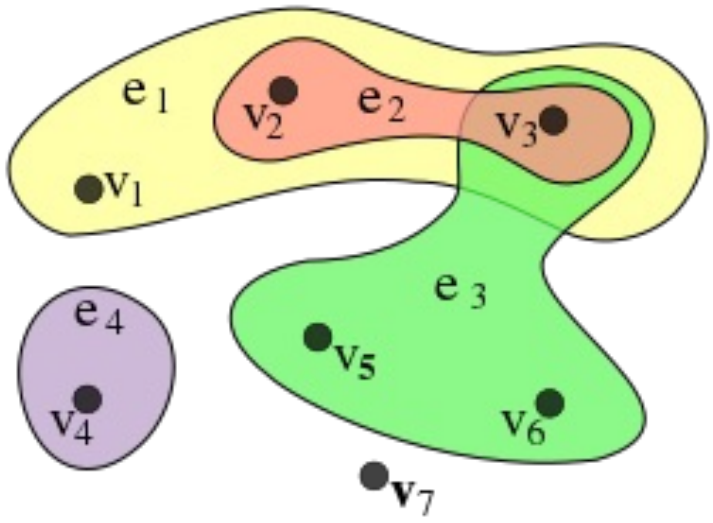
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

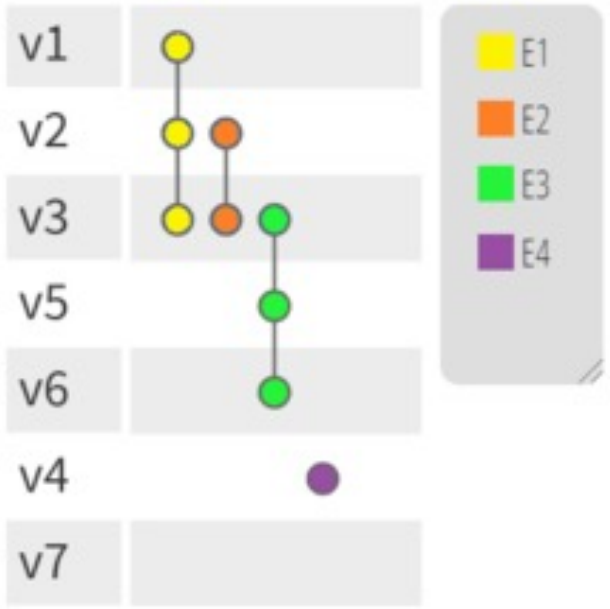
PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



Hypergraph

A hyperedge relates an arbitrary set of nodes

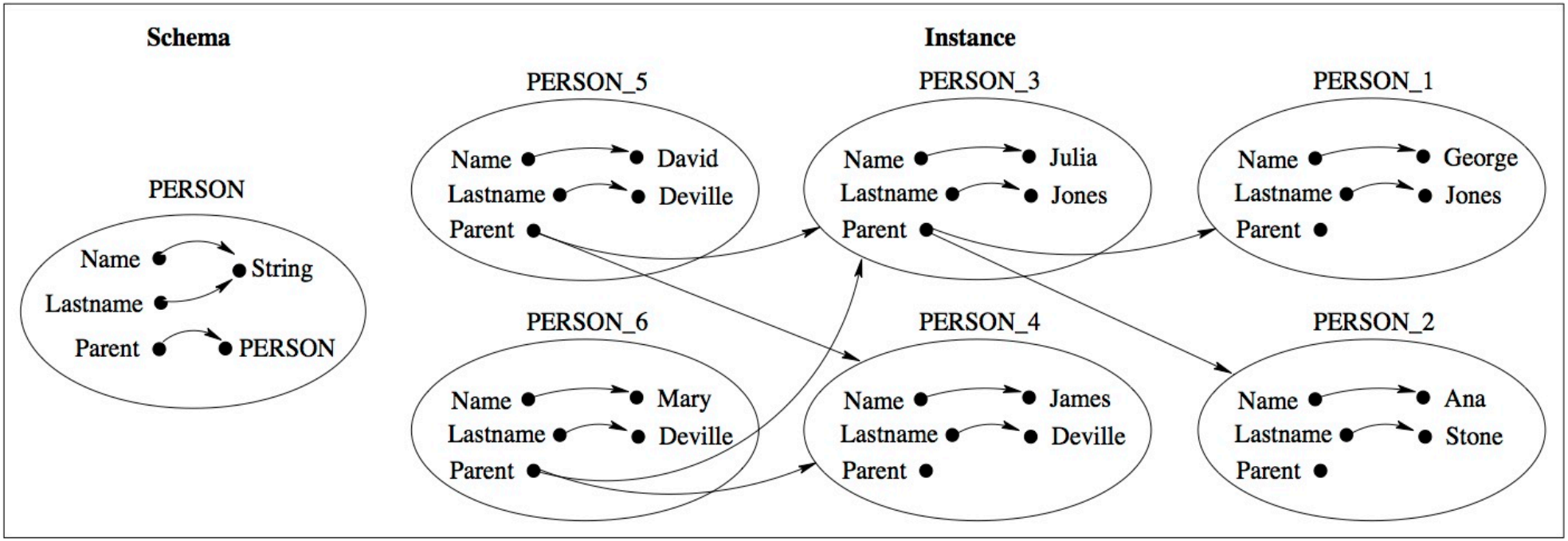




$$X = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} \text{ and}$$
$$E = \{e_1, e_2, e_3, e_4\} = \{\{v_1, v_2, v_3\},$$
$$\{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}.$$

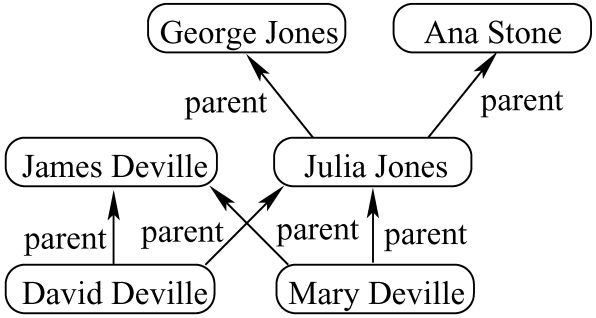
Hypernode

A hypernode is a directed graph whose nodes can themselves be graphs (or hypernodes), allowing nesting of graphs.



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

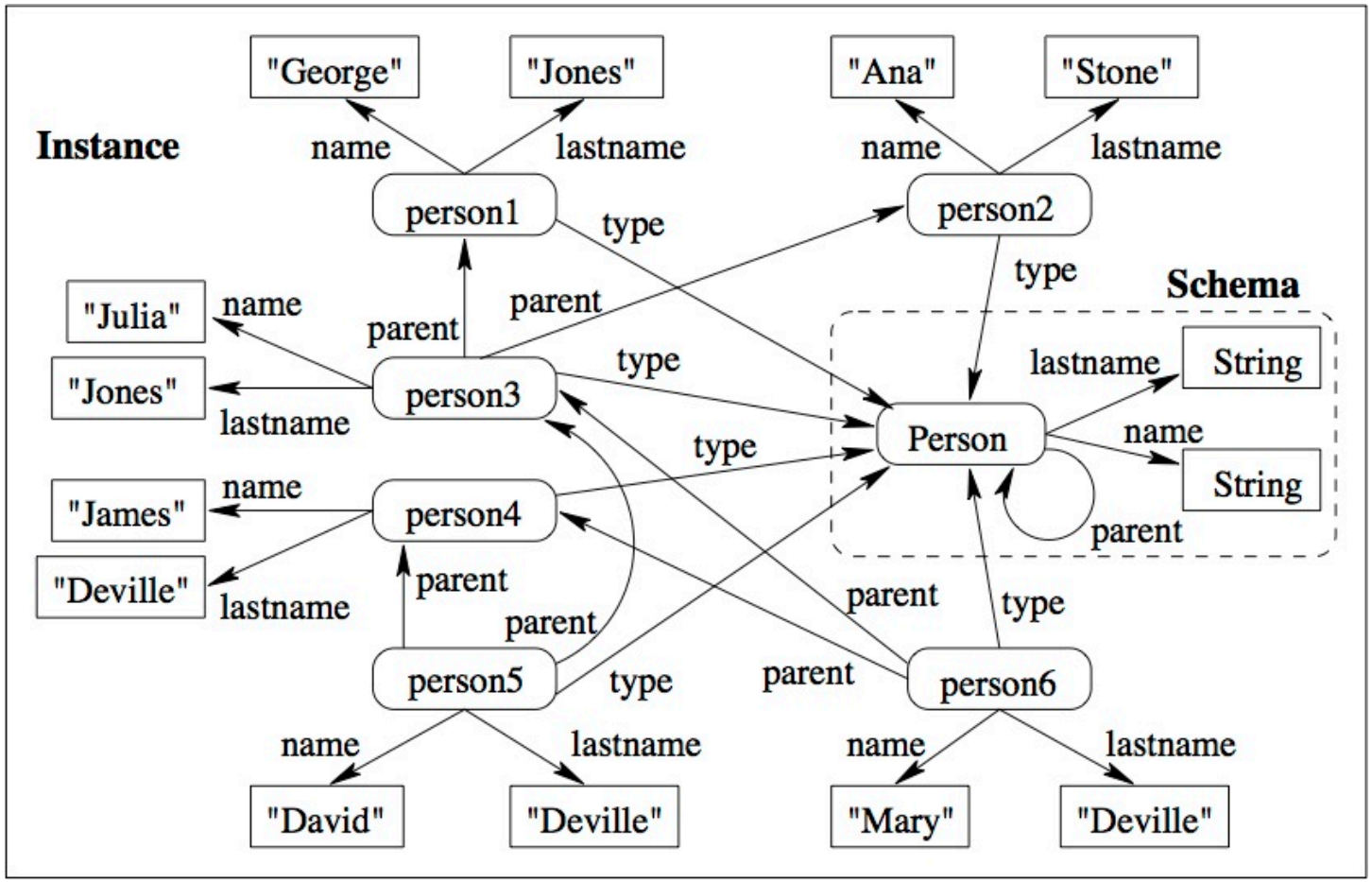
PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



Currently, two main graph models

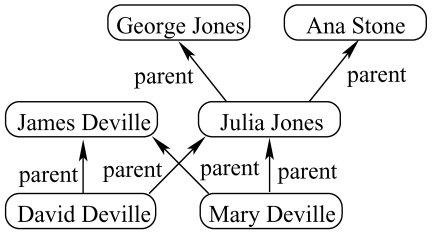
- RDF (triples)
- Property graphs

RDF



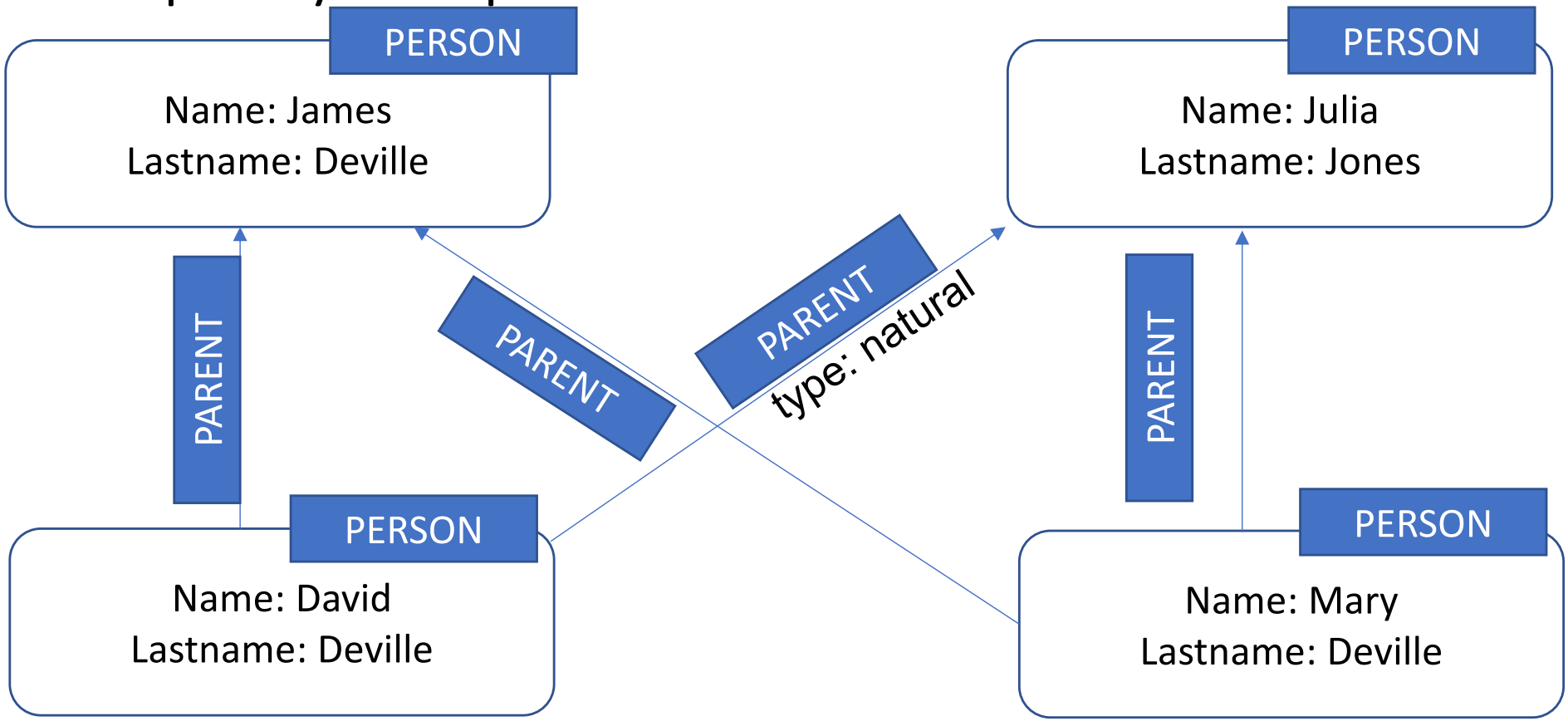
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

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



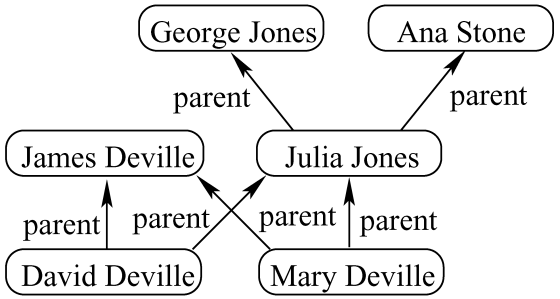
(subject, predicate, object) triples

Property Graph

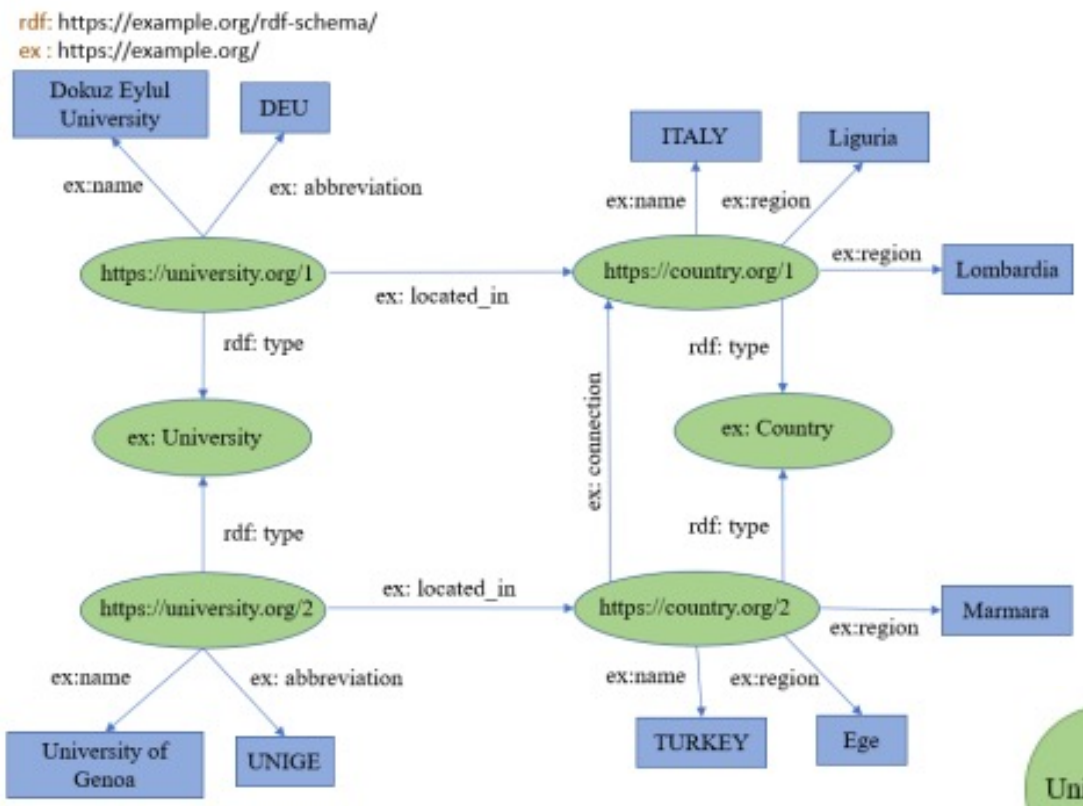


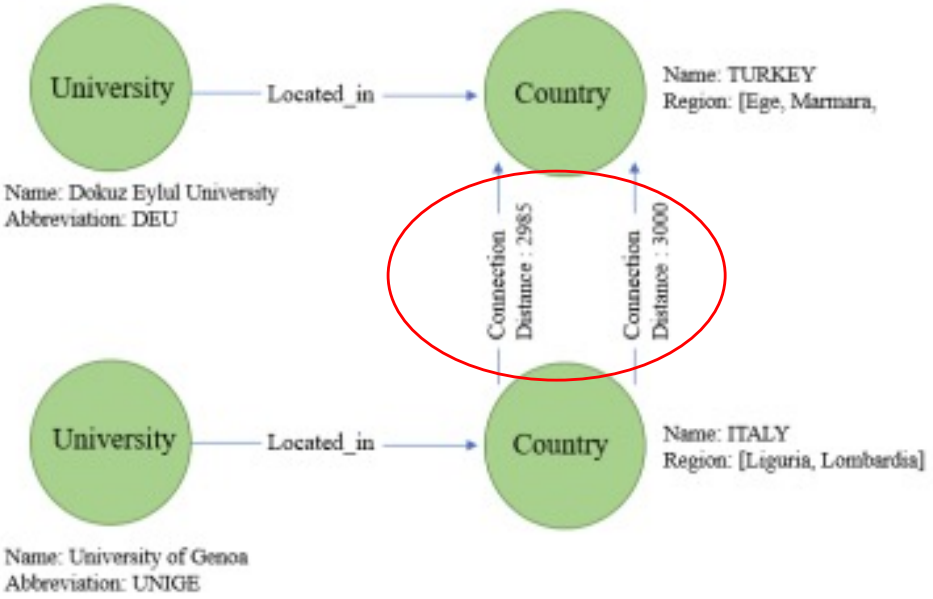
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

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



RDF & Property Graph

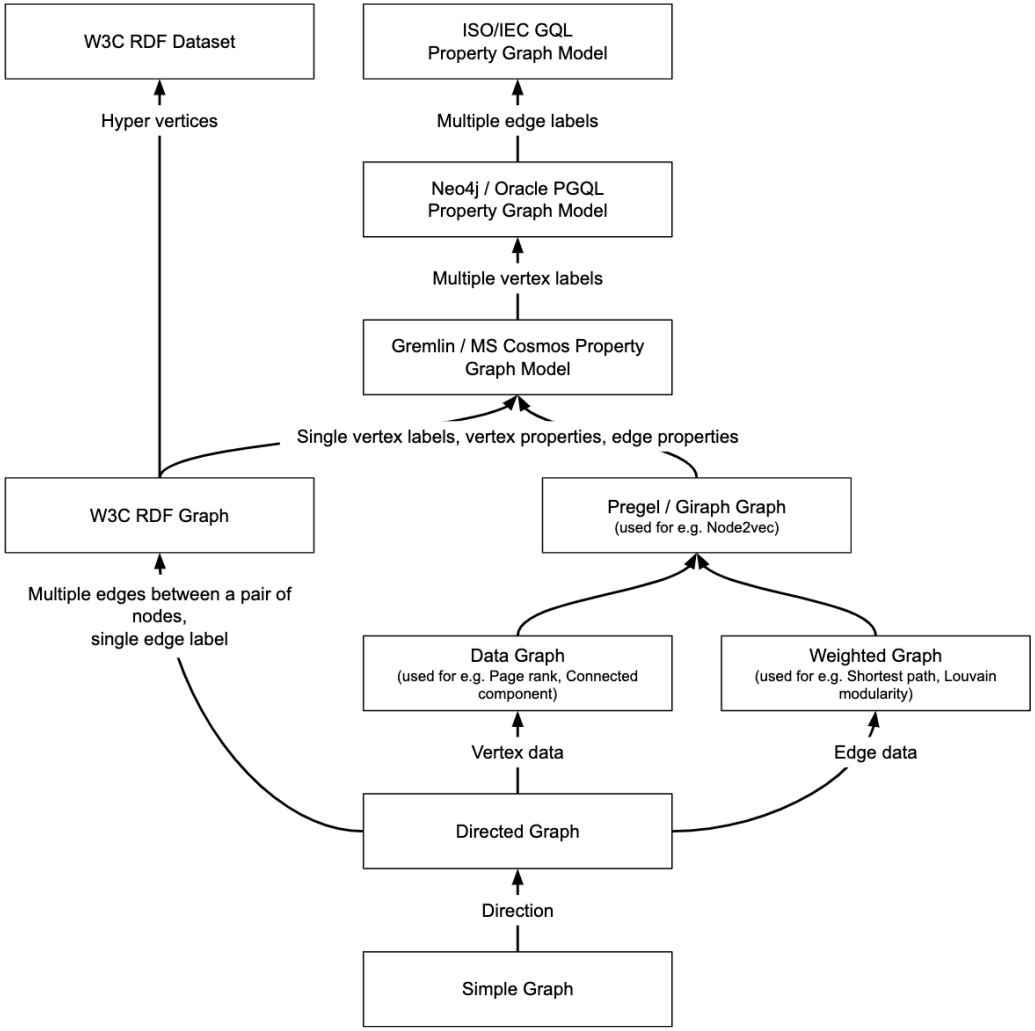




we'll discuss transformations in detail later (after introducing the constructs of the two data models)

A lattice of data models

- How expressive and human-friendly is a data model?
- A data model per use case
- Need of making different data models interoperable via mappings or direct translations



A.Bonifati

Property graph & data model interoperability

- Property Graph as an object model (neo4j, Apache Tinker pop & multimodel Azure Cosmos, OrientDB, Oracle Graph)
  - But we can view it as an extended relational model (graph as an indexing and access layer over relational data e.g. SAP Hanah graph, Tigergraph)
  - And as an access layer on top of RDF (BlazeGraph, StarDog)

  - See later discussion about interoperability

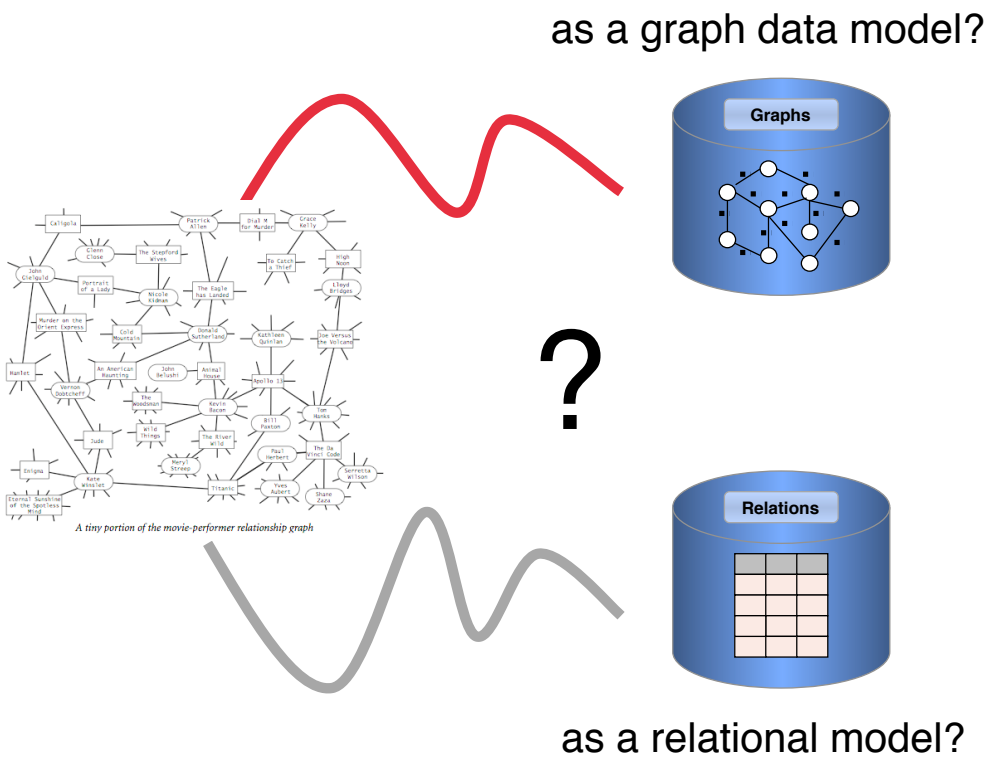
Graph Interchange Format

- GraphML
  - XML file containing a graph element, within which is an unordered sequence of node and edge elements
  - Each node element has a unique id
  - Each edge has source and target
  - Allows the specification of directed, undirected, hypergraphs, …
  - Originally designed for graph visualization
  - Supported by many graph-based systems (neo4j, tinkerpop, …)

Graph Query Languages

Graph Query Languages

PROPERTY	Neighborhoods	Adjacent Edges	Degree of a Node	Path	Fixed-length path	Distance	Diameter



Just a list of typical requests?

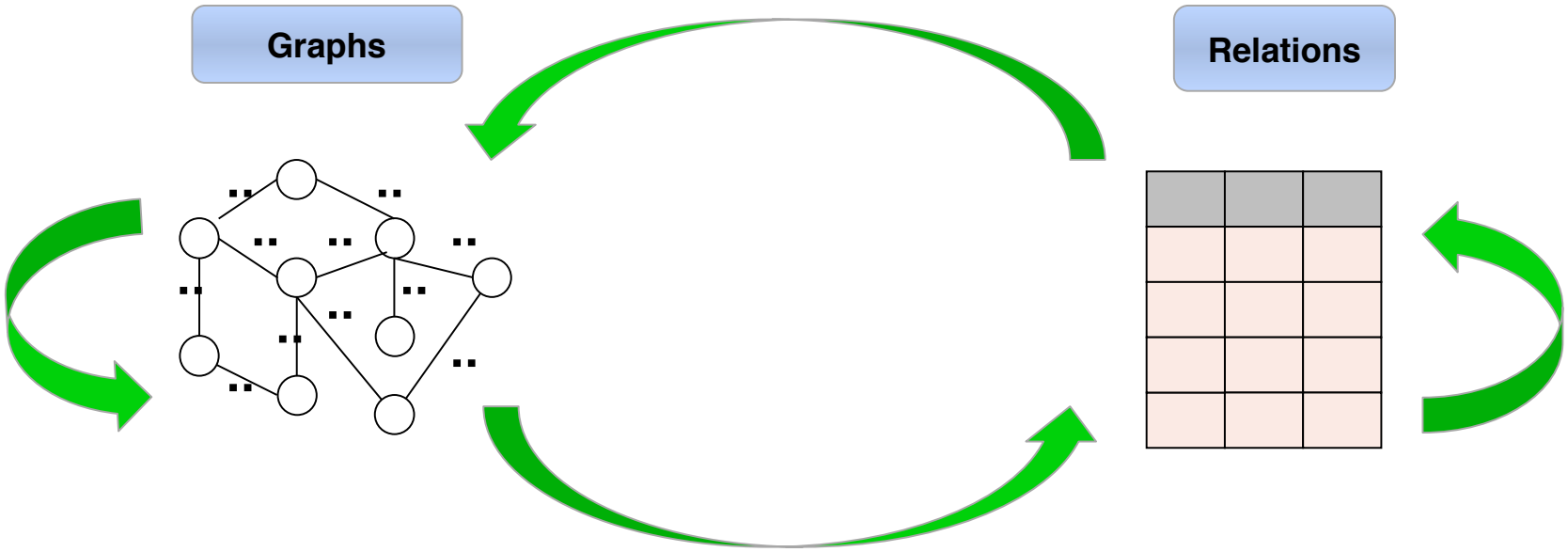
A. “Basic” Graph Queries

1. Pattern matching
2. Adjacency / neighborhood
3. Reachability / connectivity
4. Summarization
5. …

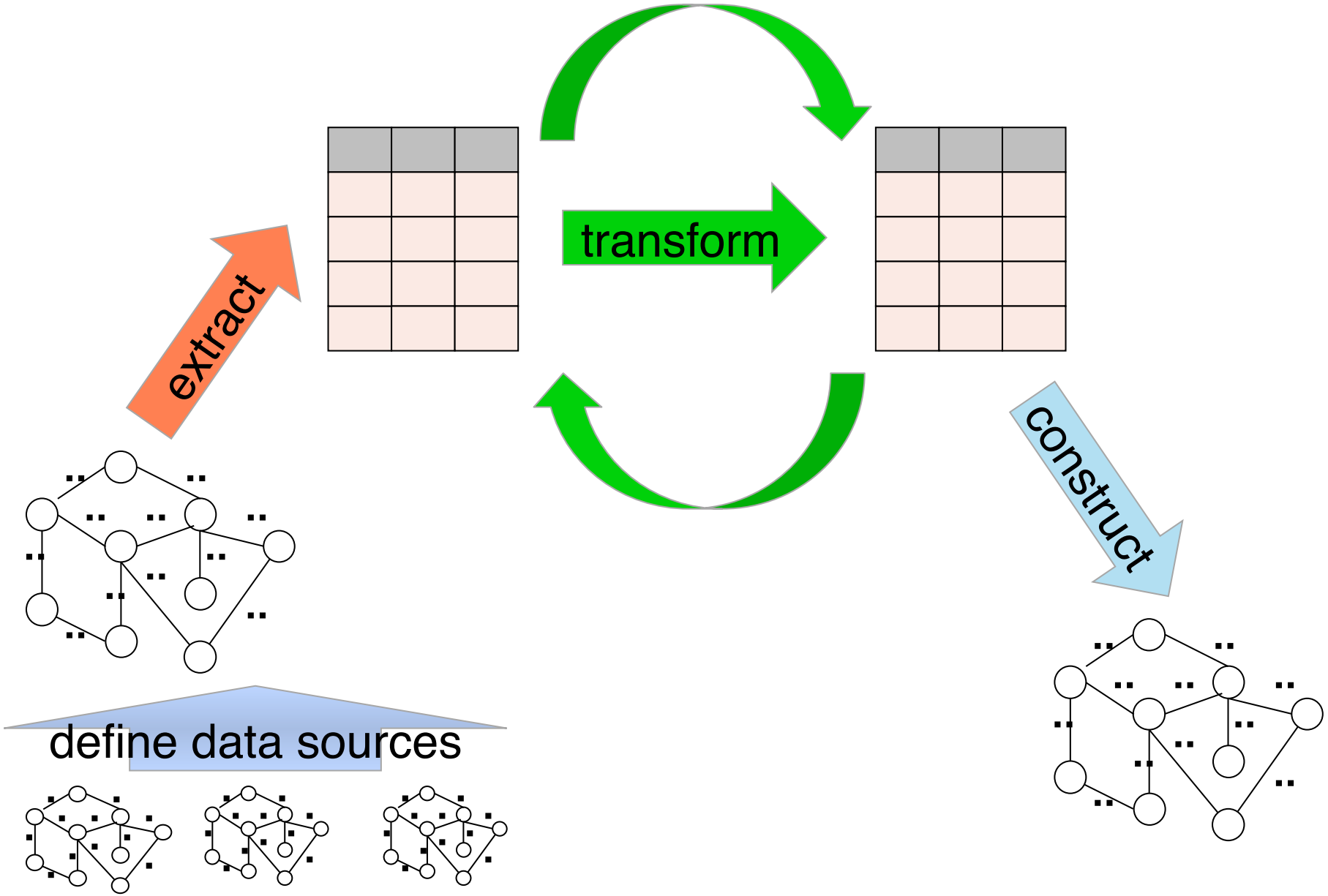
B. Analytical Queries

1. Centrality measures
2. Diameter and other global properties
4. Graph properties and parameters
5. …

Graph Query Languages

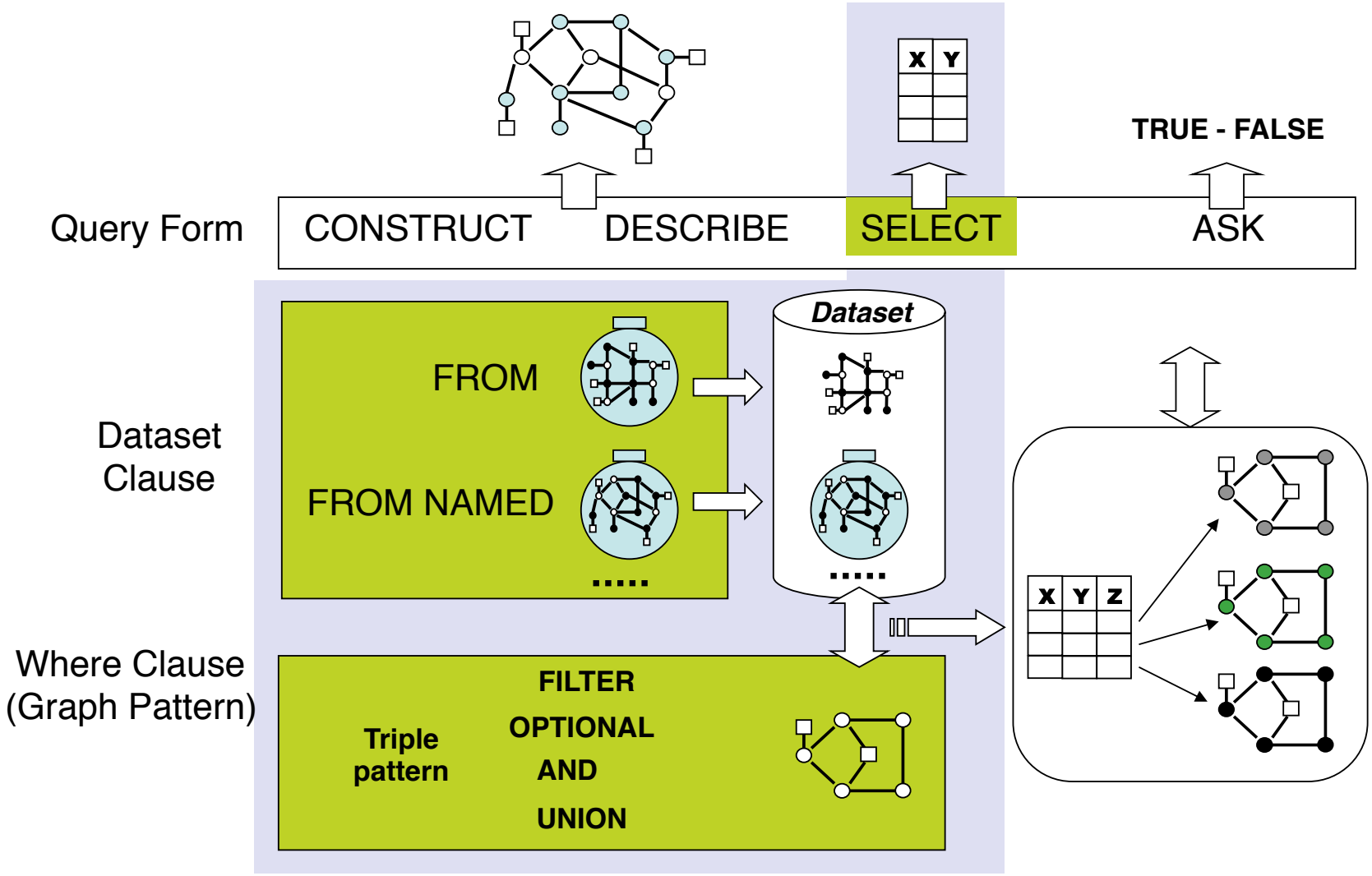


Graph Query Languages



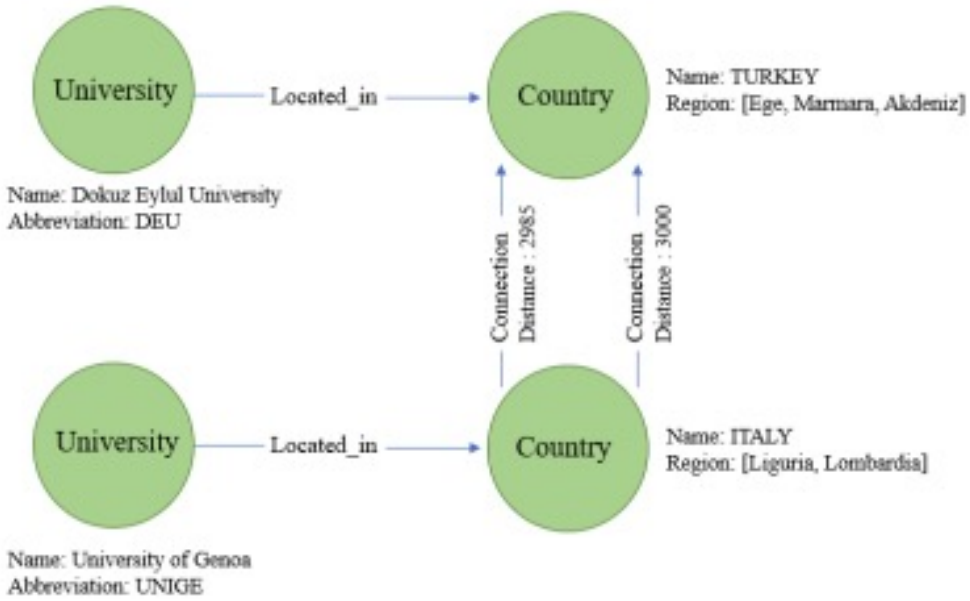
Language	Define Source	Extract	Transform	Construct
SQL	FROM		WHERE	SELECT
SPARQL	FROM, Service	pattern matching	operators	Select, ASK, Contruct,
Cypher		MATCH	WHERE	RETURNS

SPARQL Query



Cypher - bindings

```
MATCH (u:University)-[:located_in]->(c:Country)
RETURN c.name
```



u	c
University Name: Dokuz Eylul University Abbreviation: DEU	Country Name: TURKEY Region: [Ege, Marmara, Akdeniz]
University Name: University of Genoa Abbreviation: UNIGE	Country Name: ITALY Region: [Liguria, Lombardia]

GQL – Graph Query Language

https://www.gqlstandards.org

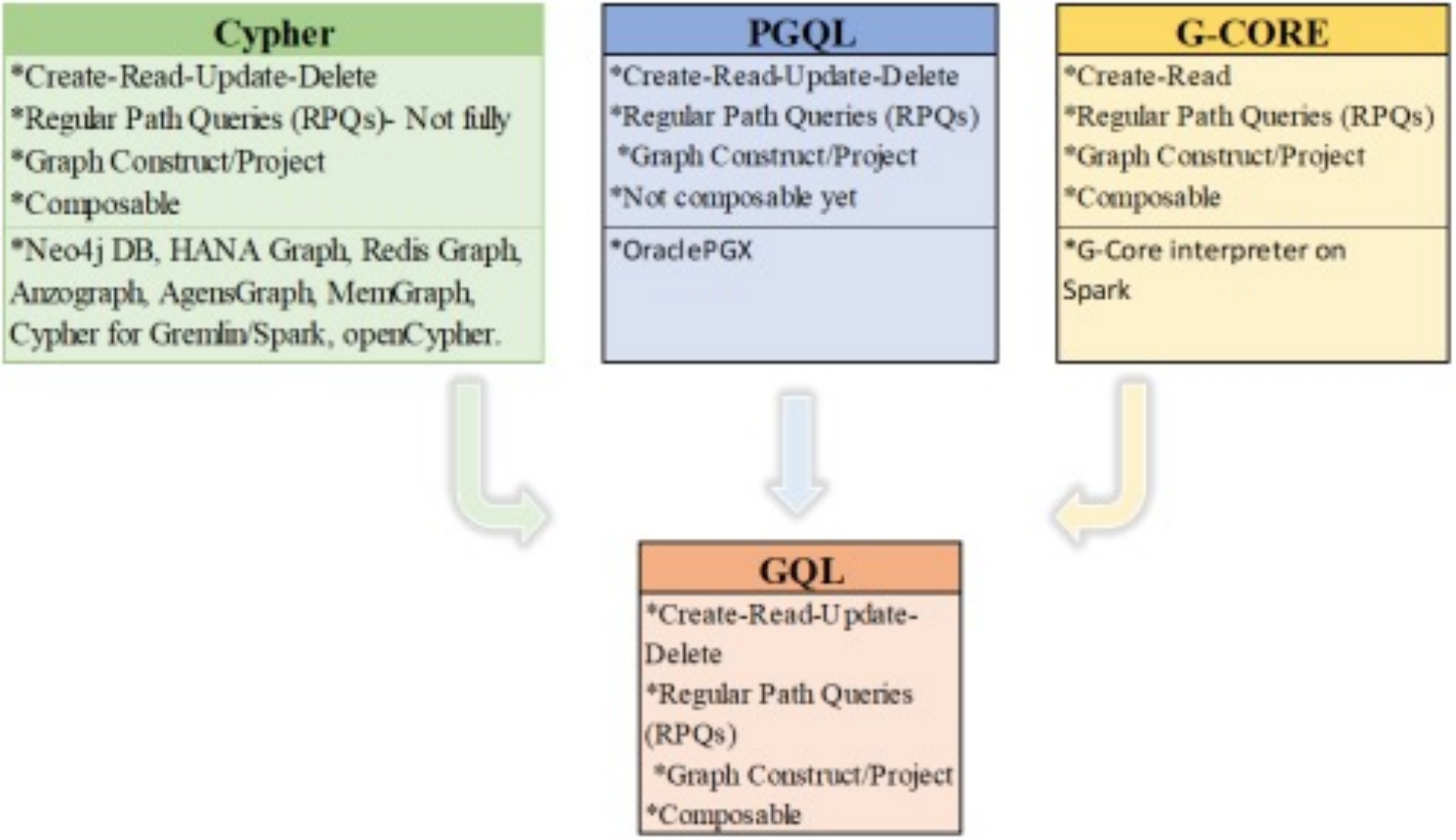
Ongoing ISO effort

Standalone graph query language to complement SQL

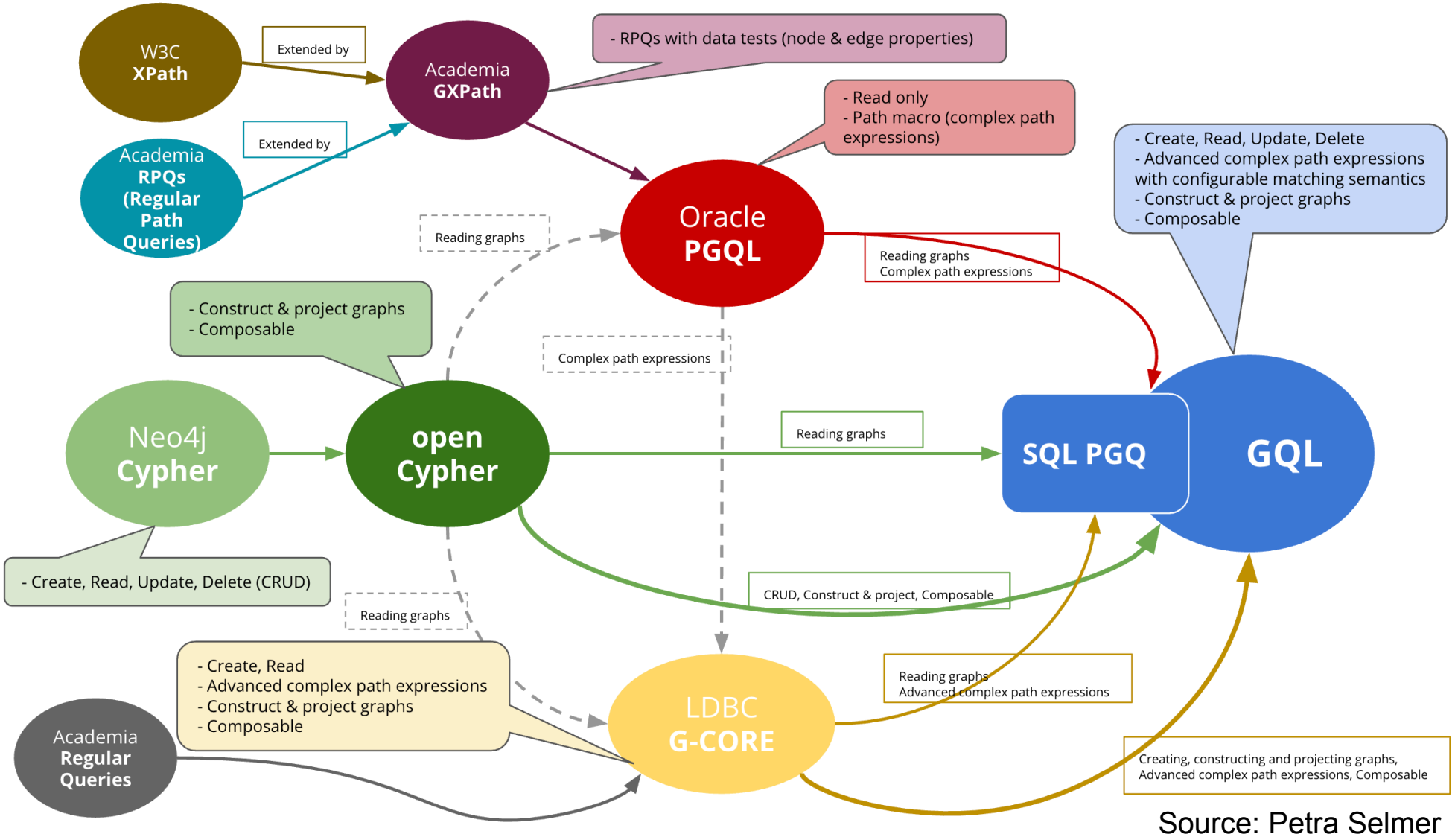
Manifesto https://gql.today

Since 2017 work has been proceeding on extending SQL with read-only property graph extensions based on the pattern-matching paradigm of Cypher and PGQL. SIGMOD 2017 saw the publication of the future-looking G-CORE paper on fresh directions in PG querying, matched by implementation of compositional queries and graph views in Cypher for Apache Spark. Since spring 2018 the property graph world has been coalescing around the idea of a single GQL language, drawing on all of these precedents, open to other inputs, and closely coordinated with key aspects of SQL and its ecosystem.

GQL – Towards a standard



GQL



Source: Petra Selmer

Programmatic interfaces

- In addition to query languages, graphs are accessed through programmatic interfaces
  - Proprietary or common APIs (mostly Tinkerpop)
  - Scripting and processing languages (Apache Gremlin, SAP GraphScript)
  - MapReduce, Spark and other processing framework

Gremlin – Queries as Graph Traversals

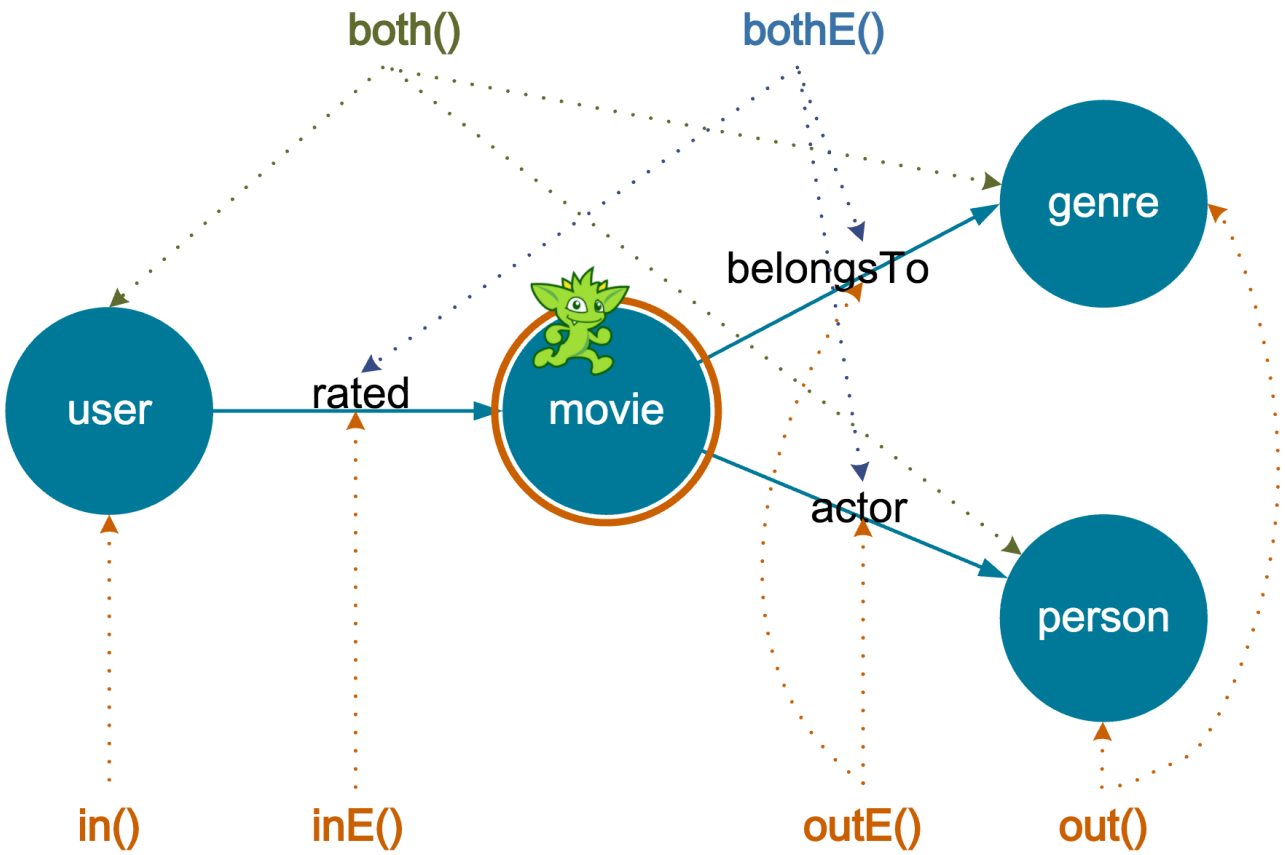
- Traversing means moving from one node to another along the relationship edges
  - As a node can have more than one relationship, traversal is not trivial
  - There are algorithms that try to optimise the traversal of a graph

- A graph traversal starts with a chosen node, either a specified root, or any given node
  - It can follow INCOMING or OUTGOING nodes, so go in either direction
  - Can traverse DEPTH_FIRST or BREADTH_FIRST

- Useful for asking “Who rated movie A?” or “Which movie B acted in?”

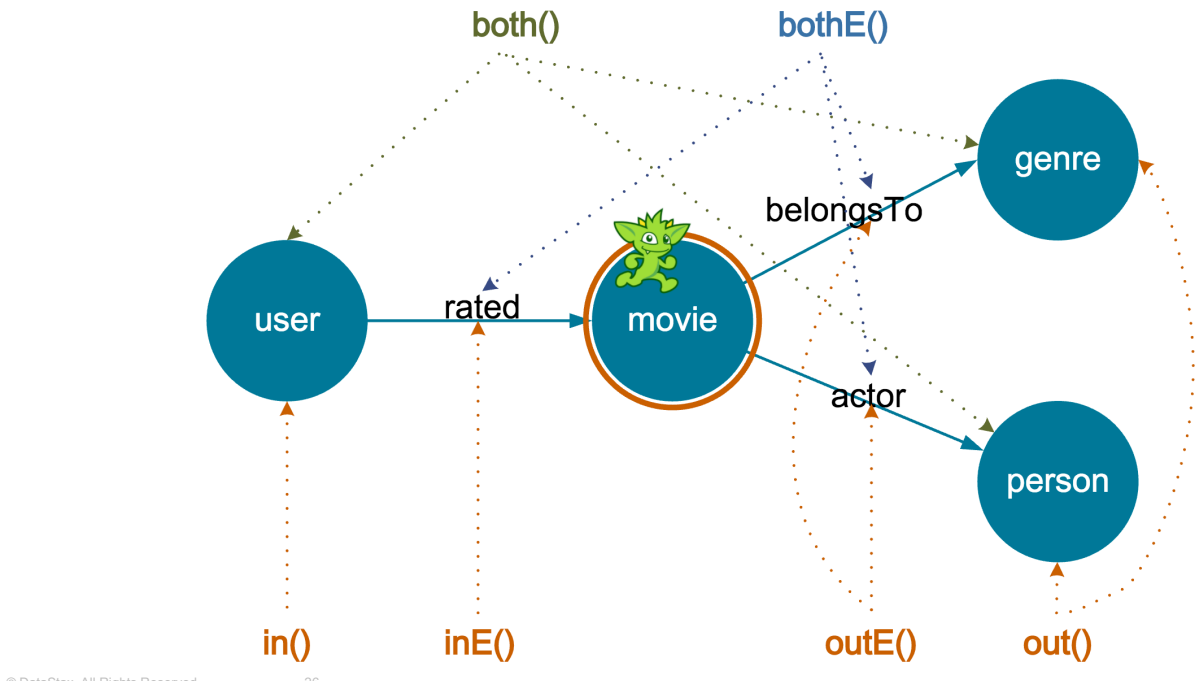
Gremlin – Simple Traversal Steps

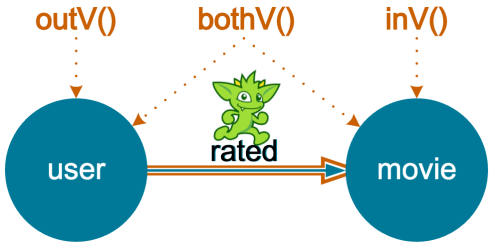
Simple Traversal Steps



Gremlin – Simple Traversal Steps

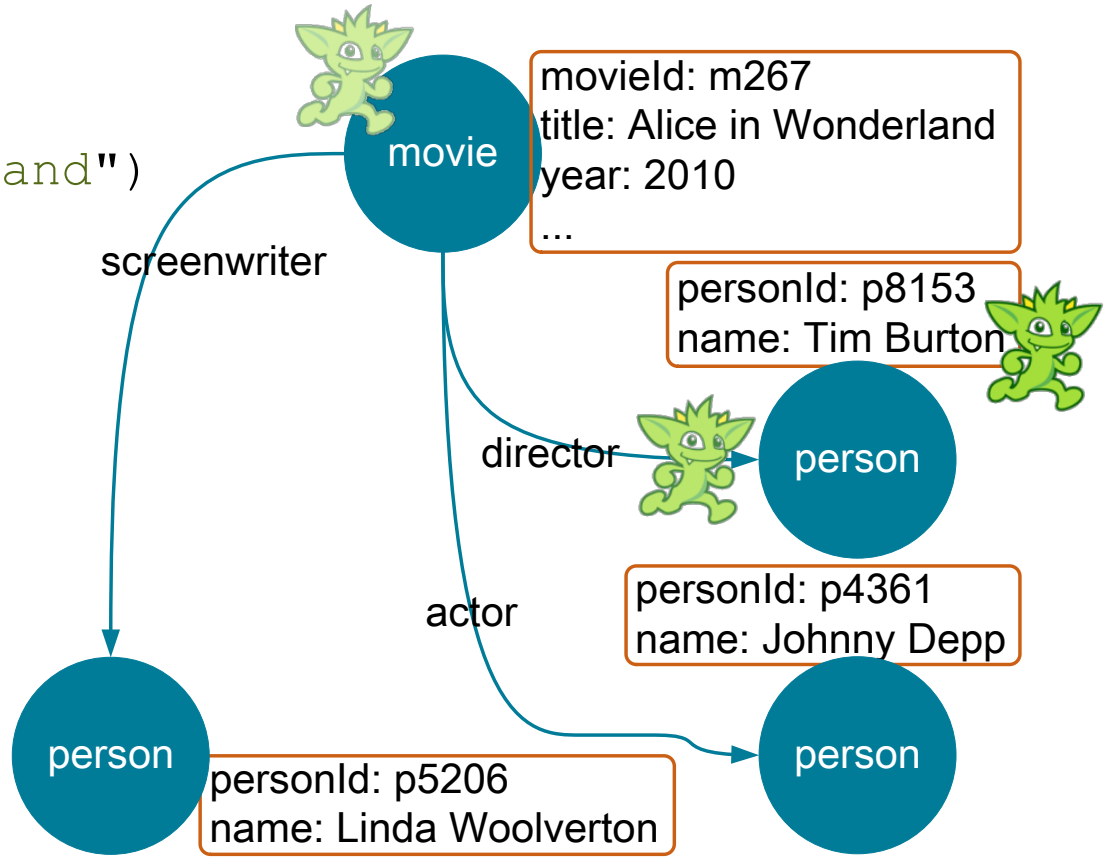
Simple Traversal Steps





Gremlin – Simple Traversal Steps

```
g.V().has("title",
       "Alice in Wonderland")
     .has("year",2010)
     .out("director")
     .values("name")
```



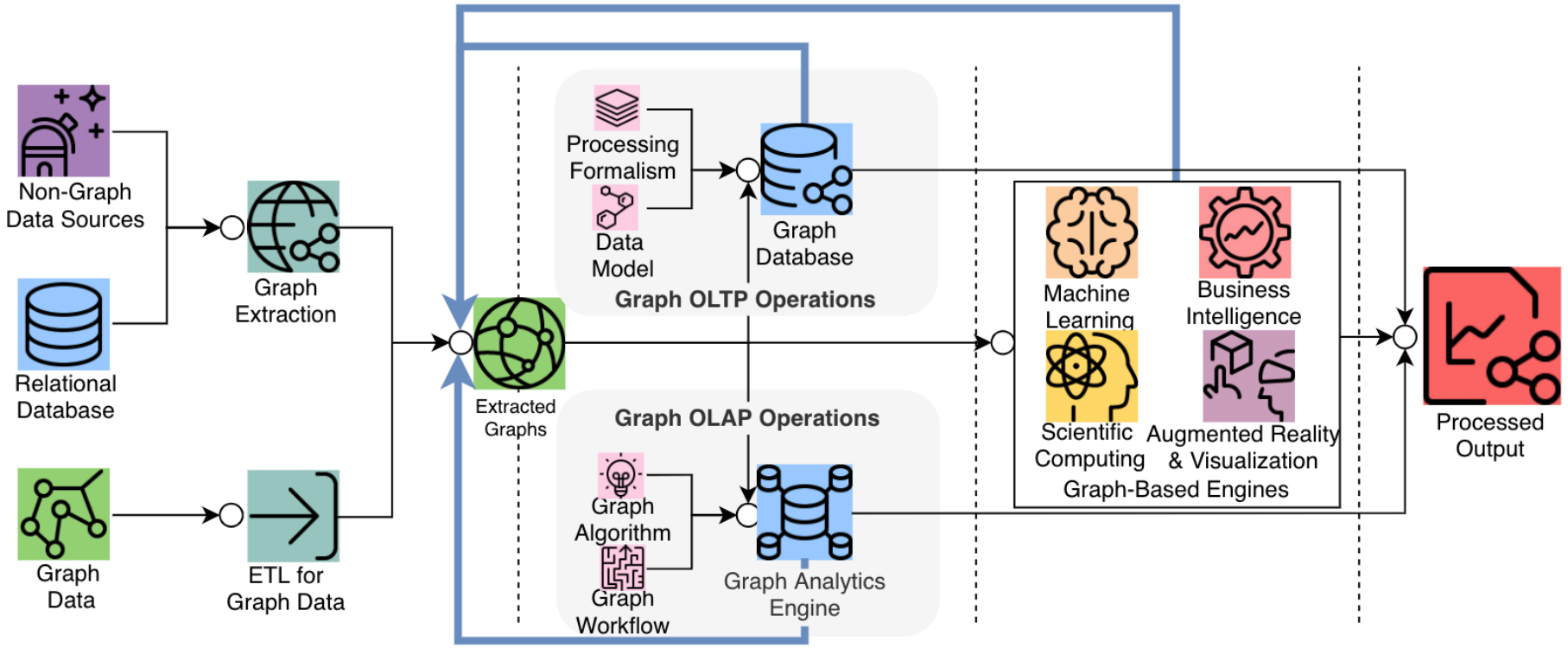
Gremlin - Graph Traversal Steps

Lambda steps	*map*	*flatMap*	*filter*	*sideEffect*	*branch*
Derived steps	*id*, *label*, *match*, *path*, *select*, *order*, …	*coalesce*, *in*, *inE*, *inV*, *out*, …	*and*, *coin*, *has*, *is*, *or*, *where*, …	*aggregate*, *inject*, *profile*, *property*, *subgraph*, …	*choose*, *repeat*, *union*, …
Other steps	*barrier*, *cap*, …				
Step modulators	*as*, *by*, *emit*, *option*, …				
Predicates	*gt*, *eq*, *lt*, *neq*, *within*, *without*, …				

Graph Databases and Systems

Towards a complex data ecosystem

- Several functionally different steps integrating OLTP/OLAP, from data source injection to graph database functionalities and advanced processing (ML, BI, Augmented Reality)

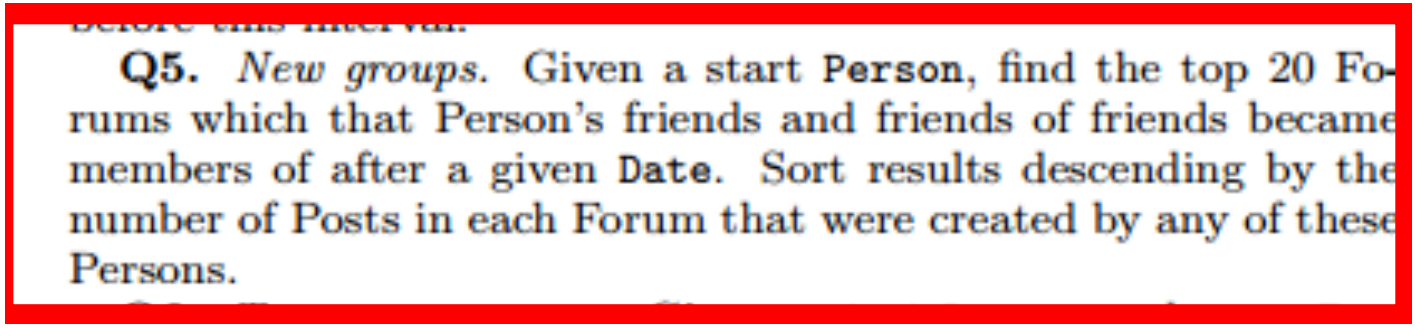


Operations on Graphs

- Online query processing
  - Shortest path query
  - Subgraph matching query
  - SPARQL query
- Offline graph analytics
  - PageRank
  - Community detection
- Other graph operations
  - Graph generation, visualization, interactive exploration, etc.

Interactive Workload

- Online ACID features and scalability
- System in a steady state, providing durable storage
- Updates are typically small
- Updates will conflict a small percentage of the time
- Queries typically touch a small fraction of the database



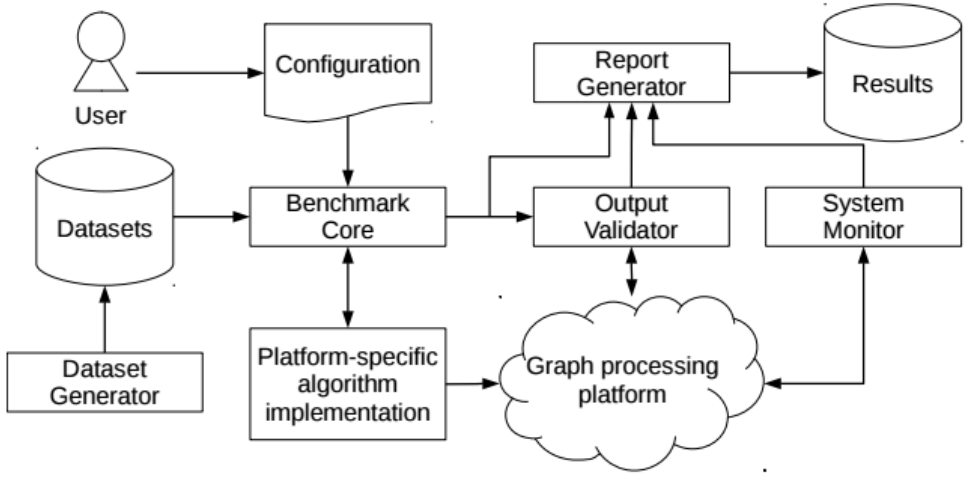
Q5. *New groups.* Given a start `Person`, find the top 20 Forums which that Person's friends and friends of friends became members of after a given `Date`. Sort results descending by the number of Posts in each Forum that were created by any of these Persons.

Business Intelligence Workload

- The workload stresses query execution and optimization
- Queries typically touch a large fraction of the data
- The queries are concurrent with trickle load
- The queries touch more data as the database grows

Graph Analytics Workload

- The analytics is done on most of the data in the graph as a single operation
- The analysis itself produces large intermediate results
- No need for isolation from possible concurrent updates



Systems

- **Graph database** systems
  - e.g. Neo4j, InfiniteGraph, DEX, Titan
- **Graph programming frameworks**
  - e.g. Giraph, Signal/Collect, Graphlab, Green Marl, Grappa
- **RDF** database systems
  - e.g. OWLIM, Virtuoso, BigData, Jena TDB, Stardog, Allegrograph
- **Relational** database systems
  - e.g. Postgres, MySQL, Oracle, DB2, SQLServer, Virtuoso, MonetDB, Vectorwise, Vertica
- **noSQL** database systems
  - e.g. HBase, REDIS, MongoDB, CouchDB, MapReduce systems like Hadoop

Workload by systems

System	Interactive	Business Intelligence	Graph Analytics
Graph databases	Yes	Yes	Maybe
Graph programming frameworks	-	Yes	Yes
RDF databases	Yes	Yes	-
Relational databases	Maybe	Maybe	Maybe, by keeping state in temporary tables, and using the functional features of PL-SQL
NoSQL Key-value	Maybe	Maybe	-

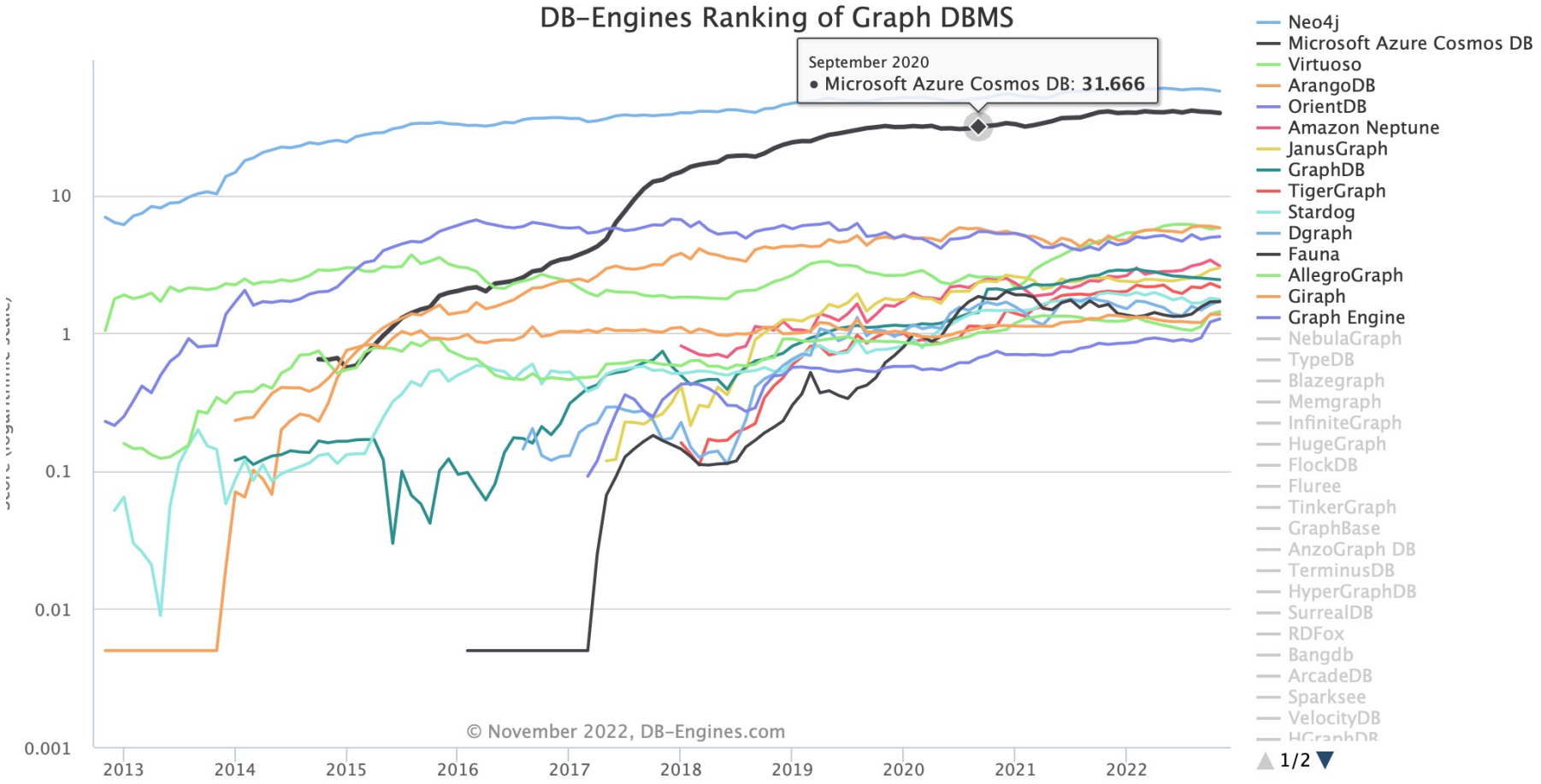
Graph Databases

1. Address need of managing graph data
2. Architecture/goals inspired by classical DBMS
3. Persistent storage of graph data
4. Transactionality
5. Closed world
6. Efficiency (over scalability)
7. () Portability (of data)
8. () Declarative query languages

	Data model				Storage method		Query facilities			Computing model	
Graph database	Simple graph	Property graph	Hypergraph	Nested graph	Native	Non-native	Query language	API	Graph algorithms	Single-node	Distributed
AllegroGraph	●				●		●	●	●	●	
ArangoDB		●			●	●	●	●		●	
Bitsy		●			●			●	●	●	
Cayley		●			●	●	●			●	●
FlockDB	●					●		●			●
GraphBase	●				●		●	●		●	
Graphd	●				●		●			●	
Horton		●				●	●				●
HyperGraphDB			●		●	●	●	●		●	●
IBM System G		●			●	●		●		●	●
imGraph	●				●			●			●
InfiniteGraph		●			●		●	●	●	●	●
InfoGrid	●				●	●		●		●	●
Neo4j		●			●		●	●	●	●	●
OrientDB		●				●	●	●	●		●
Sparksee/DEX		●			●			●	●	●	
Titan		●			●	●	●	●		●	●
Trinity		●	●			●	●	●	●		●
TurboGraph		●			●			●	●	●	

Graph database engine

- The number of graph engines is growing over the years as well as their popularity



Graph Processing Frameworks

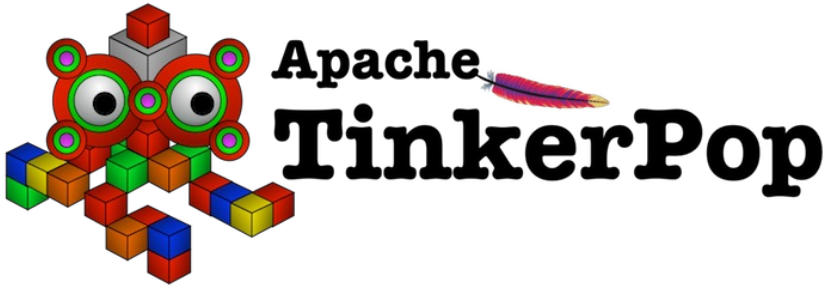
- (Offline Graph Analytical Systems)
  1. Batch processing
  2. Analysis of large graphs
  3. Facilities for graph analytical algorithms
  4. Distributed environment
  5. Multiple machines
  6. API or programming as user access

Graph Processing Frameworks

- Pregel
- ApacheGiraph
- GraphLab
- CatchtheWind
- GPS
- Mizan
- PowerGraph
- GraphX
- TurboGraph
- GraphChi

Spark Graph Fames

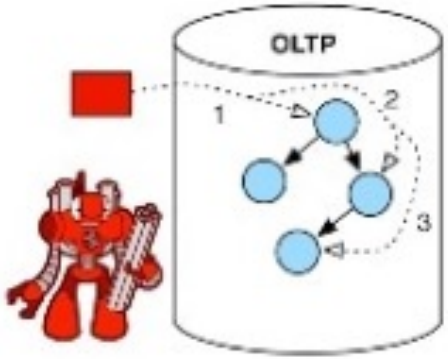
Interoperabilty
Apache Tinkerpop

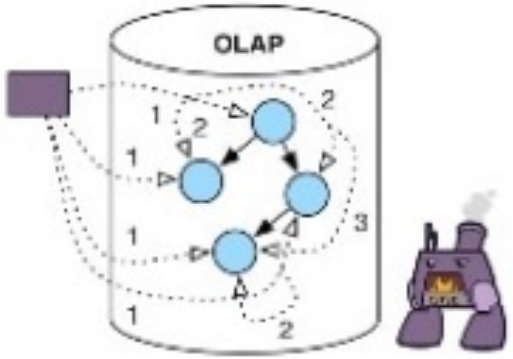


T
p
S
P

Vendor
- DataStax DSE Graph
- Azure Cosmos DB
- Neo4j
- OrientDB
- Stardog

Open Source
- Apache S2Graph
- HGraphDB
- JanusGraph
- TinkerGraph
- UniPop





Open Source (OLAP)
- Apache Giraph
- Apache Spark

To sum up …

- Graph data management has a bright future
- One size does not fit all: Need different GDB for small, medium and web scale and for different workloads
- Very diverse use cases, standardization attempts for query languages ongoing
- What matters in a graph is its **topology** [if you do not need it, stay in the relational world]
- Need better interoperation between relational (tables) and graph data: GQL
- Interoperability among graph systems: Apache Tinkerpop

References

- Angles, Renzo, and Claudio Gutierrez. "An introduction to graph data management." *Graph Data Management*. Springer, Cham, 2018. 1-32.
- https://arxiv.org/abs/1801.00036

Slides Credits

- Introduction to Graph Data Management, Claudio Gutierrez, EDBT Summer School 2015
- Peter Boncz, Graph Benchmarking, EDBT Summer School 2015