Big Graph Processing Systems

Angela Bonifati, Lyon 1 University & CNRS Liris (France)

eBISS Summer School 2022 (Cesena, Italy)

About myself

- Professor at Lyon 1 University (France)
- Leader of the Database group at Liris CNRS lab (France)
- Adjunct Professor at the University of Waterloo (Canada)
- Co-authored several publications in major venues of the data management field, two books and an invited paper in ACM Sigmod Record 2018
- Program Chair of ACM Sigmod 2022 and currently Chair of the EDBT Executive Committee (2020-2024).
- Program Director of the International Master DISS (Data and Intelligence for Smart Systems) at Lyon 1 University

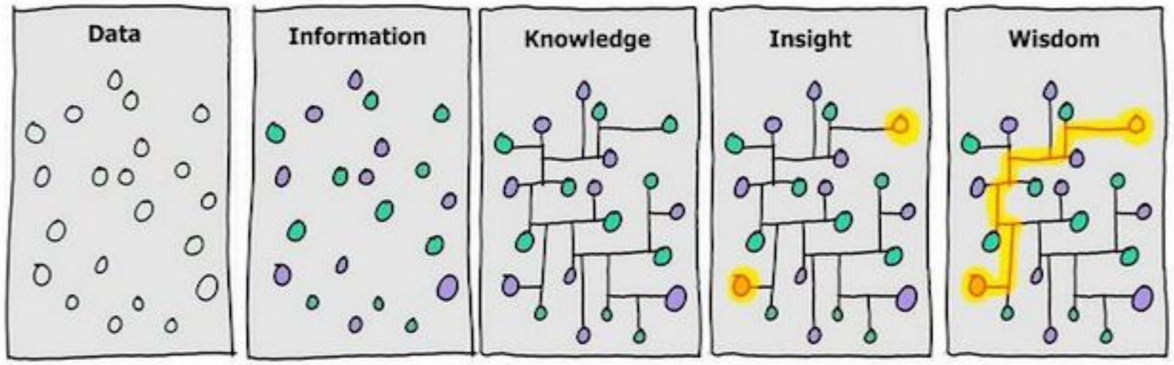
Graphs are everywhere!

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.]

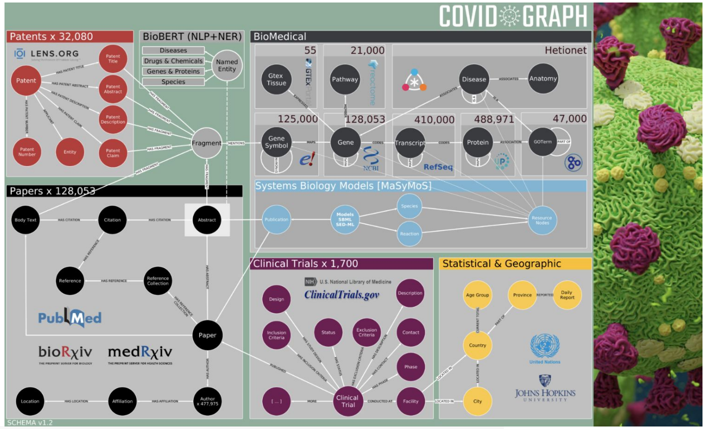
Graphs as unifying abstractions

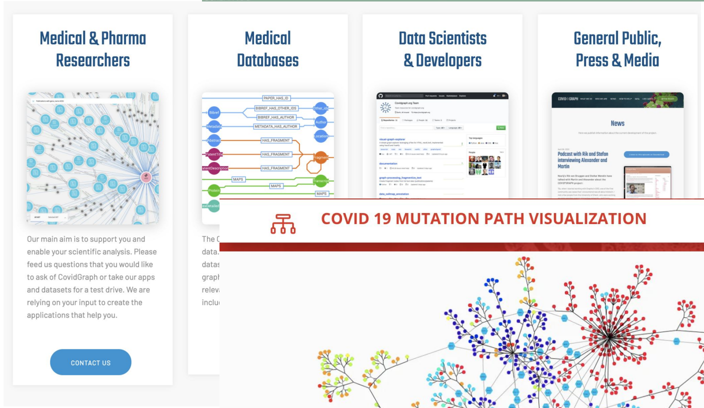
- Graphs are natural abstractions for representing interconnected objects when encoding, explaining and predicting real-world and digital-world phenomena.
- Graphs are underpinning several data management ecosystems, in societal, scientific, RDF, product and digital domains according to a recent user survey [Sahu20].
- Nevertheless, the data models, query languages and system requirements needed for graphs are constantly evolving.

[Sahu20] Siddhartha Sahu et al:The ubiquity of large graphs and surprising challenges of graph processing: extended survey. VLDB J. 29(2-3): 595-618 (2020)

A plethora of applications

- 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




Medical & Pharma Researchers

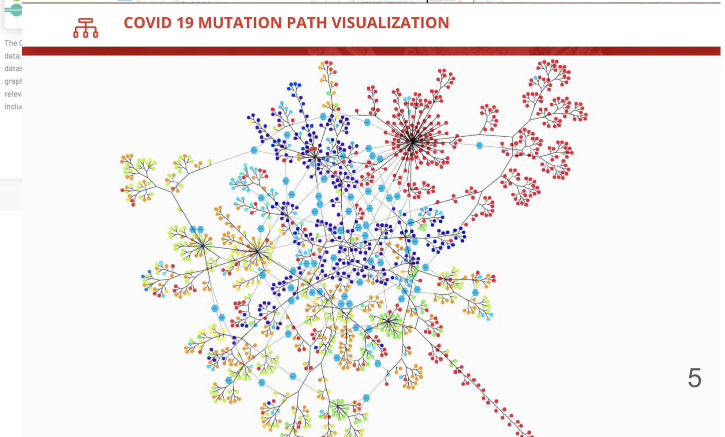
Medical Databases

Data Scientists & Developers

General Public, Press & Media

Our main aim is to support you and enable your scientific analysis. Please feed us questions that you would like to ask of CovidGraph or take our apps and datasets for a test drive. We are relying on your input to create the applications that help you.

CONTACT US

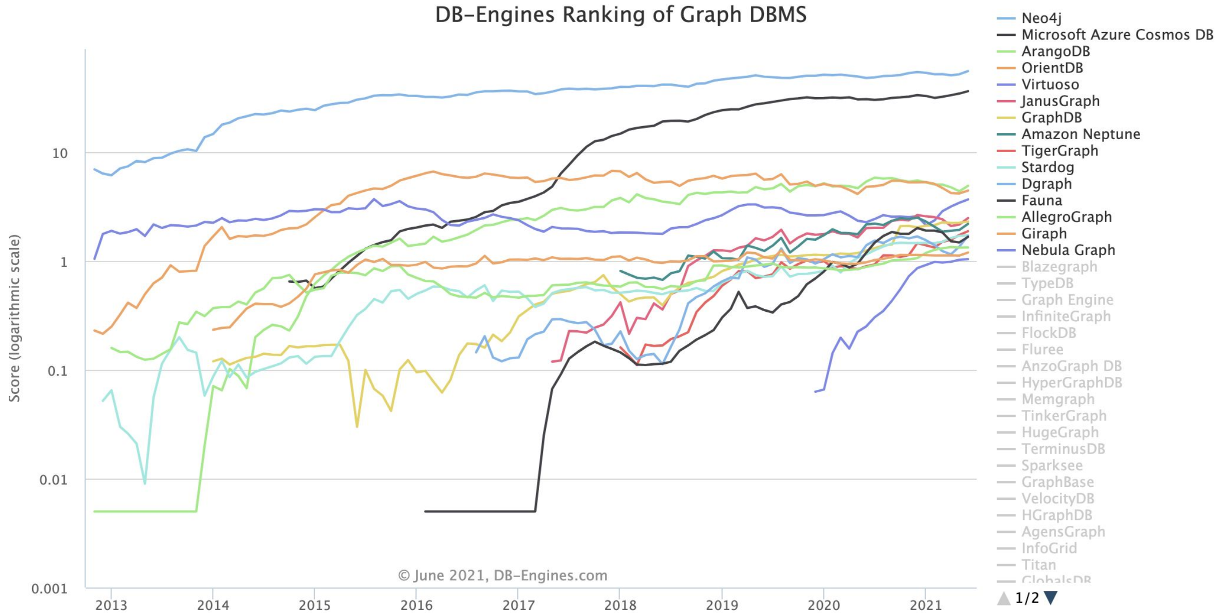


Web-scale Graph Processing

- Google PageRank: Shift from MapReduce to Vertex-centric computation
  - Both distributed but the latter improves locality, while providing linear scalability and achieving better performances
- Other systems support more elaborate computational models (task-based instead of distributed i.e. Facebook and Apache Giraph)

Several graph database engines on the rise

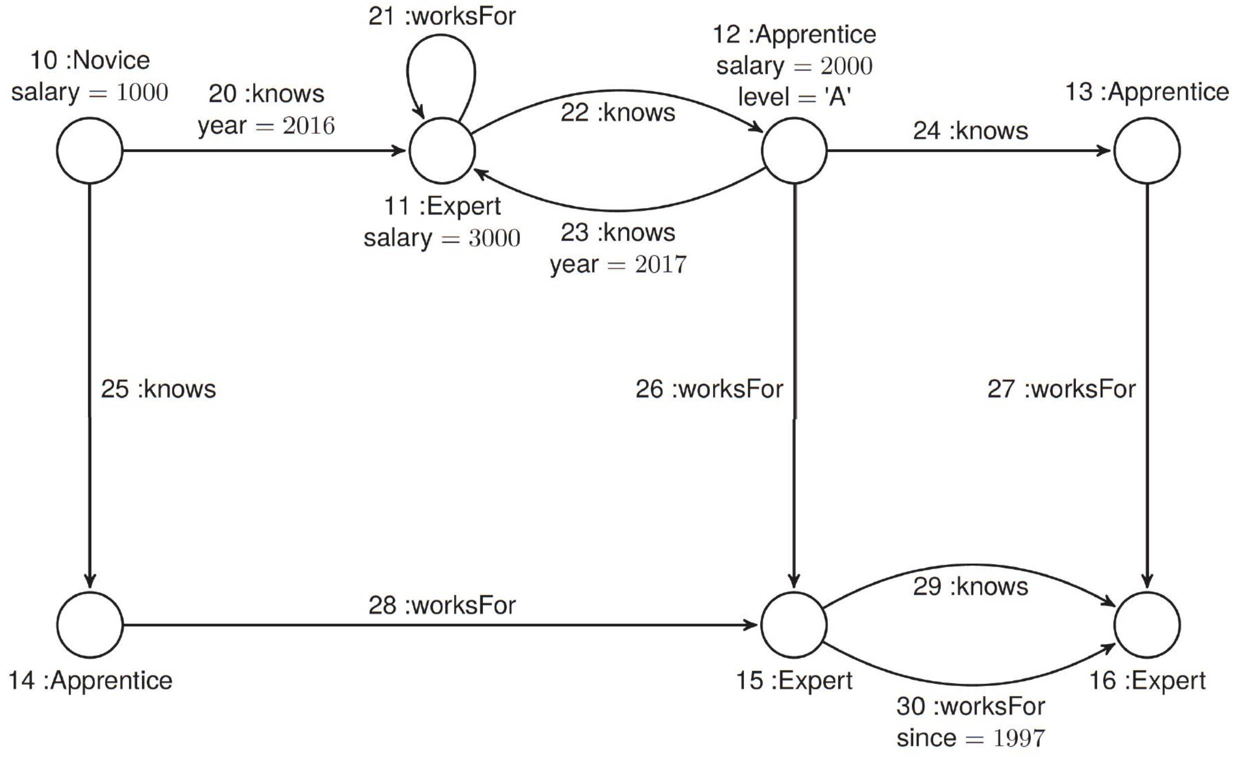
- The number of graph engines is growing over the years as well as their popularity



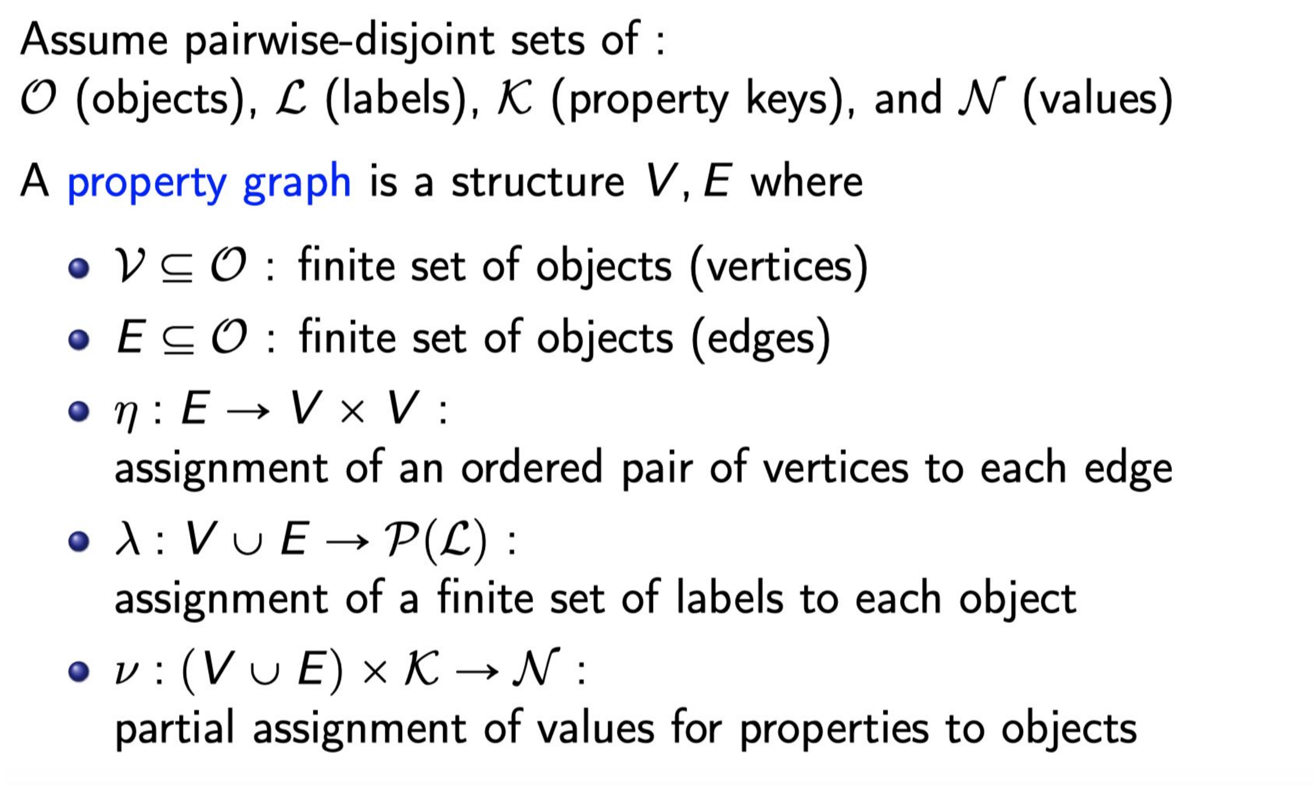
Expressivity of the graphs/queries

- Dependence on the chosen data model and on how do humans conceptualize graphs
- Interoperability issues (due to multiple heterogeneous data sources) are to be taken into account
- A data model lattice to navigate across data models, balancing understandability and expressive power
- New algebraic frameworks needed to account for an increasing variety of graph workloads

Property graph example



Property graphs



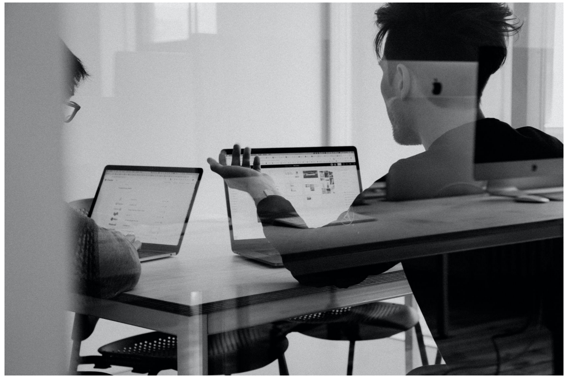
Assume pairwise-disjoint sets of :
$\mathcal{O}$ (objects), $\mathcal{L}$ (labels), $\mathcal{K}$ (property keys), and $\mathcal{N}$ (values)

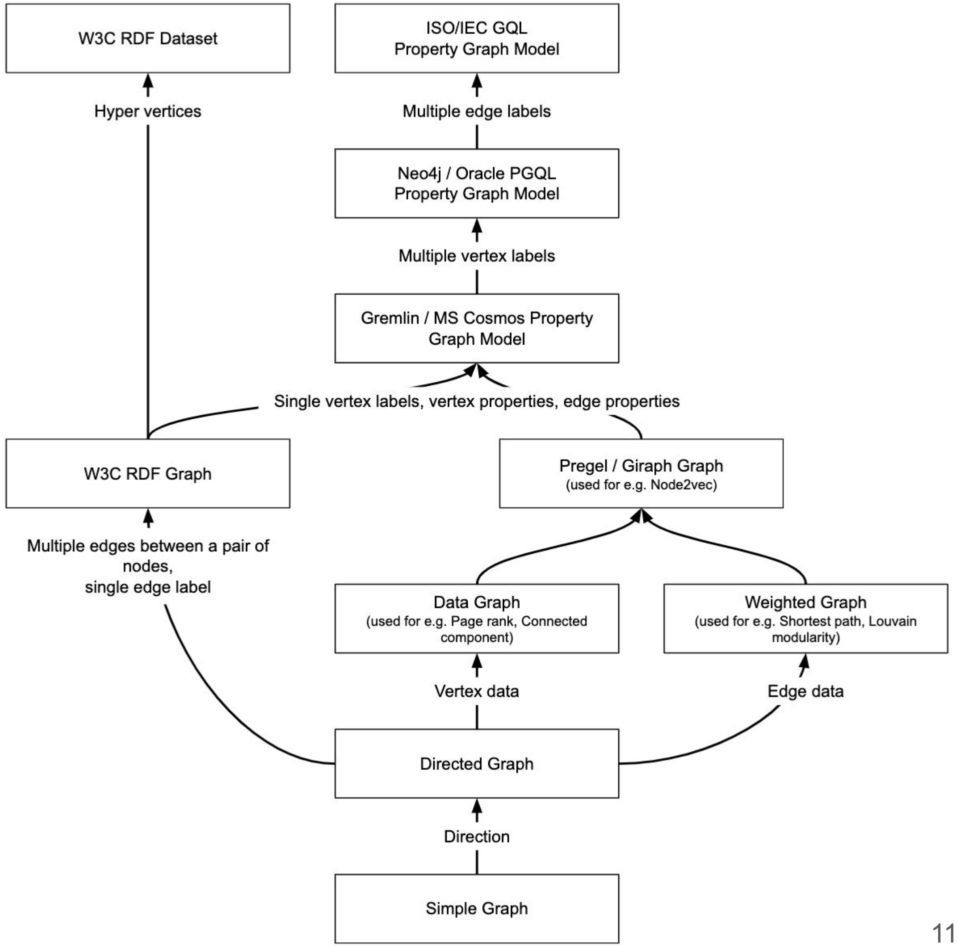
A property graph is a structure V, E where

- $\mathcal{V} \subseteq \mathcal{O}$: finite set of objects (vertices)
- $E \subseteq \mathcal{O}$: finite set of objects (edges)
- $\eta : E \to V \times V$:
  assignment of an ordered pair of vertices to each edge
- $\lambda : V \cup E \to \mathcal{P}(\mathcal{L})$:
  assignment of a finite set of labels to each object
- $\nu : (V \cup E) \times \mathcal{K} \to \mathcal{N}$:
  partial assignment of values for properties to objects

A lattice of data models

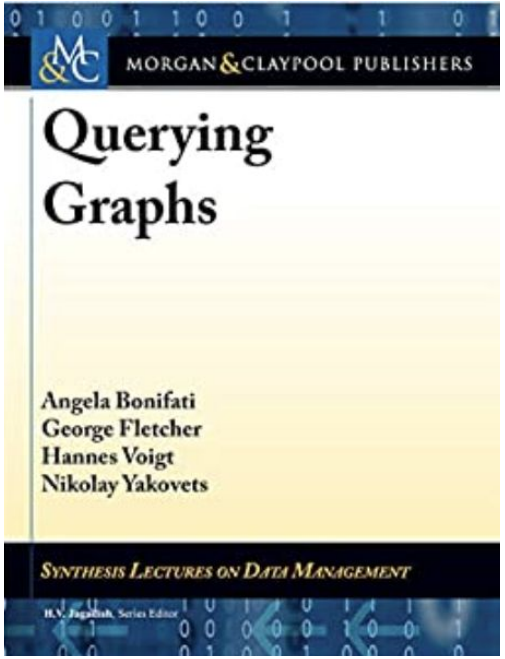
- A data model per use case
- Need of making different data models interoperable via mappings or direct translations
- How expressive and human-friendly is a data model?




W3C RDF Dataset
ISO/IEC GQL Property Graph Model
Hyper vertices
Multiple edge labels
Neo4j / Oracle PGQL Property Graph Model
Multiple vertex labels
Gremlin / MS Cosmos Property Graph Model
Single vertex labels, vertex properties, edge properties
W3C RDF Graph
Pregel / Giraph Graph (used for e.g. Node2vec)
Multiple edges between a pair of nodes, single edge label
Data Graph (used for e.g. Page rank, Connected component)
Weighted Graph (used for e.g. Shortest path, Louvain modularity)
Vertex data
Edge data
Directed Graph
Direction
Simple Graph

Basic ingredients

Graph Queries



Querying Graphs

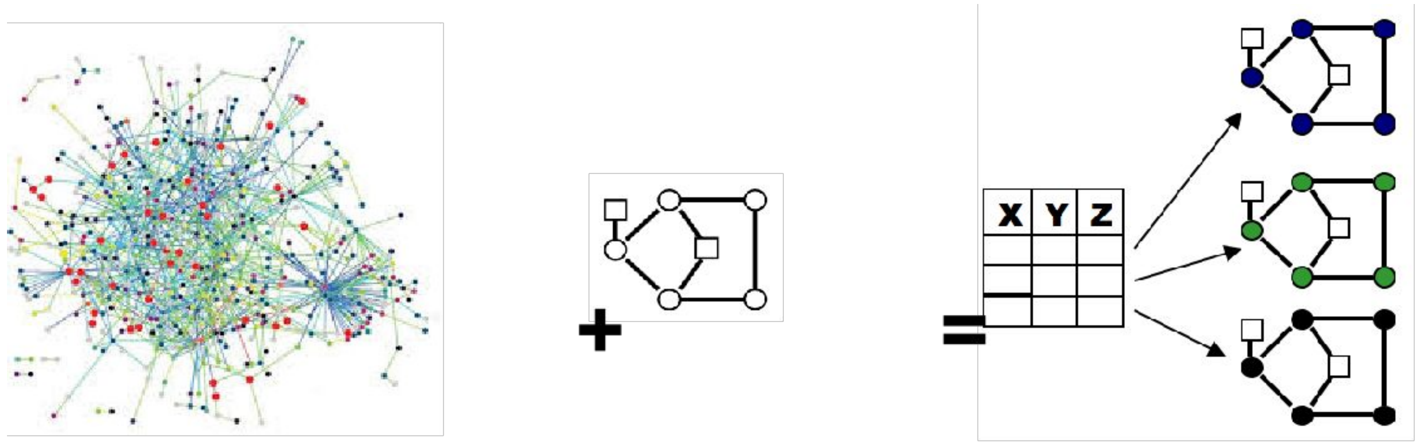
Angela Bonifati
George Fletcher
Hannes Voigt
Nikolay Yakovets

MORGAN & CLAYPOOL PUBLISHERS

SYNTHESIS LECTURES ON DATA MANAGEMENT

H.V. Jagadish, Series Editor

Logic as the underlying formalism of graph query languages



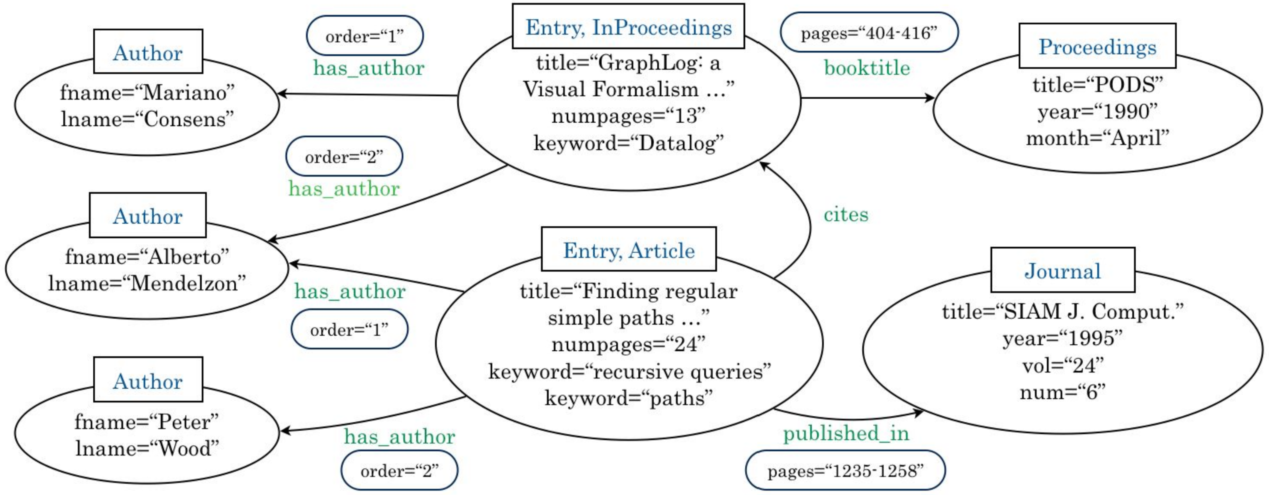
Input graph Graph pattern Binding table

- Semantics of the Graph Query Languages based on non-recursive Datalog with negation
- Several concrete query languages: OpenCypher, Oracle PGQL, …then GQL and SQL/PGQ in the future (ISO/IEC W3G standardization process ongoing with the participation of the LDBC community)

Graph Pattern Matching and Filtering

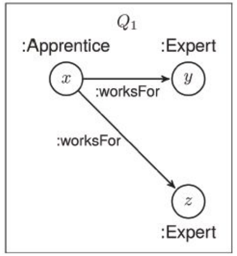
```
SELECT n2.fname AS Author1, n3.fname AS Author2, n1.title AS EntryTitle
FROM "biblio"
MATCH (n1:Entry)-[e1:has_author]->(n2:Author), (n1)-[e2:has_author]->(n3)
WHERE n2 != n3 AND e1.order < e2.order
```

Author1	Author2	EntryTitle
"Mariano"	"Alberto"	"GraphLog: a Visual Formalism ..."
"Alberto"	"Peter"	"Finding regular simple paths ..."

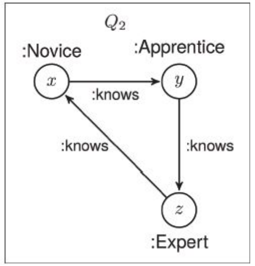


From basic graph patterns to complex graph patterns

- Basic graph patterns including joins (Conjunctive Queries - CQs), e.g.

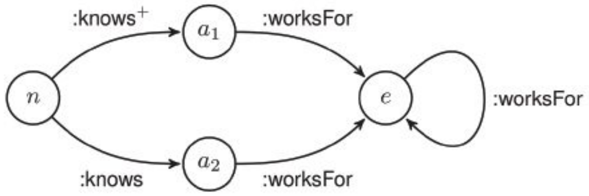


(x,y,z) <— :Apprentice (x), :Expert (y), :Expert(z), :worksFor(x,y), :worksFor(x,z)



(x,y,z) <— :Novice (x), :Apprentice(y), :Expert(z), :knows(x,y), :knows(y,z), :knows(z,x)

- Complex graph patterns with joins and recursion (Conjunctive Regular Path Queries - CRPQs), e.g.



(n,e) <— :knows$^+$(n,a_1), :worksFor(a_1,e), :knows (n,a_2), :worksFor(a_2,e), :worksFor(e,e)

Regular Path Queries (RPQs)

- A regular path query (RPQ) over the set of edge labels Σ is expressed as a regular expression over Σ.
- The answer *Q(D)* to an **RPQ Q** over a database D is the set of pairs of nodes connected in D by a directed path traversing a sequence of edges forming a word in the regular language *L(Q)* defined by Q.
  - If a ∈ *L*, then a ∈ RPQ.
  - If e ∈ RPQ, then e^{-1} ∈ RPQ.
  - If e, f ∈ RPQ, then (e)/(f) ∈ RPQ.
  - If e, f ∈ RPQ, then e + f ∈ RPQ.
  - If e ∈ RPQ, then e+ ∈ RPQ.

Beyond Regular Path Queries (UCRPQs)

- **UC2RPQ**: Unions of Conjunctions of Regular Path Queries
  - A union of conjunctive regular path queries (UCRPQ) is a finite non-empty set R ⊆CRPQ, each element of which is of the same arity m. Each R is a rule of the form:

$$(z_1, \ldots, z_m) \leftarrow \alpha_1(x_1, y_1), \ldots, \alpha_n(x_n, y_n)$$

    where the α are RPQs and x_i, y_j are variables and each z_i is chosen in the set of variables {x_1, y_1, ….,x_n,y_n}
  - Core constructs of the W3C’s SPARQL 1.1 1, , Oracle’s PGQL and Neo4j’s openCypher
  - Well understood theoretical properties (e.g., polynomial data complexity [Bar13])

[Bar13] P. Barceló. Querying graph databases. PODS’13.

Query evaluation semantics

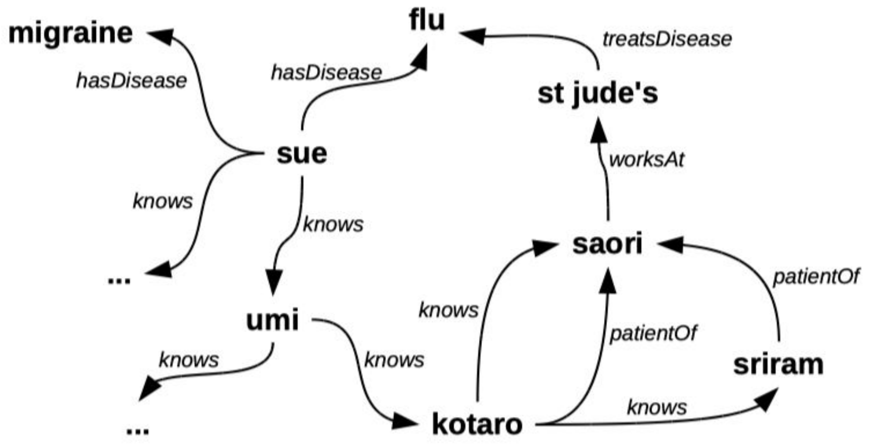
- The semantics Q(G) of evaluating query a Q on graph G is based on embeddings of the rule body’s of Q in G:

$$Q(G) = \bigcup_{head \leftarrow body \in Q} \{h(head) \mid h(body) \subseteq G\}$$

  where h is a homomorphism, i.e., a function with domain N ∪ Variables and range N that is the identity on N.

- Under **homomorphism**, different query variables can be mapped to the same vertex
- Alternatively, under **isomorphism**, the mapping between a query variable and a graph vertex is injective

Homomorphism-based vs. Isomorphism-based semantics

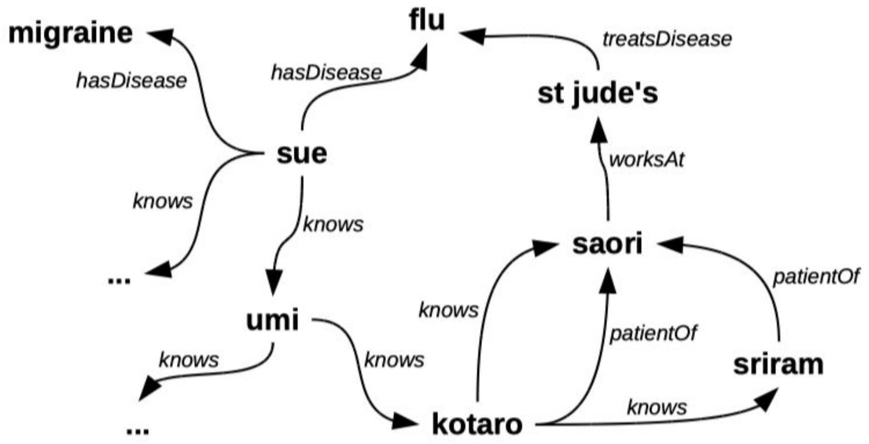


Example: Patients and their friends (**homomorphisms**)

Q = ⟨?p, ?f ⟩ ← (?p, knows, ?f), (?p, patientOf , ?d)

Q(G)= {⟨kotaro, saori⟩, ⟨kotaro, sriram⟩, . . .}

Homomorphism-based vs. Isomorphism-based semantics

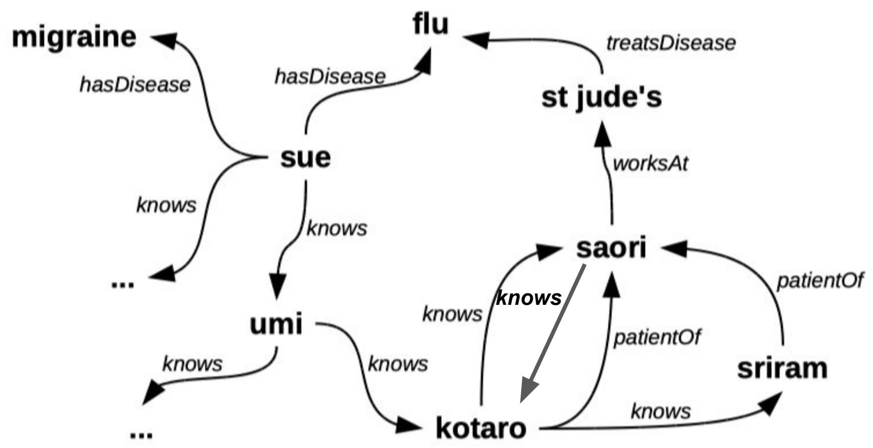


Example: Patients and their friends (**isomorphisms**)

Q = ⟨?p, ?f ⟩ ← (?p, knows, ?f), (?p, patientOf , ?d)

Q(G)= {⟨k~~otaro, saori⟩,~~ ⟨kotaro, sriram⟩, . . .}

Path semantics for Regular Path Queries (RPQs)

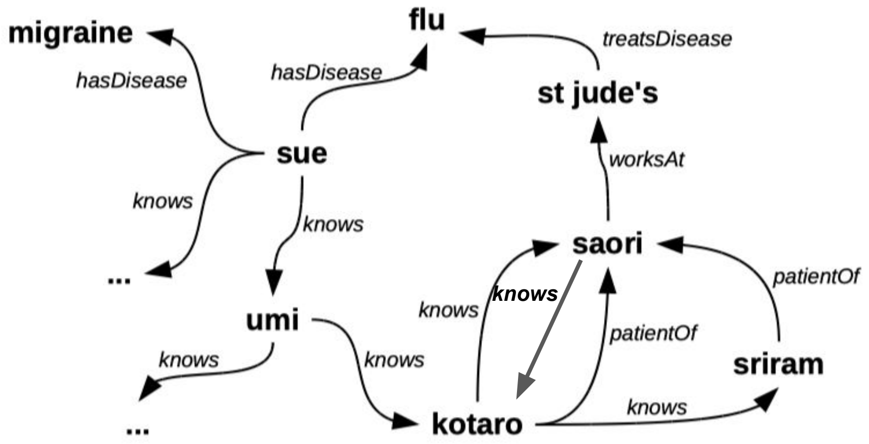


Example: People who know each other (**arbitrary path semantics**)

Q = 〈?p, ?f 〉 ← (?p, knows*, ?f)

Q(G)= {〈kotaro, saori〉, 〈saori, kotaro〉, 〈kotaro, kotaro〉, 〈saori, saori〉, 〈sue, umi〉, 〈umi, kotaro〉, 〈sue, kotaro〉, 〈kotaro, sriram〉, 〈saori, sriram〉, . . .}

Path semantics for Regular Path Queries (RPQs)



Example: People who know each other (**simple path semantics**)

Q = ⟨?p, ?f ⟩ ← (?p, knows*, ?f)

Q(G)= {⟨kotaro, saori⟩, ⟨saori, kotaro⟩, ~~⟨kotaro, kotaro⟩, ⟨saori, saori⟩, ⟨~~sue, umi⟩, ⟨umi, kotaro⟩, ⟨sue, kotaro⟩, ⟨kotaro, sriram⟩, ⟨saori, sriram⟩, . . .}

Algebraic operators for query processing

- An algebra for complex graph queries (UCRPQs)
- Example:

$$e \;::=\; \ell \mid e^* \mid e \cup e \mid \bowtie^{\Phi,c}_{pos_i,pos_j} (e,\ldots,e),$$

$$\begin{aligned}
\text{:knownExpert}(y,y)\ \mathtt{IN}\ a &\leftarrow \text{:knows}(x,y)\ \mathtt{AS}\ k, \text{:Expert}(z), k.year > 2000, y = z.\\
\text{:related}(x,y)\ \mathtt{IN}\ b &\leftarrow \text{:knows}(x,y).\\
\text{:related}(x,y)\ \mathtt{IN}\ b &\leftarrow \text{:worksFor}(x,y).\\
\text{result}(x) &\leftarrow \text{:knownExpert}(x,x), \text{:related}^*(x,y),\\
&\quad\; x.salary < 5000, y.salary < x.salary.
\end{aligned}$$

$$\bowtie^{\emptyset}_{\text{src}_1} (\bowtie^{\Phi,x}_{\text{trg}_1,\text{trg}_1} (\text{:knows}, (\text{:knows} \cup \text{:worksFor})^*)),$$

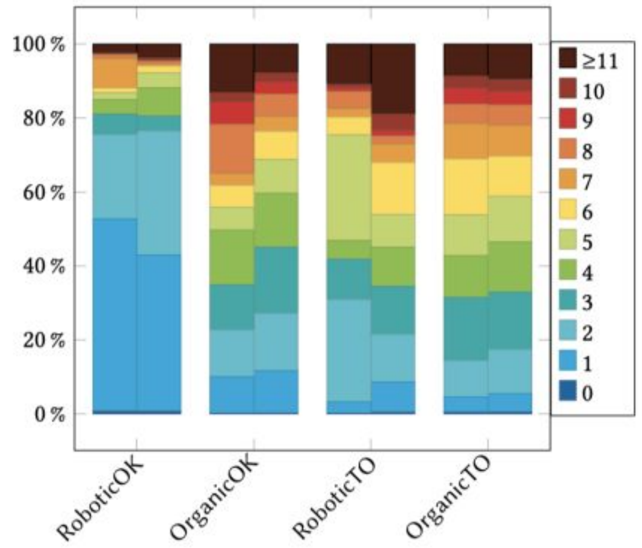
$$\begin{aligned}
&\text{edge}_1.year > 2000 \wedge \lambda(\text{trg}_1) = \text{:Expert} \wedge \text{trg}_1.salary < 5000\\
&\qquad\qquad \wedge\, \text{trg}_1.salary > \text{trg}_2.salary \wedge \text{trg}_1 = \text{src}_2
\end{aligned}$$

Logic and Algebra for graph processing and learning

- Logical reasoning can help drive the interplay between graph databases, statistical learning, and symbolic learning:
    - Understand how to combine ontological reasoning, property graph querying and graph embeddings
    - Help predict recurrent structures into graphlets to be used in query processing and algebraic operators by using graph neural networks
    - Encode probabilistic models and causal inference relationships in property graphs to form the basis of graph neural networks

[Zu21] Z. Wu et al.: A Comprehensive Survey on Graph Neural Networks. IEEE Trans. Neural Networks Learn. Syst. 32(1): 4-24 (2021)
[Ma21] G. Ma et al.: Deep graph similarity learning: a survey. Data Min. Knowl. Discov. 35(3): 688-725 (2021)

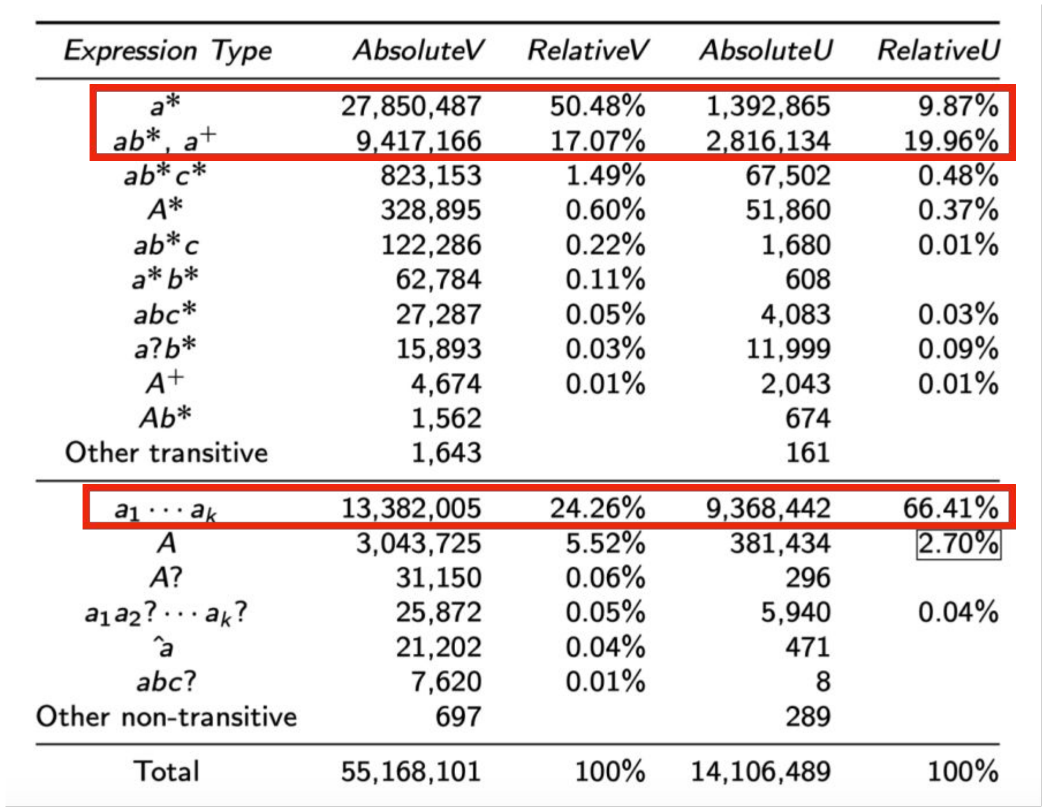
How do real-world queries look like: the Wikidata lesson



- Higher percentages of large TO queries and organic (OK+TO) queries
- Avg nr. of triples for OK (TimeOut) queries is 2.58 (5.65)
- Highest nr. of triples is 67 (found in 68 queries in the logs)
- The largest property path length is 19

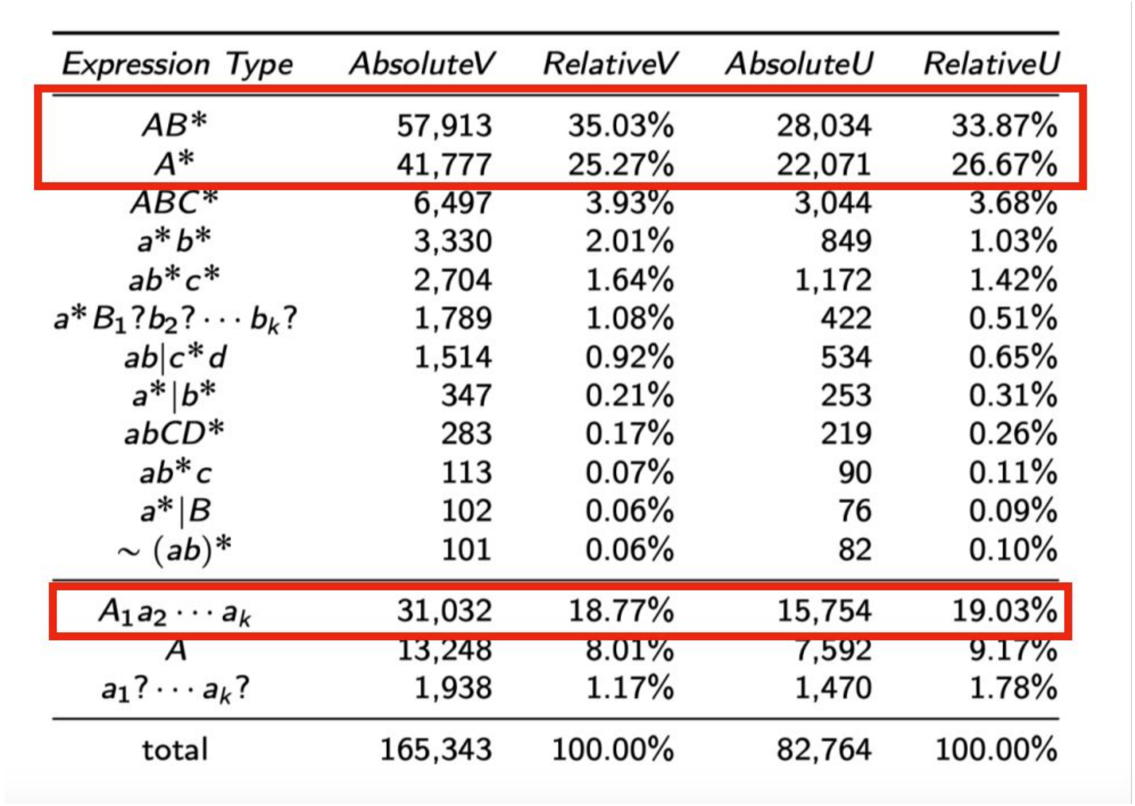
[Bo19] A. Bonifati, W. Martens, T. Timm: Navigating the Maze of Wikidata Query Logs. In Proceedings of the Web Conference, 2019

Characteristics of Robotic Property Paths



Expression Type	AbsoluteV	RelativeV	AbsoluteU	RelativeU
a^*	27,850,487	50.48%	1,392,865	9.87%
ab^*, a^+	9,417,166	17.07%	2,816,134	19.96%
ab^*c^*	823,153	1.49%	67,502	0.48%
A^*	328,895	0.60%	51,860	0.37%
ab^*c	122,286	0.22%	1,680	0.01%
a^*b^*	62,784	0.11%	608	
abc^*	27,287	0.05%	4,083	0.03%
$a?b^*$	15,893	0.03%	11,999	0.09%
A^+	4,674	0.01%	2,043	0.01%
Ab^*	1,562		674	
Other transitive	1,643		161	
$a_1 \cdots a_k$	13,382,005	24.26%	9,368,442	66.41%
A	3,043,725	5.52%	381,434	$\boxed{2.70\%}$
$A?$	31,150	0.06%	296	
$a_1 a_2? \cdots a_k?$	25,872	0.05%	5,940	0.04%
$\hat{}a$	21,202	0.04%	471	
$abc?$	7,620	0.01%	8	
Other non-transitive	697		289	
Total	55,168,101	100%	14,106,489	100%

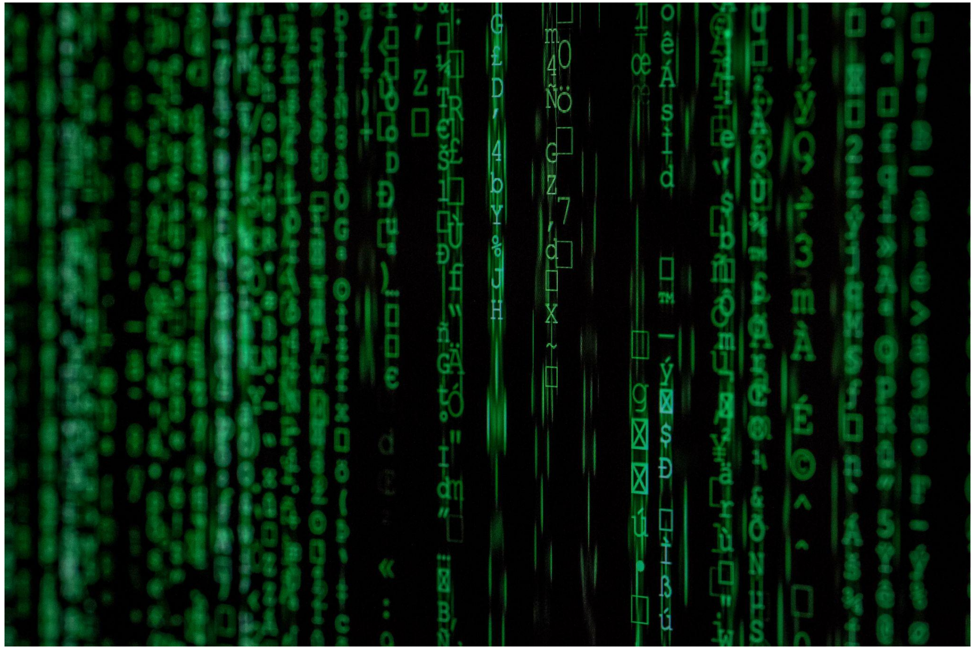
Characteristics of Organic Property Paths



Expression Type	AbsoluteV	RelativeV	AbsoluteU	RelativeU
$AB*$	57,913	35.03%	28,034	33.87%
$A*$	41,777	25.27%	22,071	26.67%
$ABC*$	6,497	3.93%	3,044	3.68%
$a*b*$	3,330	2.01%	849	1.03%
$ab*c*$	2,704	1.64%	1,172	1.42%
$a*B_1?b_2?\cdots b_k?$	1,789	1.08%	422	0.51%
$ab\|c*d$	1,514	0.92%	534	0.65%
$a*\|b*$	347	0.21%	253	0.31%
$abCD*$	283	0.17%	219	0.26%
$ab*c$	113	0.07%	90	0.11%
$a*\|B$	102	0.06%	76	0.09%
$\sim (ab)*$	101	0.06%	82	0.10%
$A_1a_2\cdots a_k$	31,032	18.77%	15,754	19.03%
A	13,248	8.01%	7,592	9.17%
$a_1?\cdots a_k?$	1,938	1.17%	1,470	1.78%
total	165,343	100.00%	82,764	100.00%

Basic ingredients

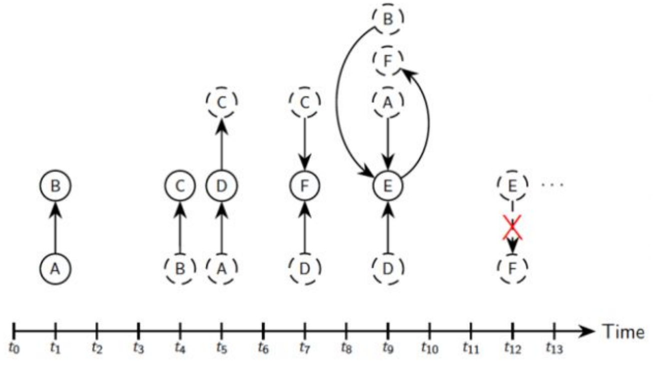
Streaming property graphs



Dynamic and streaming aspects

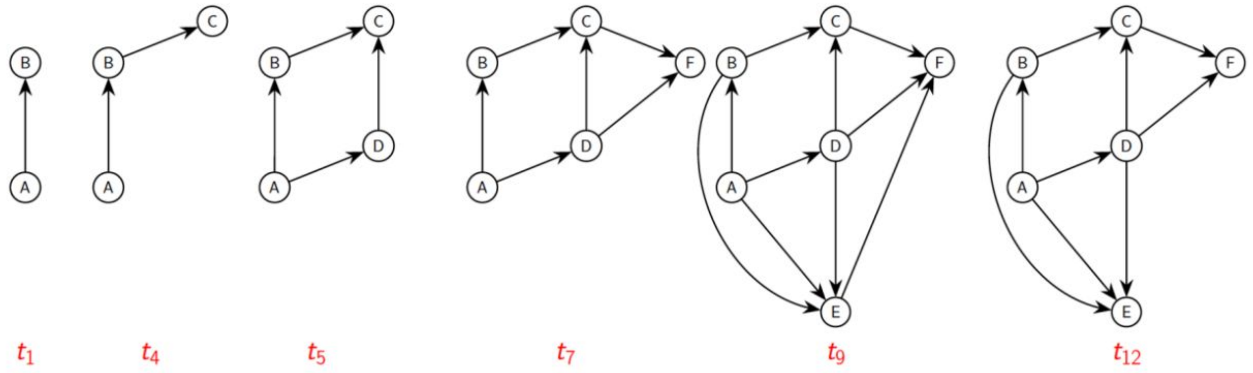
- Dynamic graphs are graphs that can accommodate updates (insertions, deletions, changes) and allow querying on the new/old state
- Streaming graphs are graphs that are unbound as new data arrives at high-speed.
- Current systems and libraries (Gelly/Apache Flink) focus on aggregates/projections
- However, more complex query processing operators taking into account recursion, path-oriented semantics etc. need to be investigated
- Graph processing systems are also inherently dynamic and need to respond to all these challenges

Streaming graphs



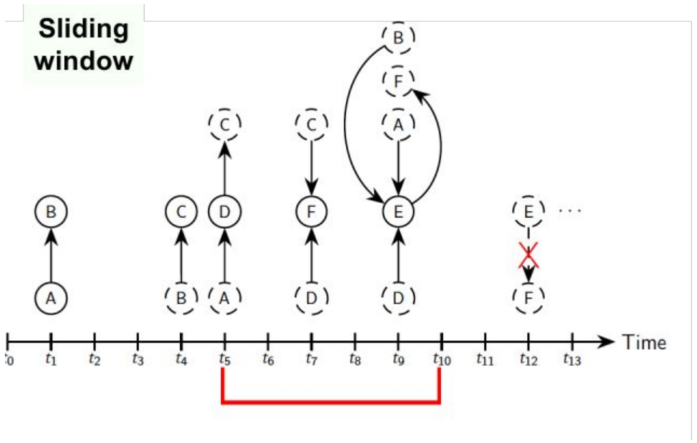
B
F
A
C
C
E
B
C
D
F
E
E
A
B
A
D
D
F
Time
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13}

- Combines two difficult problems: streaming+graphs
- Unbounded ⇒ don't see entire graph
- Streaming rates can be very high



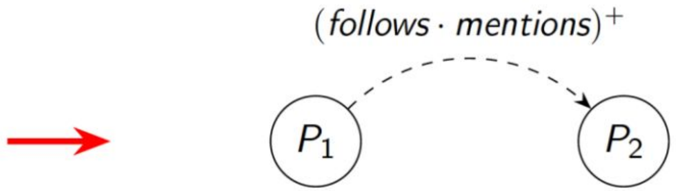
Streaming graph models

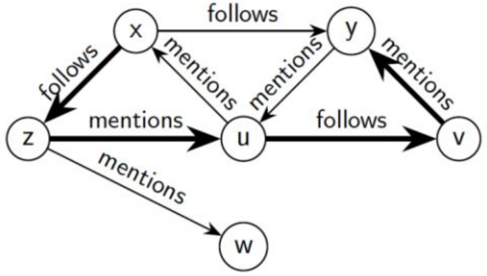
- Window-based semantics (use window to batch edges)
- Continuous semantics (edges are batched as they come)
- Complex vs. Simple operations



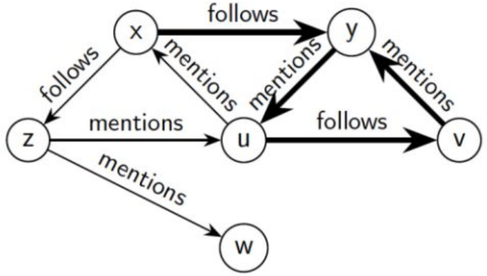
Streaming RPQs

$$Q_1 = (follows \cdot mentions)^+$$





Simple paths

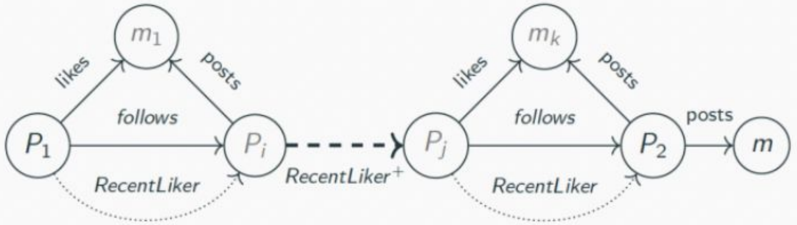


Arbitrary paths

[Pa20] A. Pacaci, A. Bonifati and T. Ozsu.: Regular Path Query Evaluation on Streaming Graphs. SIGMOD Conference 2020: 1415-1430

Towards a streaming graph query processor

Based on LDBC SNB Interactive Query 7:



G-CORE representation

```
PATH RL = (x)-[:follows]->(y),
          (x)-[:likes]->(m1)<-[:posts]-(y)
CONSTRUCT (p1) -[:notify]-> (m)
MATCH (p1)-/ <~RL+> /->(p2),
      (p2)-[:posts]->(m)
ON ldbc_stream WINDOW(24 hours)
```

Datalog program:

$$RL(u_1, u_2) \leftarrow l(u_1, m_1), f(u_1, u2), p(u_2, m_1)$$
$$Answer(u, m) \leftarrow RL^+(u, u_2), p(u_2, m)$$

Streaming graph algebra

G-CORE Query:

```
PATH RL = (x)-[:follows]->(y),
          (x)-[:likes]->(m1)<-[:posts]-(y)
CONSTRUCT (p1) -[:notify]-> (m)
MATCH (p1)-/ <~RL+> /->(p2),
      (p2)-[:posts]->(m)
ON ldbc_stream WINDOW(24 hours)
```

SGA Expression

$$S_l = \sigma_{l=likes}(\mathcal{W}^{24}(S))$$

$$S_f = \sigma_{l=follows}(\mathcal{W}^{24}(S))$$

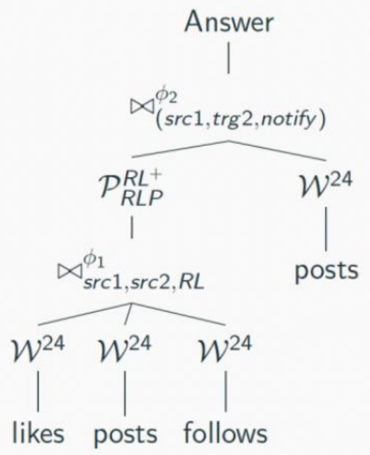
$$S_p = \sigma_{l=posts}(\mathcal{W}^{24}(S))$$

$$S_{RecentLiker} = \bowtie_{\phi}^{src1,src3,RecentLiker}(S_{likes}, S_{follows}, S_{posts})$$

$$S_{Related} = \mathcal{P}_{RecentLiker^+}^{Notify}(S_{RecentLiker})$$

$$Answer = \bowtie_{\phi}^{src1,trg2,Notify}(S_{Related}, S_p)$$

Logical query plan



Lack of interoperability and potential solutions

- The lack of interoperability for graph processing systems hinders fair comparison and benchmarking.
- Different communities involved (data management, data mining, large-scale systems and ML)
- How to alleviate the problem?
- E.g. Indexing and sampling might help to improve and predict the performances of graph queries

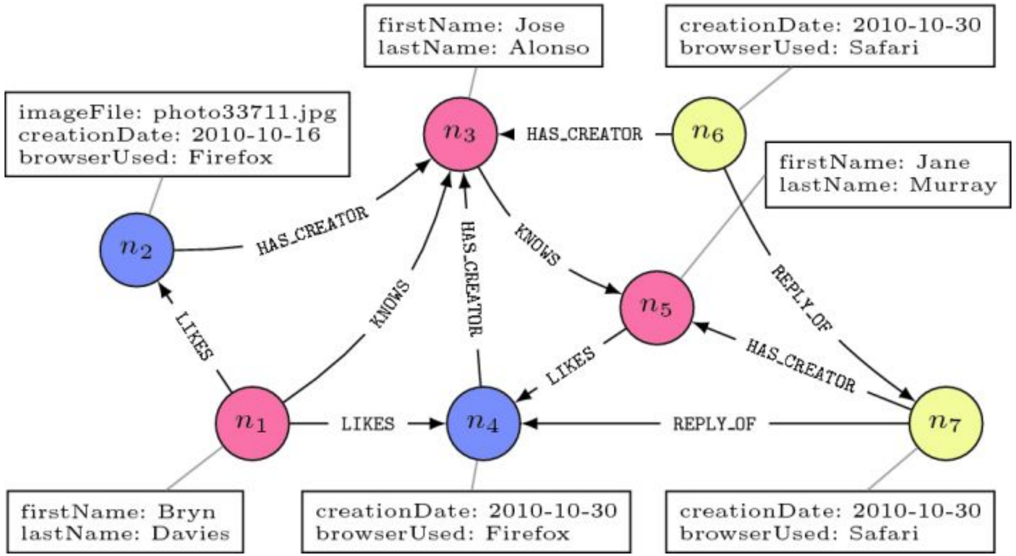
Indexing for graph queries

- Much of the attention has been devoted to LCR query (s, t, LC): Checking whether there is a path from s to t using only edges with labels in LC
  - LC (label constraint) (l1 ∪ ... ∪ lk)+, where ∪ is disjunction
- GTC (Generalized Transitive Closure): for every pair of vertices (u,v), recording whether u reaches v, and
  - the minimal label set from u to v, i.e., the set of labels, where the element cannot be removed anymore, e.g., two paths from u to v with label sets {a,b} and {a}, the minimal one is {a}
- All existing works about LCR queries are trying to compress GTC effectively with efficient query processing
- Little attention has been devoted to other types of label constraints (e.g. with concatenation instead of disjunction) - more during Chao's part!

[Ha16] M. S. Hassan et al. Graph indexing for shortest-path, finding over dynamic sub-graphs. In SIGMOD 2016.
[Va17] L. Valstar et al. Landmark indexing for evaluation of label-constrained reachability queries. In SIGMOD 2017.

Basic ingredients

Schemas and constraints for Property graphs



The quest for schemas in property graphs

- Graph Databases are schema-less:
  - NoSQL for efficiently storing & processing graph-shaped data.
  - No a priori schema constraints → error-prone data integration
- In our previous work, we focused on schema validation and evolution for property graphs
  - Underlying property graph model:
    - (labeled multigraph with key/value lists attached to nodes & edges)
    - → rich formalism amenable to schema discovery

[Bo19] Angela Bonifati, Peter Furniss, Alastair Green, Russ Harmer, Eugenia Oshurko, Hannes Voigt: Schema Validation and Evolution for Graph Databases. ER 2019: 448-456

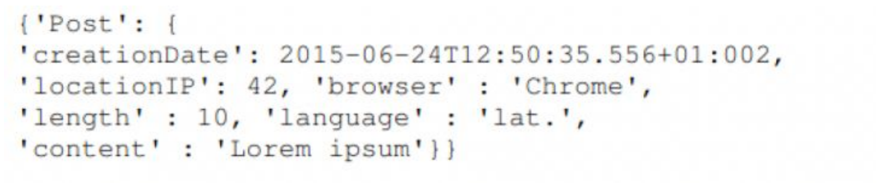
Towards schema inference for PGs

- Ongoing PG Schema standardisation process (ISO SC32/ WG3).
- Existing schema inference mechanisms are basic:
  - no hierarchies
  - no complex types
- Schema inference using MapReduce (MRSchema):
  - considers either node labels or node properties → trade-off
  - property co-occurrence information loss (label-oriented approach) vs. extraneous type inference (property-oriented approach).

[Ba19] Mohamed Amine Baazizi, Dario Colazzo, Giorgio Ghelli, Carlo Sartiani:Parametric schema inference for massive JSON datasets. VLDB J. 28(4): 497-521 (2019)
[Lb21] Hana Lbath, Angela Bonifati, and Russ Harmer. “Schema Inference for Property Graphs”. In: EDBT. 2021, pp. 499–504.

GMM Schema

- Base Types (*BT*): set of element types (L,K,O, Eb), where: $L \in \mathit{L}$: set of labels, $K \in \mathit{K}$: set of property names, $O \subseteq \mathit{K}$: subset of optional property names, $Eb \subset \mathit{BT}$: set of element types b extends.
- Leverages hierarchical clustering, using Gaussian Mixture Model
- Example: LDBC Post node instance

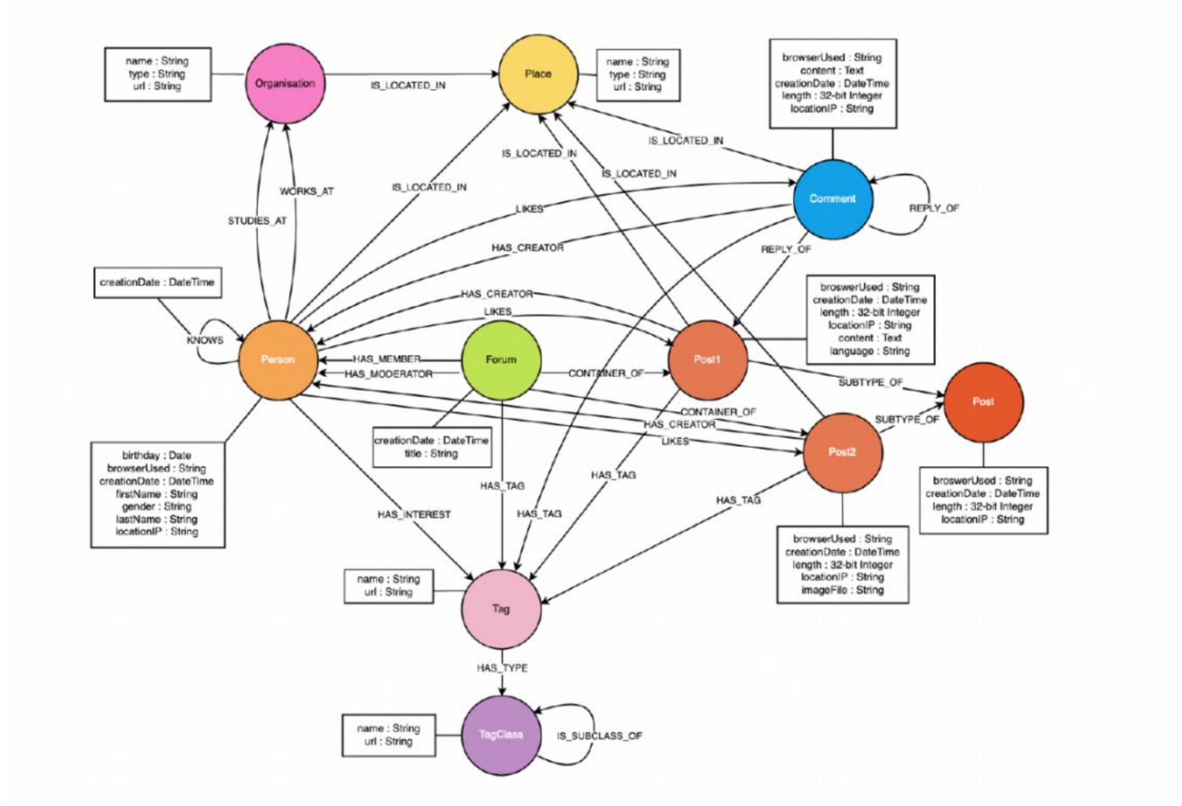


```
{'Post': {
'creationDate': 2015-06-24T12:50:35.556+01:002,
'locationIP': 42, 'browser' : 'Chrome',
'length' : 10, 'language' : 'lat.',
'content' : 'Lorem ipsum'}}
```

  - Base type: ({Post},K, {language, content}, $\varnothing$), where K = {creationDate, locationIP, length, content}.

[Bo22a] Angela Bonifati, Stefania Dumbrava, Nicolas Mir: Hierarchical Clustering for Property Graph Schema Discovery. EDBT 2022: 2:449-2:453
[Bo22b] A. Bonifati et al. “DiscoPG: Property Schema Discovery and Exploration” VLDB 2022 (demo track - to appear)

LDBC inferred schema

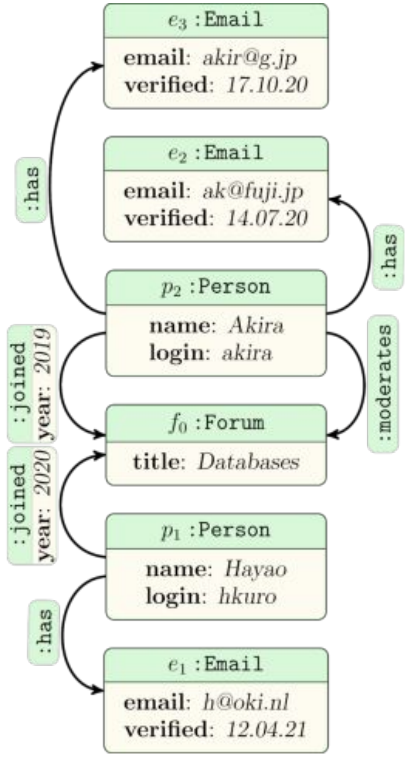


Limited Support for Keys in Graph Databases

- Landscape is **diverse**:
  - Some systems offer property-based primary keys for nodes
  - Some systems support uniqueness
  - Some systems support mandatoriness
- Yet we need to support all of these, and more, to satisfy current practical needs.
- There is already a **significant drift** between database vendors
  - We need to get on the same page
  - We need to bring the best of academic work to the needs of industry

PG-Keys: keys for property graphs

- Declaratively specify the scope of the key and its values in your favourite PG query language (a parameter of PG-Keys). Here we use Cypher-like syntax.
- For instance
    - FOR p WITHIN (p:Person) IDENTIFIER p.login; says that “each person is identified by their login”, and
    - FOR f WITHIN (f:Forum)<-[:joined]-(:Person) IDENTIFIER f.name, p WITHIN (f)<-[:moderates]-(p:Person); says that “each forum with a member is identified by its name and moderator”.



[An21] Renzo Angles, Angela Bonifati, Stefania Dumbrava, George Fletcher, Keith W. Hare, Jan Hidders, Victor E. Lee, Bei Li, Leonid Libkin, Wim Martens, Filip Murlak, Josh Perryman, Ognjen Savkovic, Michael Schmidt, Juan F. Sequeda, Slawek Staworko, Dominik Tomaszuk:PG-Keys: Keys for Property Graphs. SIGMOD Conference 2021: 2423-2436

Ongoing standardization

- Graph standard data model, data manipulation language and data definition language, as well as the schema language for property graphs constitute one of the most important challenges in the upcoming years
- ISO/IEC GQL Project together with the LDBC community effort gathering researchers, industry representatives and graph database vendors
  - Goal: standardize languages for making them adoptable by existing implementations
  - https://www.gqlstandards.org/
  - https://ldbcouncil.org/gql-community/overview/

LDBC member companies and institutions

ANT GROUP

ORACLE LABS

ArangoDB

intel

aws

CHOROGRAPH

MEM GRAPH

KATANA GRAPH

JCC Consulting, Inc.

TigerGraph

*Sparsity

1794

ENS ÉCOLE NORMALE SUPÉRIEURE

ontotext

neo4j

Birkbeck UNIVERSITY OF LONDON

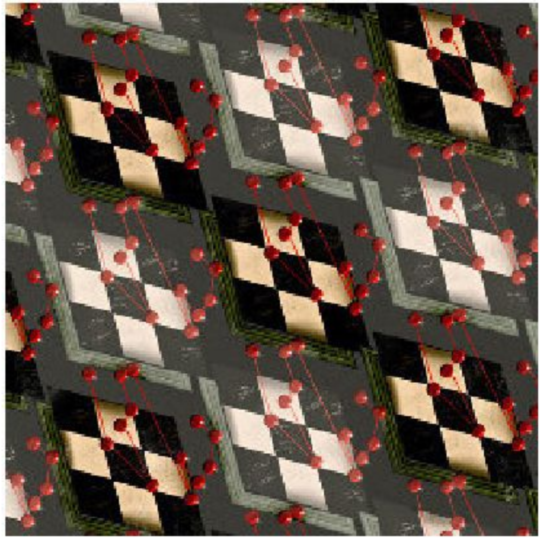
FORTH

THE UNIVERSITY of EDINBURGH

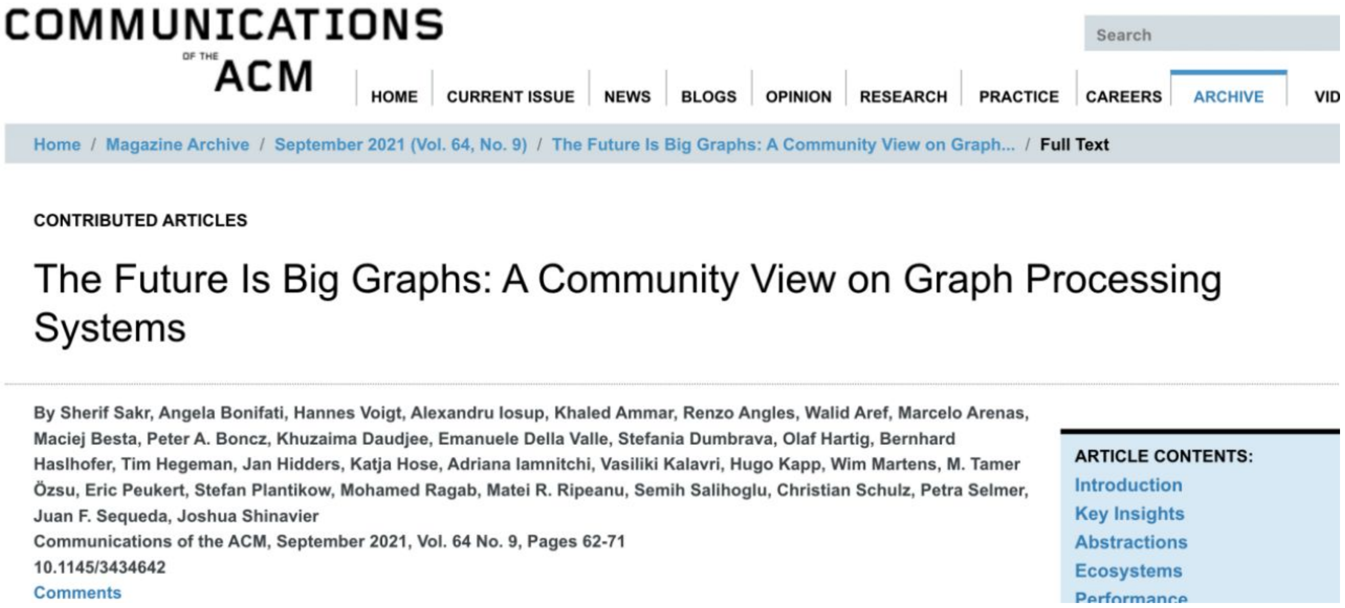
and many Associate Members ...

Towards concluding

Our vision on Graph Processing Systems



Our vision [CACM 21]



COMMUNICATIONS OF THE ACM

Search

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE | CAREERS | ARCHIVE | VID

Home / Magazine Archive / September 2021 (Vol. 64, No. 9) / The Future Is Big Graphs: A Community View on Graph... / Full Text

CONTRIBUTED ARTICLES

The Future Is Big Graphs: A Community View on Graph Processing Systems

By Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei R. Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier

Communications of the ACM, September 2021, Vol. 64 No. 9, Pages 62-71

10.1145/3434642

Comments

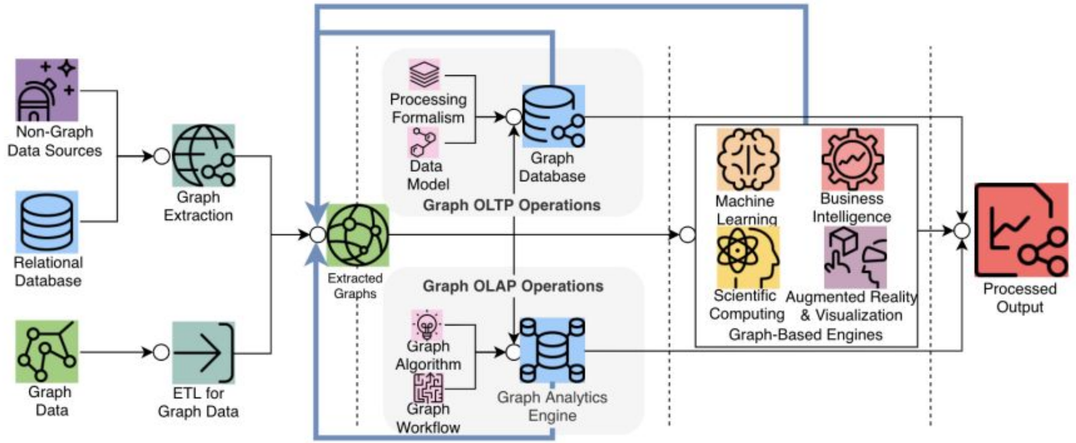
ARTICLE CONTENTS:

Introduction
Key Insights
Abstractions
Ecosystems
Performance

https://cacm.acm.org/magazines/2021/9/255040-the-future-is-big-graphs/

Graph processing ecosystems

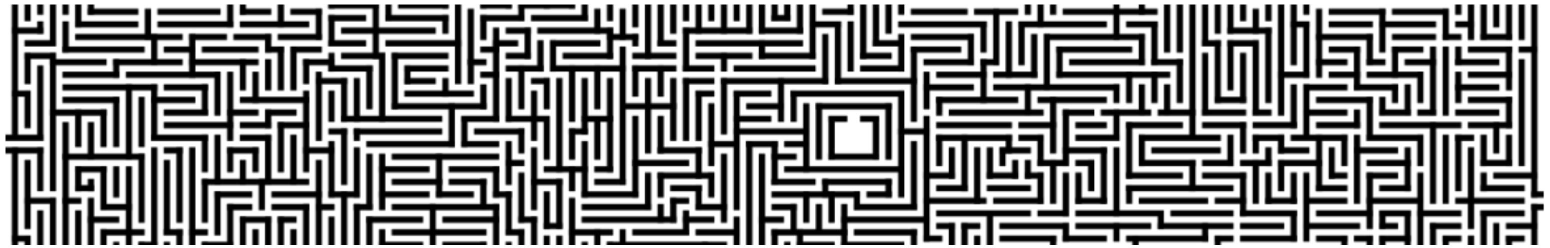
- Complex workflows combining OLTP and OLAP processing are needed in order to handle heterogeneous data and heterogeneous queries and algorithms in full-fledged graph ecosystems



[Sa21] S.Sakr et al. The Future is Big Graphs! a community view on graph processing systems. Commun. ACM 64(9): 62-71 (2021)

Graph analytics at scale

Multi-hop analysis faces combinatorial scaling problem: Every step deeper into the graph multiplies the number of choices and cases to consider



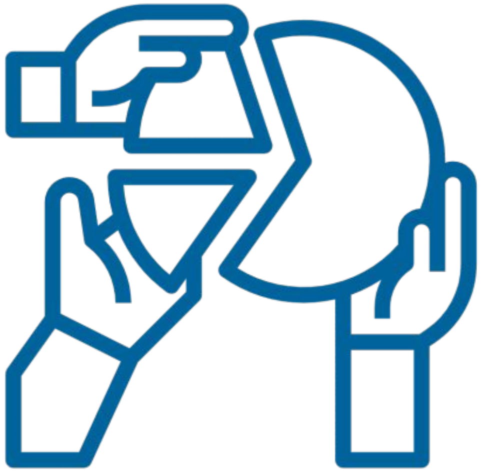
Dealing with this technical challenge is not the typical business interest of a user.

Which challenges are ahead of us to ready graph processing systems for the future?

Challenges to overcome: **Abstractions, Ecosystems, Performance**

Challenges require expertise of many different fields

- Computer systems
- Data management systems
- Data management theory
- Data analytics
- Visualization
- Human computer interaction
- ML/Artificial Intelligence
- …


[collaborate by ArmOkay from the Noun Project]

Graph Processing Ecosystems: Underlying Principles and Knobs

- Complex workloads combining pre-processing, OLAP/OLTP, application-driven components
- Standard data models and query languages
- Reference Architecture
- Scale-up vs. Scale-out
- Dynamic and Streaming endeavours

Big Graph Processing Systems - Open research directions

- How to make the graph data models and query languages interoperable?
- How to define meaningful metrics to execute a graph algorithm, program, query or workflow?
- How to generate heterogeneous workloads (analytical/transactional, batch/streaming, temporal/spatial etc)?
- How to benchmark entire graph pipelines and ecosystems (including ML and simulation)?

Thank you and Q&A

“If you put your mind to it you can accomplish anything.”

-Dr. Emmett Brown