北京大學
PEKING UNIVERSITY

Unified Graph Query Plan Representation and Optimization

Lei Zou

Peking University

Data Management Lab, Wangxuan Institute of Computer Technology

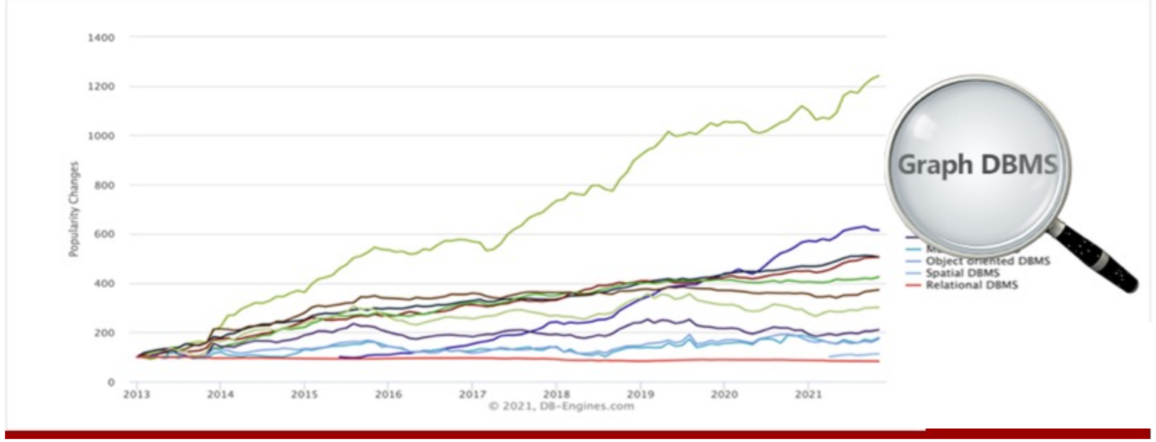
To Collaborators

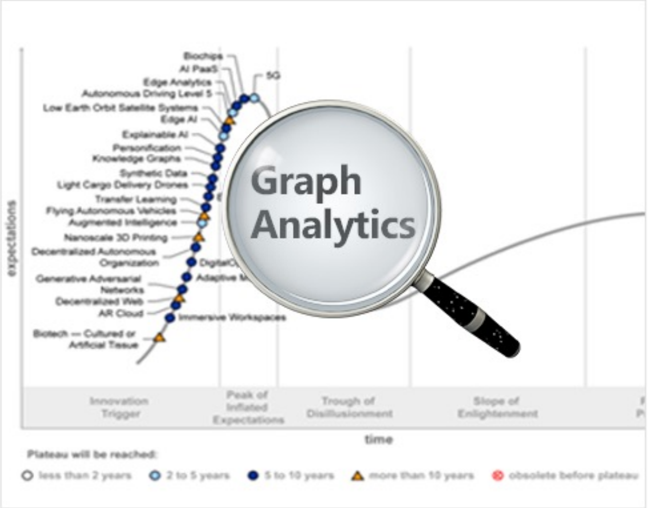
- **Prof. Tamer Ozsu, University of Waterloo**
- **Prof. Jeffrey Xu Yu, The Chinese University of Hong Kong**
- **Yue Pang, Peking University**
- **Linglin Yang, Peking University**
- **Dr. Xiangyang Gou, the University of New South Wales**

Outline

- Graph Databases Overview
- Unified Graph Query Plan Representation
- Graph Query Optimization: Cardinality Estimation
- Conclusions

Graph Databases Overview





Graph databases: a type of NoSQL database

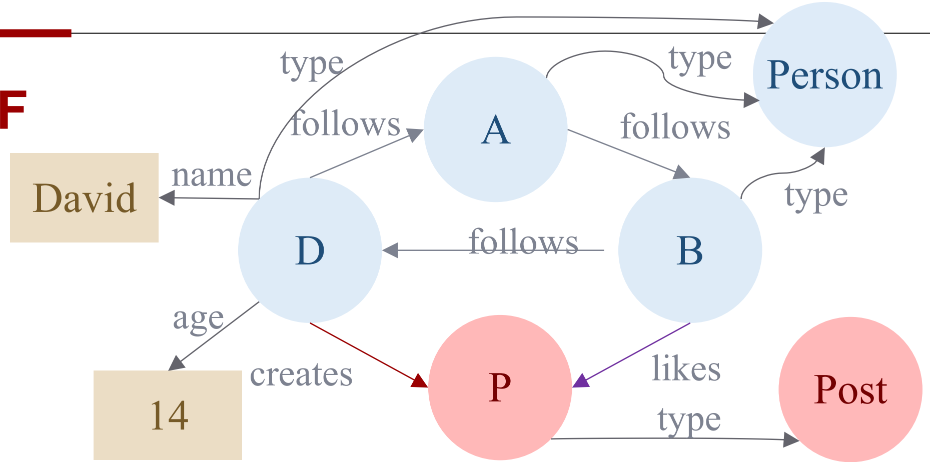
Focuses on storing and querying the relationships between entities using the graph abstraction

Distinction from relational databases

- **(Semi-)Schema-less:** do not have data schemas a priori, flexible
- **Adjacency storage:** materializes the relationships between entities, accessible without joins
- **Deep queries:** subgraph queries with complex join patterns, path queries with unlimited recursion

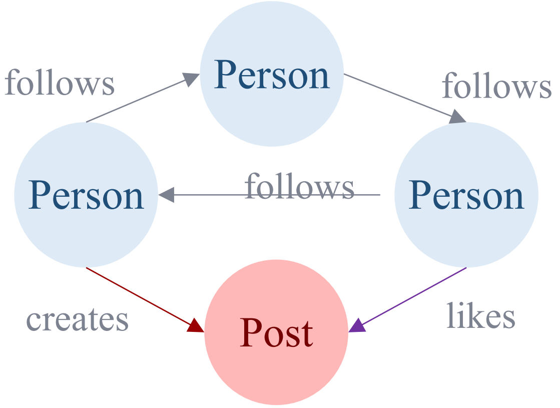
Graph Data & Query Model

RDF



Property Graph

Name	David
Age	45



SPARQL

```
SELECT ?person WHERE {
   ?david <type> <Person> .
   ?david <name> “David” .
   ?david <knows>+ ?person .
}
```

GQL

```
MATCH (:Person WHERE
name=‘David’)-[:knows*1..]-
>(person)
RETURN person
```

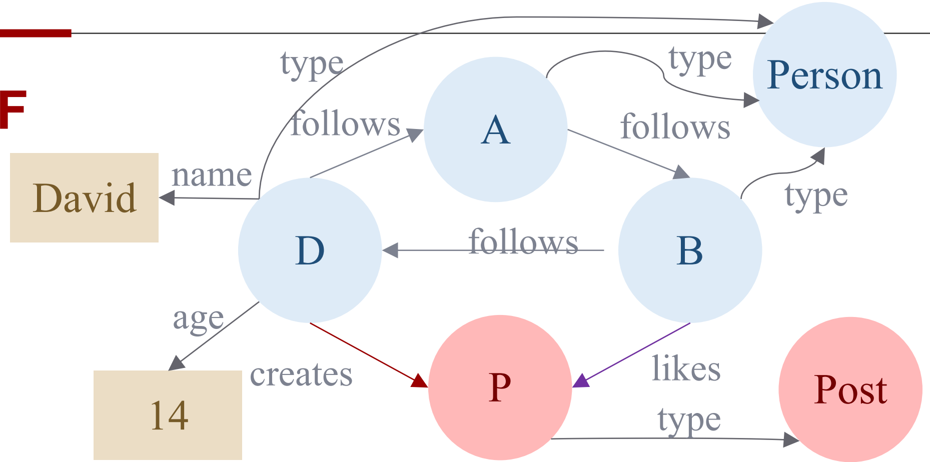
Differences

- **Property graphs** represents properties by key-value tables associated with vertices & edges
- **RDF** represents properties by creating property vertices and linking the entity vertices & edges with them
- Affects the storage layout, query optimization and evaluation

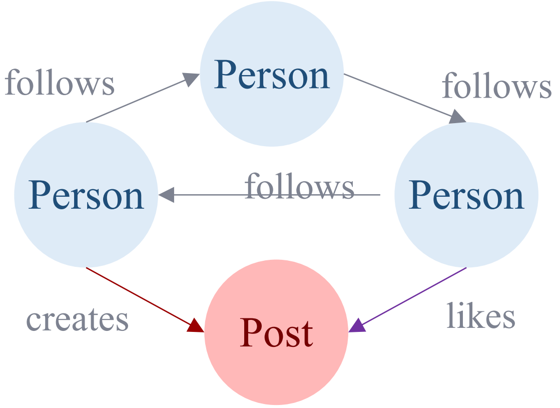
Graph Data & Query Model

北京大学 PEKING UNIVERSITY

RDF



Name	David
Age	45



Property Graph

SPARQL

```
SELECT ?person WHERE {
   ?david <type> <Person> .
   ?david <name> "David" .
   ?david <knows>+ ?person .
}
```

```
MATCH (:Person WHERE
name='David')-[:knows*1..]-
>(person)
RETURN person
```

GQL

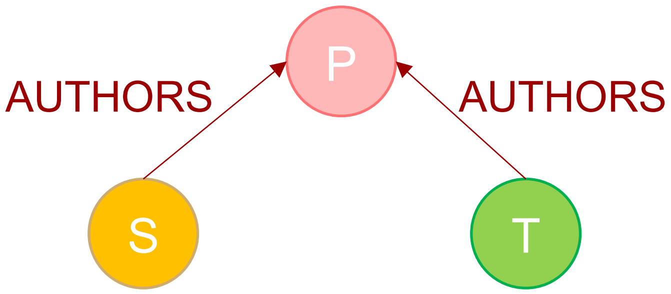
Similarities

- The core query constructs are the same:
  - **Conjunctive graph query, i.e., subgraph matching**
  - **Regular path query**

Graph Data & Query Model

Conjunctive graph query

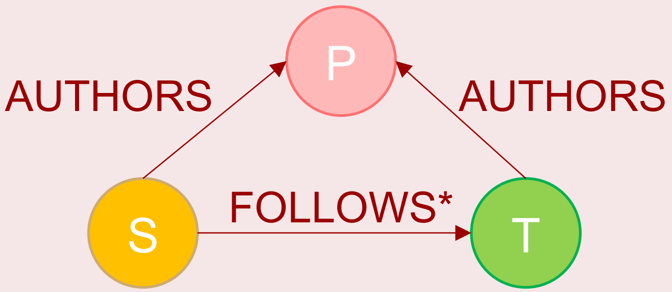
Given a **query graph**, find **subgraphs in the data graph** that are **isomorphic** to it



Regular path query

Given a **regular expression** on the **set of edge labels**, find vertices pairs in the data graph connected by **paths** with edge label sequences that can be **recognized by the regular expression**

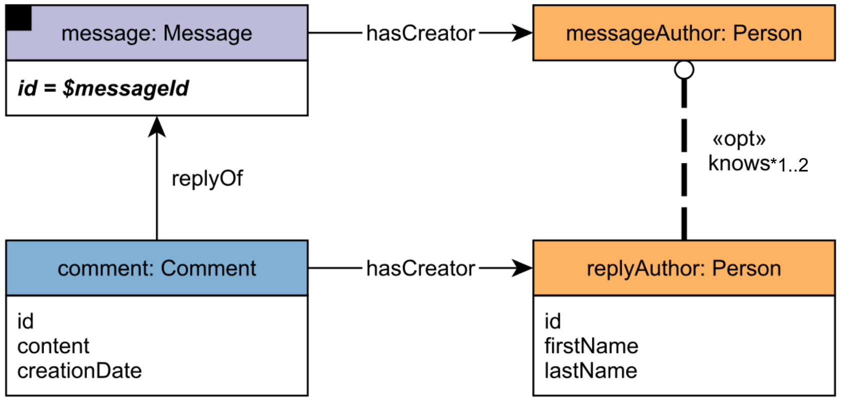
Conjunctive regular path query (CRPQ)



北京大学 PEKING UNIVERSITY

Beyond conjunctive graph & regular path queries

- Besides conjunctive regular path query (CRPQ), existing graph query languages such as SPARQL, Cypher and GQL also include other graph query constructs, OPTIONAL, AGGREGATION, UNION and top-k.



SPARQL

```
SELECT DISTINCT  ?replyAuthorId  WHERE {
    ?message :id   “1”^^xsd:long .
    ?message hasCreator ?messageCreator.
    ?repliedComment :replyOf ?message .
    ?repliedComment :hasCreator ?replyAuthor.
    ?replyAuthor :id ?replyAuthorId .
    OPTIONAL
{ ?replyAuthor :directKnows/ :directKnows? ?messageCreator. }
}
```

Cypher

```
MATCH (message {id: 1})
MATCH (replyAuthor)<-[:hasCreator]-(repliedComment)-[:replyOf]->(message)-[:hasCreator]->(messageCreator)
OPTIONAL MATCH (replyAuthor)-[:directKnows*1..2]->(messageCreator)
RETURN DISTINCT replyAuthor.id AS replyAuthorId
```

2. Graph Query Evaluation: Graph Algebra

A streaming graph algebra incorporating regular path queries and subgraph queries[14]

Subgraph Matching Operator

$$
\begin{aligned}
\bowtie_{\Phi}^{src,trg,d}(S_{l_1},\cdots,S_{l_n}) = \big[&(u,v,d,[ts,exp),\mathcal{D}:e(u,v,l)) \mid \\
&\exists t_i = (src_i,trg_i,l_i,[ts_i,exp_i),\mathcal{D}_i) \in S_{l_i}, 1 \le i \le n \\
&\wedge \Phi((src_1,trg_1,\cdots,src_n,trg_n)) \wedge \\
&u = src \wedge v = trg \wedge \bigcap_{1 \le i \le n} [ts_i,exp_i) \neq \emptyset \wedge \\
&ts = \max_{1 \le i \le n}(ts_i) \wedge exp = \min_{1 \le i \le n}(exp_i)\big].
\end{aligned}
$$

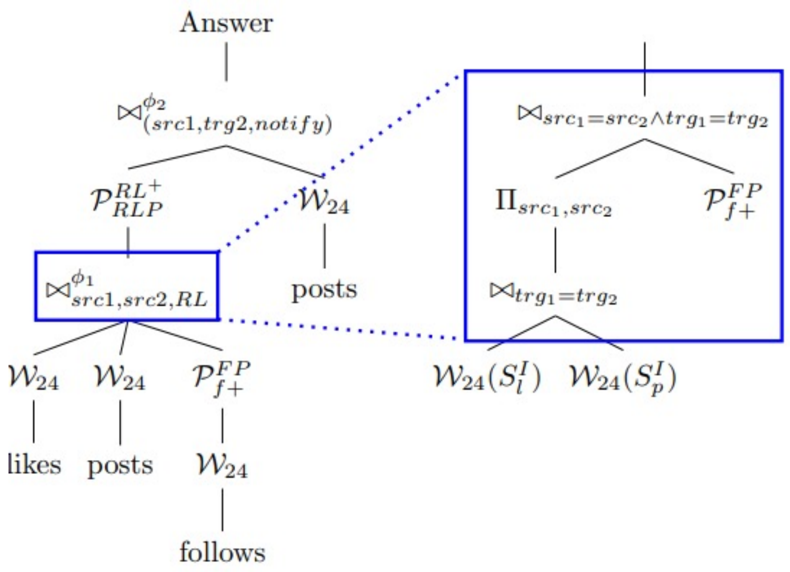
Path Navigation Operator

$$
\begin{aligned}
\mathcal{P}_R^d(S_{l_1},\cdots,S_{l_n}) = \big[&(u,v,d,[ts,exp),\mathcal{D}) \mid \exists p : u \xrightarrow{p} v \wedge \\
&\forall e_i \in p, \exists t_i = (src_i,trg_i,l_i,[ts_i,exp_i),\mathcal{D}_i) \in S_{l_i} \wedge \\
&\phi^P(p) \in L(R) \wedge \bigcap_{t \in p} [t.ts,t.exp) \neq \emptyset \wedge \\
&ts = \max_{t \in p}(t.ts) \wedge exp = \min_{t \in p}(t.exp) \wedge \mathcal{D} = p\big].
\end{aligned}
$$

Window Operator

Union Operator

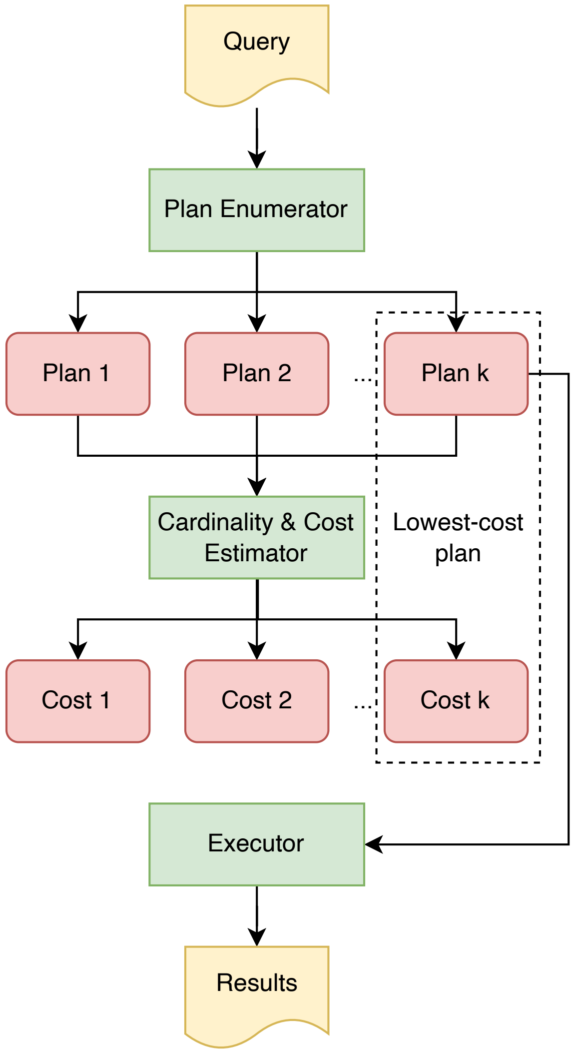
FILTER Operator



Logical Plan

[14]Pacaci A, Bonifati A, Özsu M T. Evaluating complex queries on streaming graphs[C]//2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, 2022: 272-285.

2. Graph Query Evaluation: Workflow

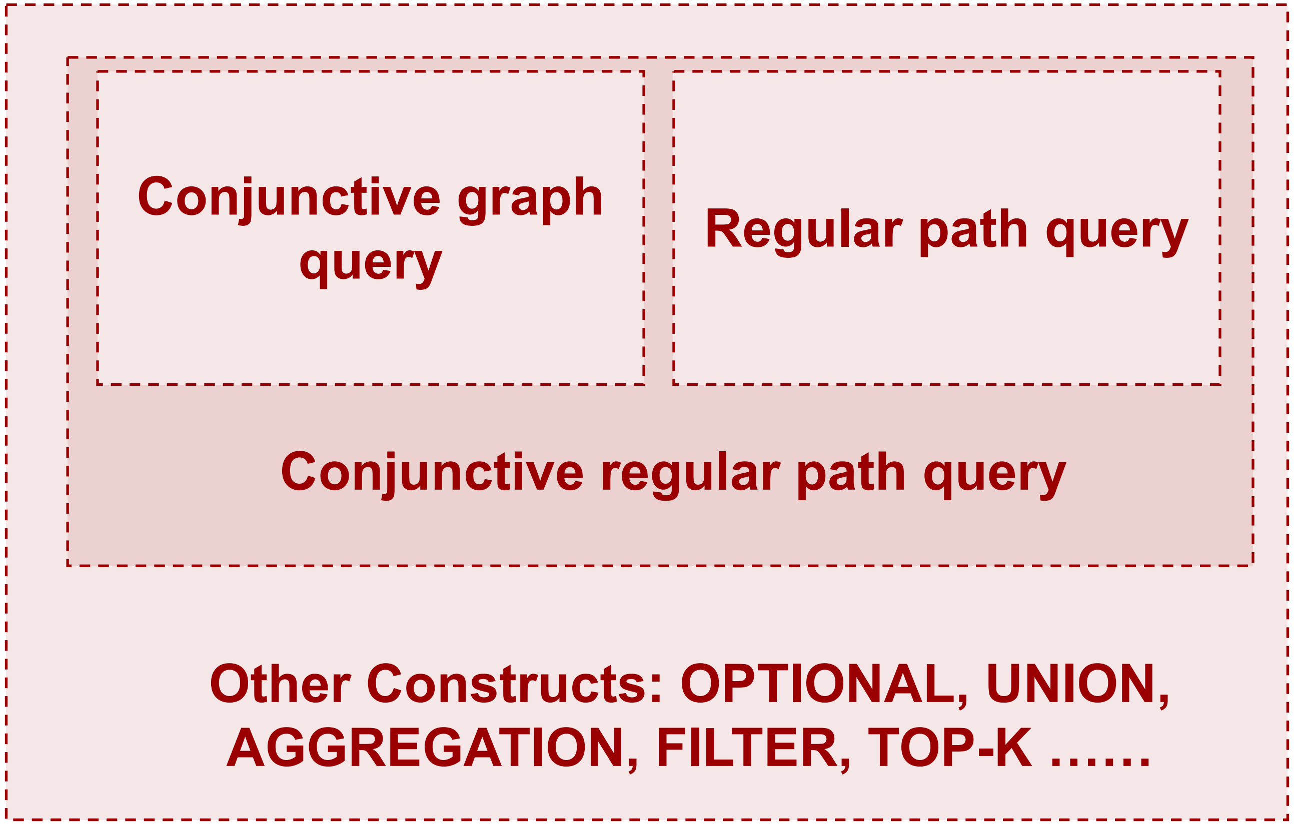


- The **overall workflow** of graph query planning is not much different from planning in relational databases
  - **Plan Enumerator:** Enumerates semantically equivalent query plans
  - **Cardinality & Cost Estimator:** estimates the cost of each query plan so that the executor can choose the expected cheapest plan

- However, there are the following **differences** in the actual planning procedure:
  - **Plan Representation:** due to the different query syntax & semantics
  - **Cost & cardinality estimation schemes:** due to the different storage scheme and physical operators

2. Graph Query Evaluation: Workflow

- Representing graph query plan in a uniform manner.

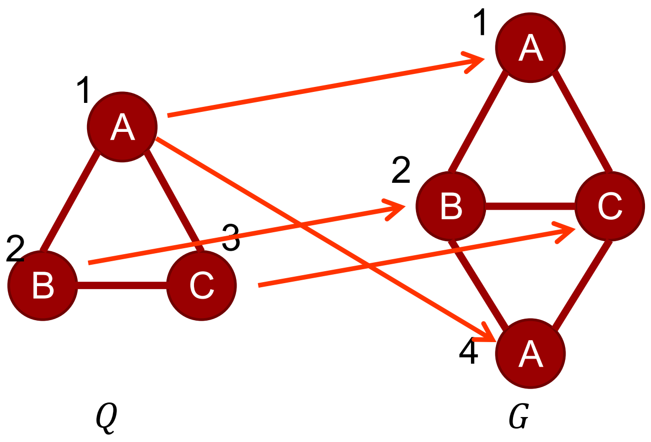


A Uniform Graph Query Engine

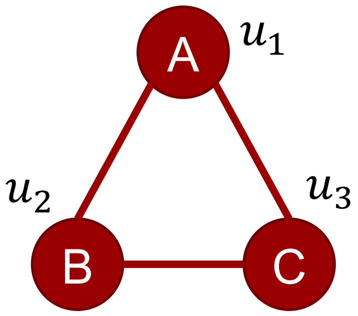
2. Graph Query Evaluation: Subgraph Query

Subgraph Query (Conjunctive Graph Query) : Given a query Q and a data graph G, Q is subgraph isomorphism to G, if and only if there exists an injective function f: V(Q) → $V(G)$, such that

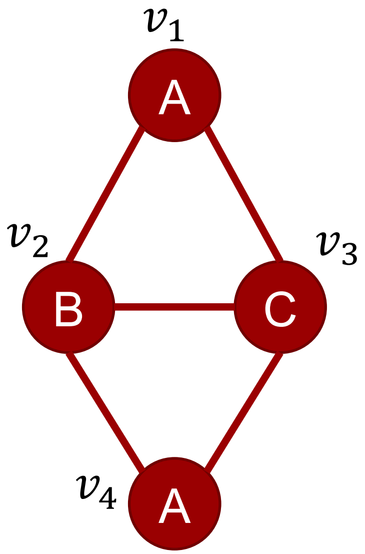
1. $\forall\, u \in V(Q), f(u) \in V(G), L_V(u) = L_V\big(f(u)\big)$, where $V(Q)$ and $V(G)$ denotes all vertices in Q and G, respectively; and $L_V(\cdot)$ denotes the corresponding vertex label.
2. $\forall \overline{u_1 u_2} \in E(Q), \overline{f(u_1)f(u_2)} \in E(G), L_E(\overline{u_1 u_2}) = L_E\big(\overline{f(u_1)f(u_2)}\big)$



Algorithms for joins: The core operator in subgraph queries



Query Graph Q



Data Graph G

Binary join

T_1

u_1	u_2
v_1	v_2
v_4	v_2
…	…

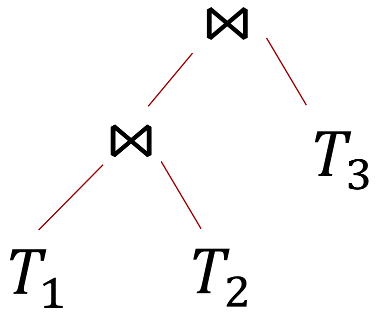
T_2

u_1	u_3
v_1	v_2
v_4	v_2
…	…

T_3

u_2	u_3
v_2	v_3

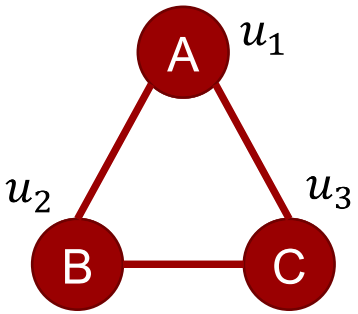
$$R = T_1 \bowtie T_2 \bowtie T_3$$



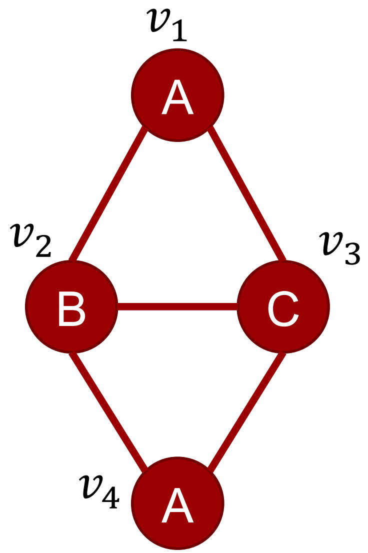
- Commonly used in relational databases
- Can be more efficient than worst-case-optimal join on acyclic query graphs

2. Graph Query Evaluation: Subgraph Query

Algorithms for joins: The core operator in subgraph queries

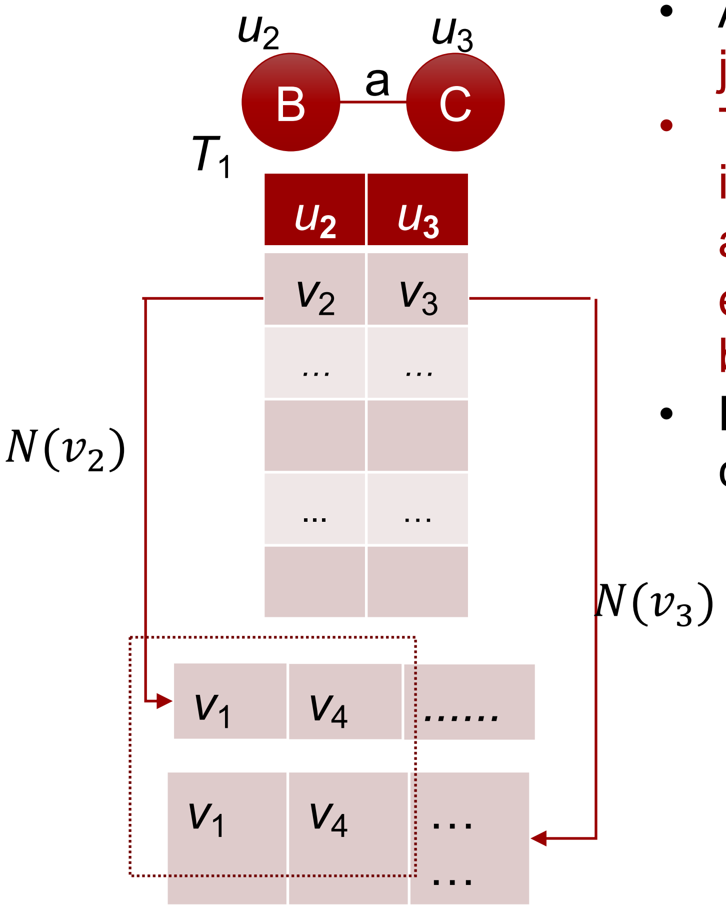


Query Graph



Data Graph

Worst-case optimal join

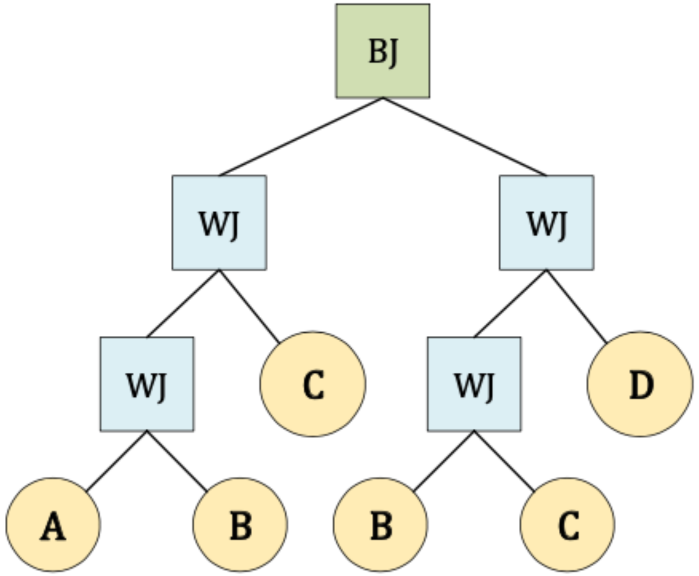


- A class of multi-way joins
- The number of intermediate results are guaranteed to not exceed the AGM bound [1]
- Especially efficient on cyclic query graphs

[1] Albert Atserias, Martin Grohe, and Dániel Marx. 2013. Size Bounds and Query Plans for Relational Joins. SIAM J. Comput. 42, 4 (2013), 1737–1767.

2. Graph Query Evaluation: Subgraph Query

A hybrid query plan representing subgraph query



An example hybrid query plan

Image taken from: Linglin Yang, Lei Yang, Yue Pang, and Lei Zou. 2022. GCBO: A Cost-based Optimizer for Graph Databases. (CIKM '22).

- Most existing works focus on **worst-case-optimal-join-only plans**, which extends the intermediate match by one query vertex at a time
  - The query planning problem is reduced to **query vertex ordering**
- [2] proposes a **hybrid plan space considering both binary and worst-case-optimal joins**
  - Benefits both acyclic and cyclic queries

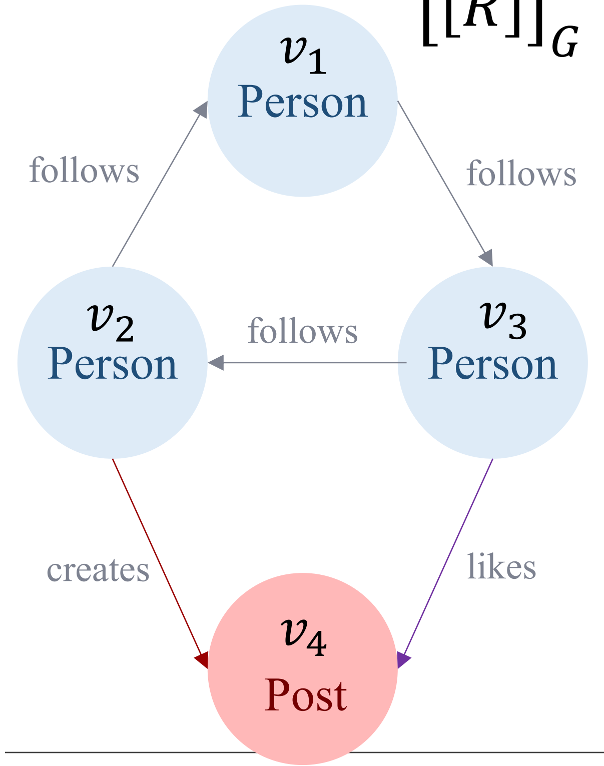
[2] C. R. Aberger, S. Tu, K. Olukotun, and C. Ré, "EmptyHeaded: A Relational Engine for Graph Processing," in Proceedings of the 2016 International Conference on Management of Data.
[3] A. Mhedhbi, and S. Salihoğlu, "Optimizing subgraph queries by combining binary and worst-case optimal joins," Proc. VLDB Endow., 2019.

- **An RPQ *R*** is a regular expression on the edge labels, with the following form:

$$R \to \epsilon \mid a \mid R_1 \,/\, R_2 \mid R_1 | R_2 \mid R? \mid R^* \mid R^+$$

R*'s result** on the edge-labeled directed graph $G = (V, E, \Sigma, \mathrm{l})$ is **the set of node pairs** with at least a path whose **edge label sequence satisfies *R

$$[[R]]_G = \{\langle u, v\rangle | \exists p \in G, p.s = u \wedge p.t = v \wedge l(p) \in L(R)\}$$



"People you follow indirectly"

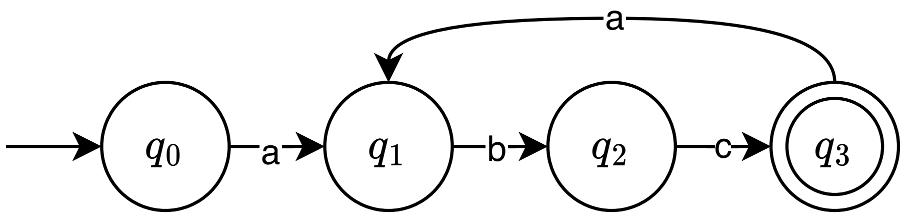
$[[follows^+]]_G$

$= \{\langle v_1, v_3\rangle, \langle v_1, v_2\rangle, \langle v_1, v_1\rangle, \langle v_3, v_2\rangle, \langle v_3, v_1\rangle, \langle v_3, v_3\rangle, \langle v_2, v_1\rangle, \langle v_2, v_3\rangle, \langle v_2, v_2\rangle\}$

"Posts liked by people you follow"

$[[follows \,/\, likes]]_G = \{\langle v_1, v_4\rangle\}$

1. Finite-automaton-based evaluation



$(a/b/c)^+$

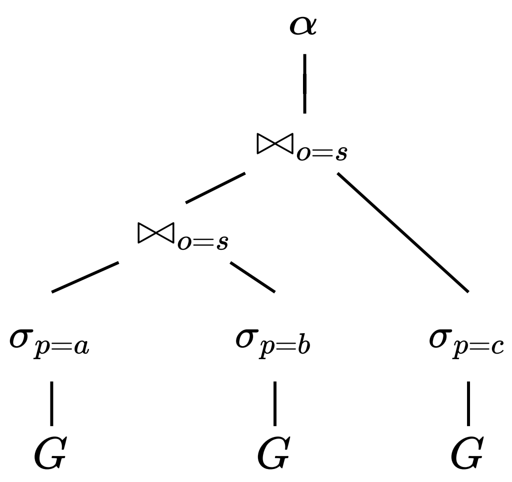
Naturally suited for native graph storage

- Converts the regular expression into a **finite automaton**
- Conceptually, compute the **product automaton** between the regex's automaton and the data graph, in which all the reachable node pairs are the results
- Realistically, the automaton **guides the search on the data graph**
- Processes Kleene closures by **looping** in the automaton

[4] André Koschmieder and Ulf Leser. 2012. Regular Path Queries on Large Graphs. In Scientific and Statistical Database Management, Anastasia Ailamaki and Shawn Bowers (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 177–194.
[5] Van-Quyet Nguyen, Quyet-Thang Huynh, and Kyungbaek Kim. 2022. Estimating Searching Cost of Regular Path Queries on Large Graphs by Exploiting Unit-Subqueries. Journal of Heuristics 28, 2 (April 2022), 149–169.

2. Graph Query Evaluation: Regular Path Query

2. α-relational-algebra-based evaluation



$(a/b/c)^{+}$

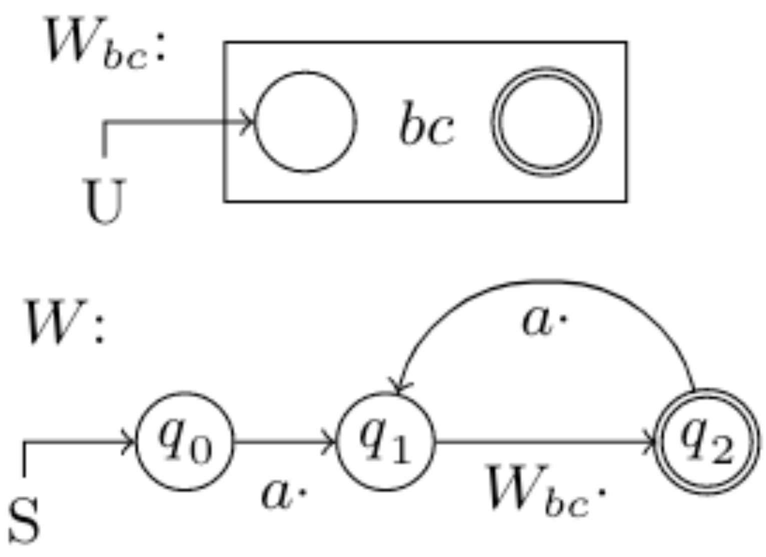
Naturally suited for hybrid storage

- Converts the regular expression into a **relational algebra tree extended with the α (fix-point) operator**
- Executes the tree bottom-up
  - α (fix-point): self-join the operand result table until no new rows are produced
- Processes Kleene closures by **α (fix-point) operators**, no intermediate states like automata
- Can be extended to support multi-query optimization [7]

[6] S. Dey, V. Cuevas-Vicenttín, S. Köhler, E. Gribkoff, M. Wang, and B. Ludäscher, "On implementing provenance-aware regular path queries with relational query engines," in Proceedings of the Joint EDBT/ICDT 2013 Workshops on - EDBT '13.
[7] Y. Pang, L. Zou, J. X. Yu, and L. Yang, "Materialized View Selection & View-Based Query Planning for Regular Path Queries," Proc. ACM Manag. Data 2, 3, Article 152 (June 2024).

PEKING UNIVERSITY 北京大學

3. A hybrid query plan for regular path query [8]

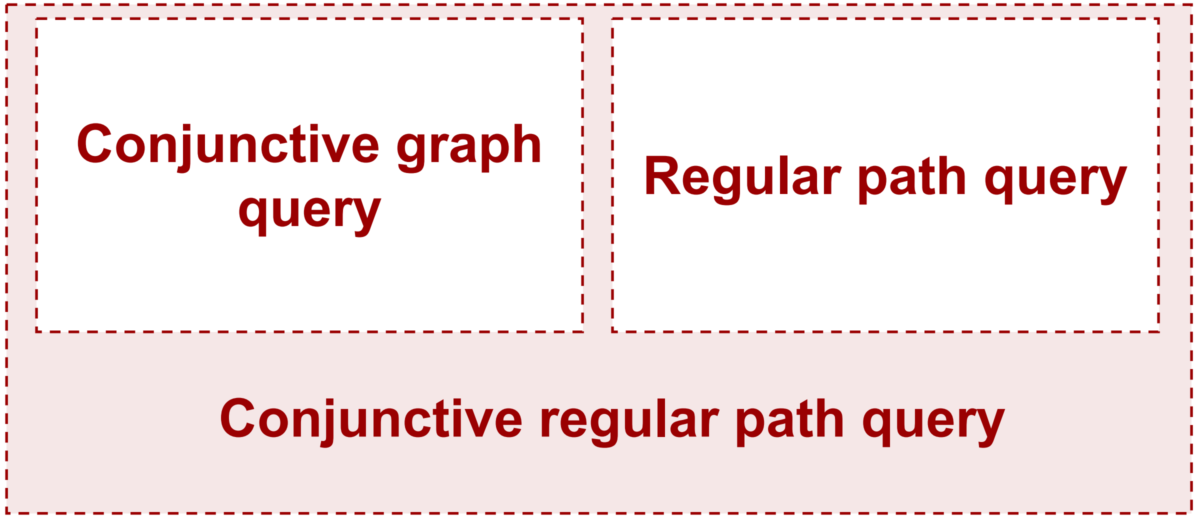


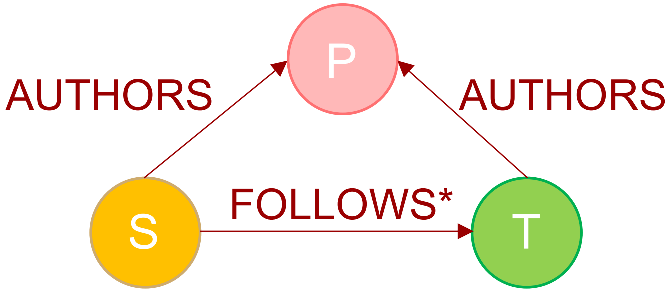
$(a/b/c)^{+}$

- Plan space **subsumes** those of the previous two evaluation methods
- Allows **multiple automata** in a plan with **bidirectional traversal**
- Can **materialize** the result of certain automata to **use as transitions** in other automata (e.g., W_{bc} in the figure)
  - Simulates intermediate result tables in relational algebra
- Allows **"partial loop caching"** of Kleene closures – a point on the spectrum between automaton loop & fix-point

[8] N. Yakovets, P. Godfrey, and J. Gryz, "Query Planning for Evaluating SPARQL Property Paths," in Proceedings of the 2016 International Conference on Management of Data.

2. Graph Query Evaluation: Conjunctive Regular Path Query





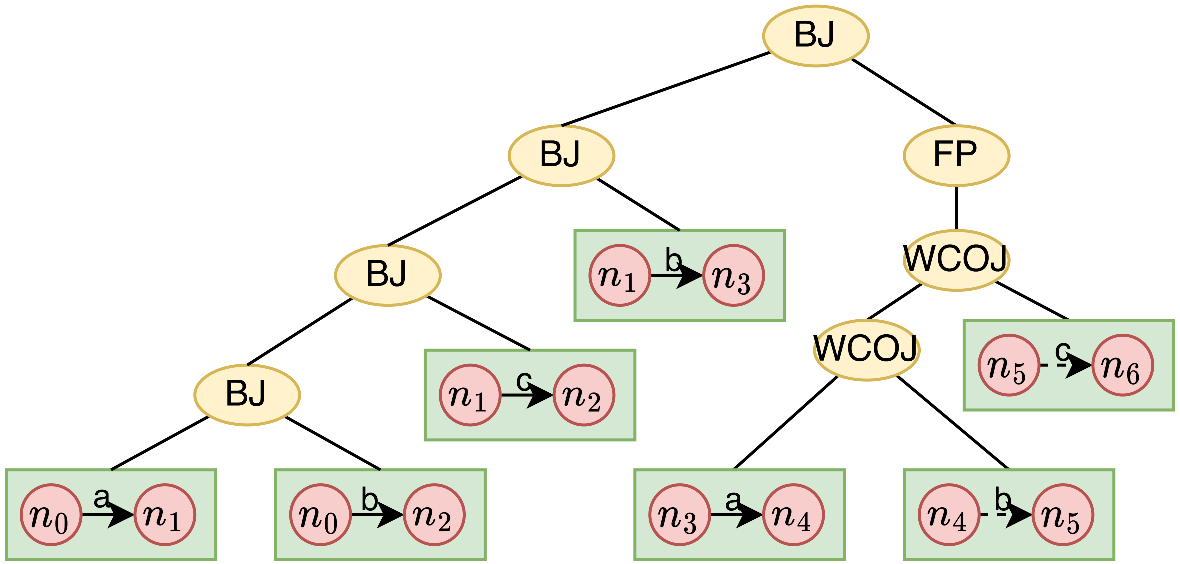
A **conjunctive regular path query (CRPQ)** is a conjunction of regular path queries (RPQs):

$$Q(\bar{x}) \leftarrow \bigwedge_{i=1}^{l} R_{a_i}(y_i, z_i) \wedge \bigwedge_{i=l+1}^{k} r_i(y_i, z_i)$$

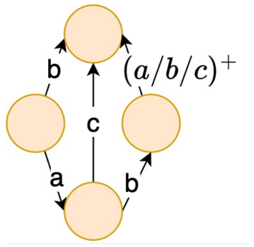
- $a_i \in \Sigma$ (edge labels)
- r_i: RPQs
- $\bar{x} = \{x_1, \dots, x_n\} \subseteq \{y_1, z_1, \dots, y_k, z_k\}$ (output variables)

We can view CRPQs as **subgraph matching queries whose edges can be specified by either edge labels or RPQs**

Binary-join-based evaluation [9]



Query:

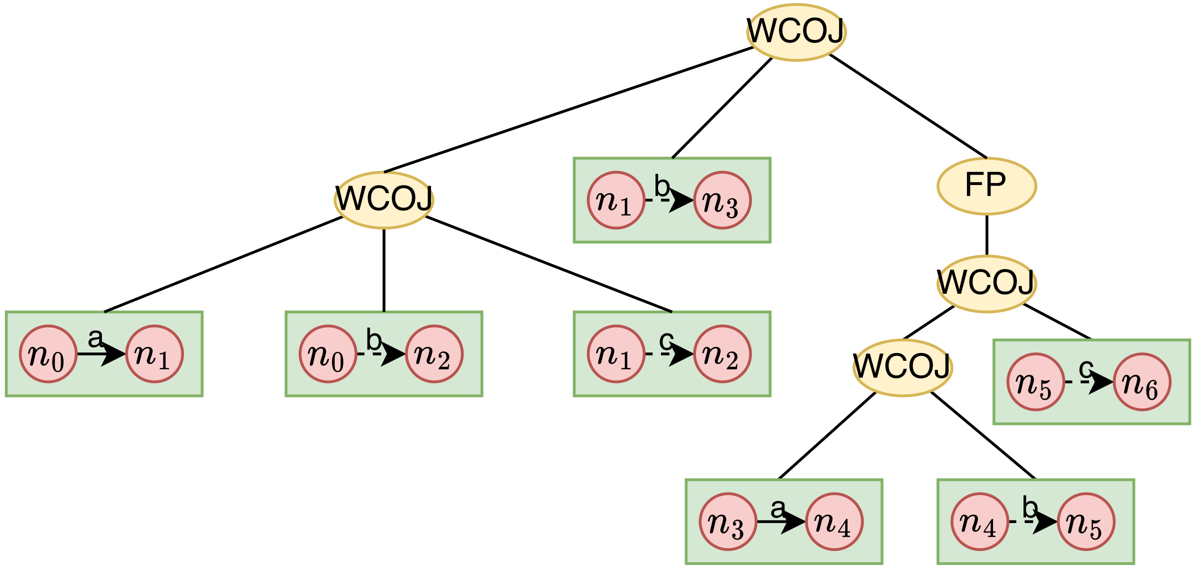


- Considers **binary-joining** the triple patterns to get the CRPQ result
- The CRPQ query planning problem is thus reduced to a **join ordering** problem
- Uses the **ALP** procedure as in the SPARQL 1.1 specification to evaluate RPQs, which is basically **BFS guided by finite automata** (i.e., **fixed RPQ plans**)
  - The WCOJ nodes in the figure are used to simulate the automata-guided traversal

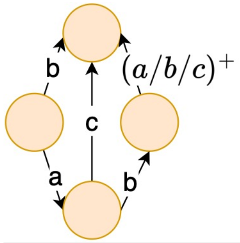
[9] J. Aimonier-Davat, H. Skaf-Molli, P. Molli, M.-H. Dang, and B. Nédelec, "Join Ordering of SPARQL Property Path Queries," in *The Semantic Web*, vol. 13870, 2023.

PEKING UNIVERSITY

Worst-case-optimal-join-based evaluation [10-11]



Query:

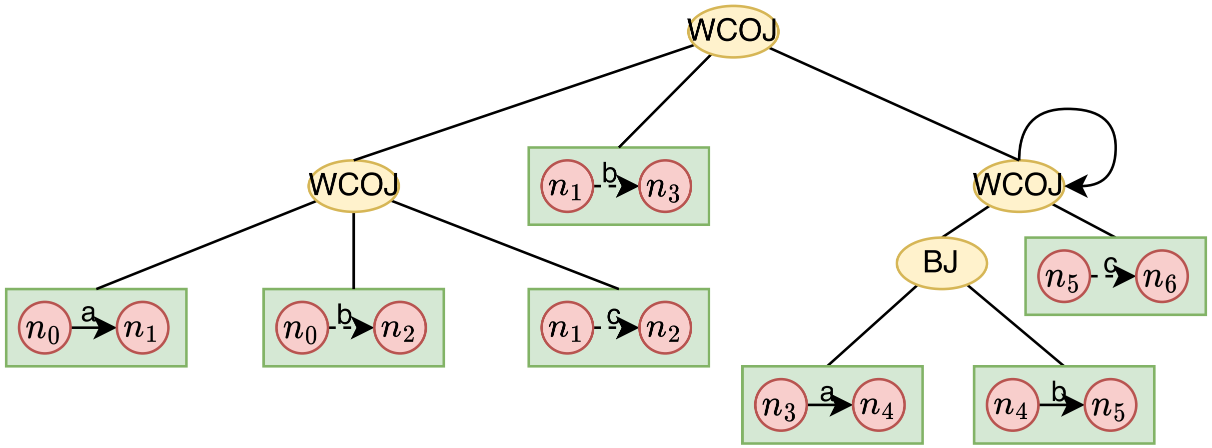


- Considers **worst-case-optimal-joining** the triple patterns to get the CRPQ result (i.e., extending **one variable at a time**)
- The CRPQ query planning problem is thus reduced to a **variable ordering** problem
- Also uses **worst-case-optimal joins to plan the RPQs without Kleene closures**, since they can be viewed as ”chain BGPs”
- Uses **fix-point** to evaluate Kleene closures in RPQs
- Constrains that **at most 1 RPQ can take part in each WCOJ**; all the others are **checked** for satisfaction after the join

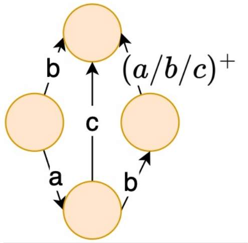
[10] T. A. Cucumides Faúndez, “Size bounds and algorithms for conjunctive regular path queries,” Mar. 2022. doi: 10.7764/tesisUC/ING/63591.
[11] N. Karalis, A. Bigerl, L. Heidrich, M. A. Sherif, and A.-C. N. Ngomo, “Efficient Evaluation of Conjunctive Regular Path Queries Using Multi-way Joins,” 2024.

2. Graph Query Evaluation: Conjunctive Regular Path Query

Unified plan space for CRPQs



Query:

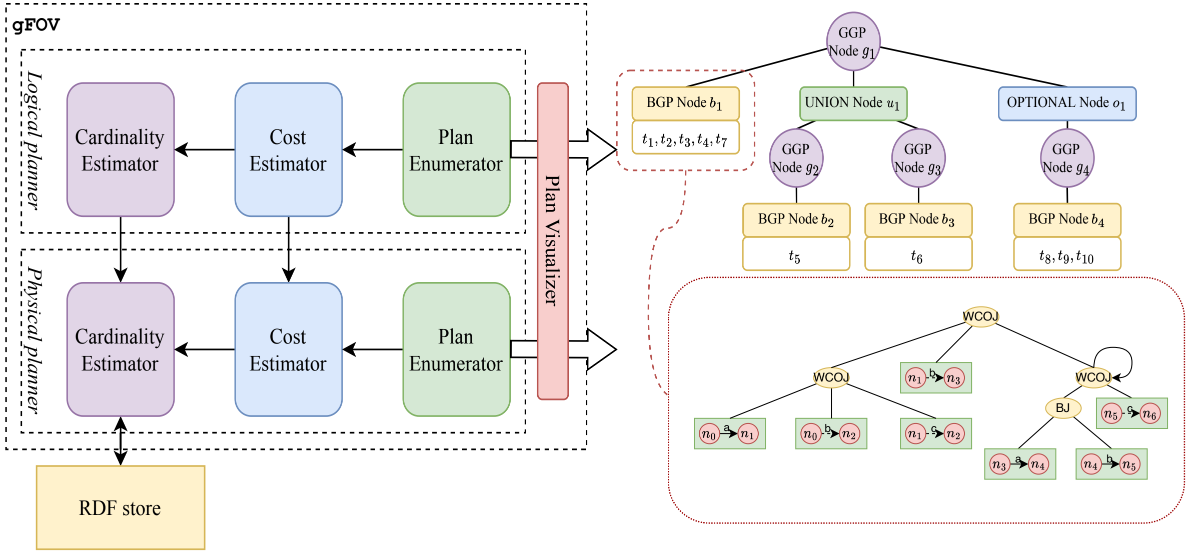


- Represents RPQ and subgraph matching by the **same set of operators**
- **Incorporates all state-of-the-art** subgraph matching and RPQ planning techniques
- Can express plans that were previously inexpressible: **hybrid subgraph matching + hybrid RPQ plan**

2. Graph Query Evaluation: Other Graph Query Constructs

Beyond conjunctive graph & regular path (CPRQ) queries

- **Scarcely any work** addresses the planning of graph query constructs beyond subgraph matching and RPQs, e.g., aggregation, top-k [13], UNION, and OPTIONAL
- Though these constructs have counterparts in relational queries, **graph queries may benefit from jointly optimizing them with subgraph matching and RPQs**
  - [12] explores this incipiently by jointly optimizing the UNION and OPTIONAL operators with conjunctive graph queries

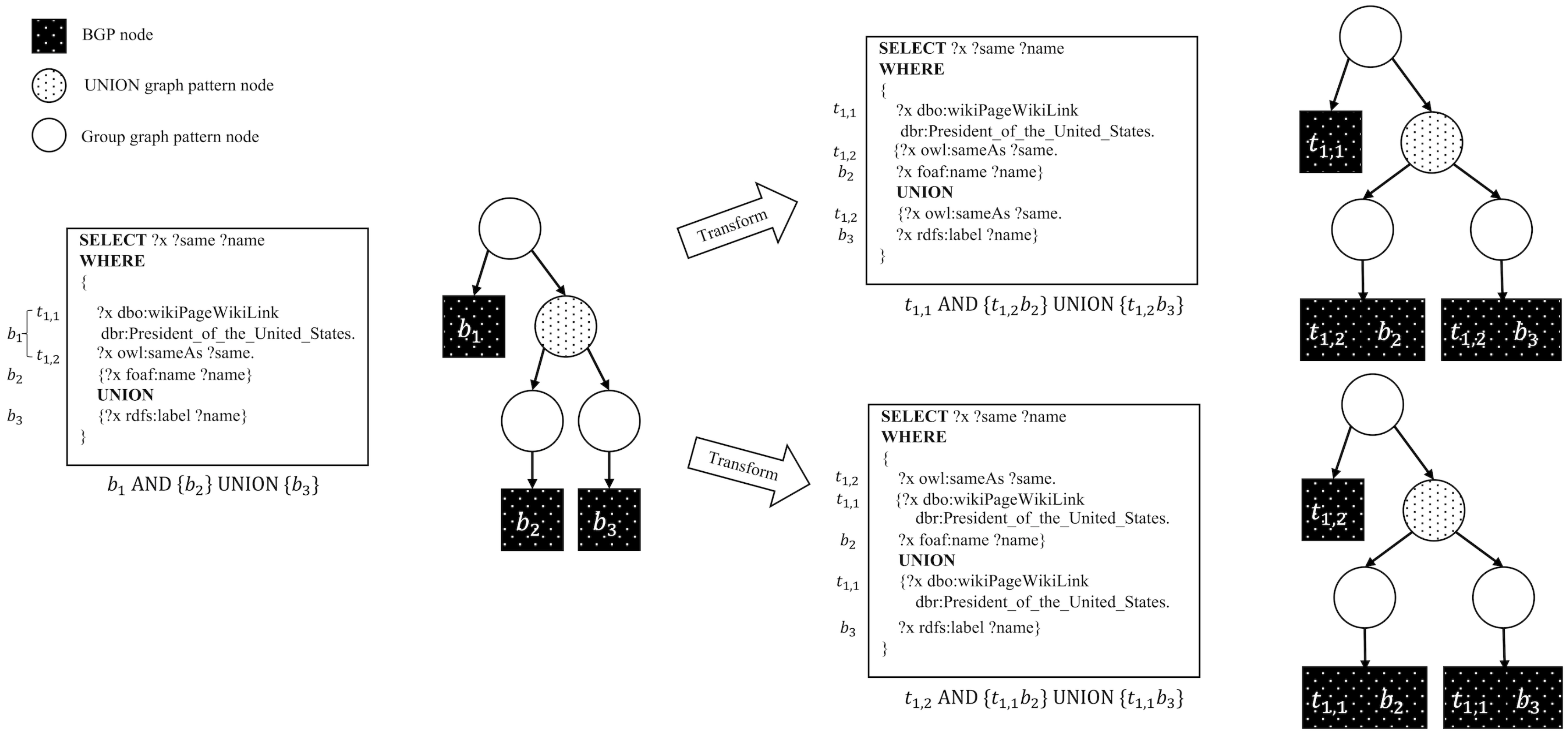


[12] Y. Pang, L. Yang, L. Zou, and M. T. Özsu, "gFOV: A Full-Stack SPARQL Query Optimizer & Plan Visualizer," CIKM 2023.
[13] L. Yang, Y. Zhou, Y. Pang, and L. Zou, "Efficient Pruned Top-K Subgraph Matching with Topology-Aware Bounds," CIKM 2024.

2. Graph Query Evaluation: Other Graph Query Constructs

Query Re-Writing: Aims to reduce the intermediate result sizes.

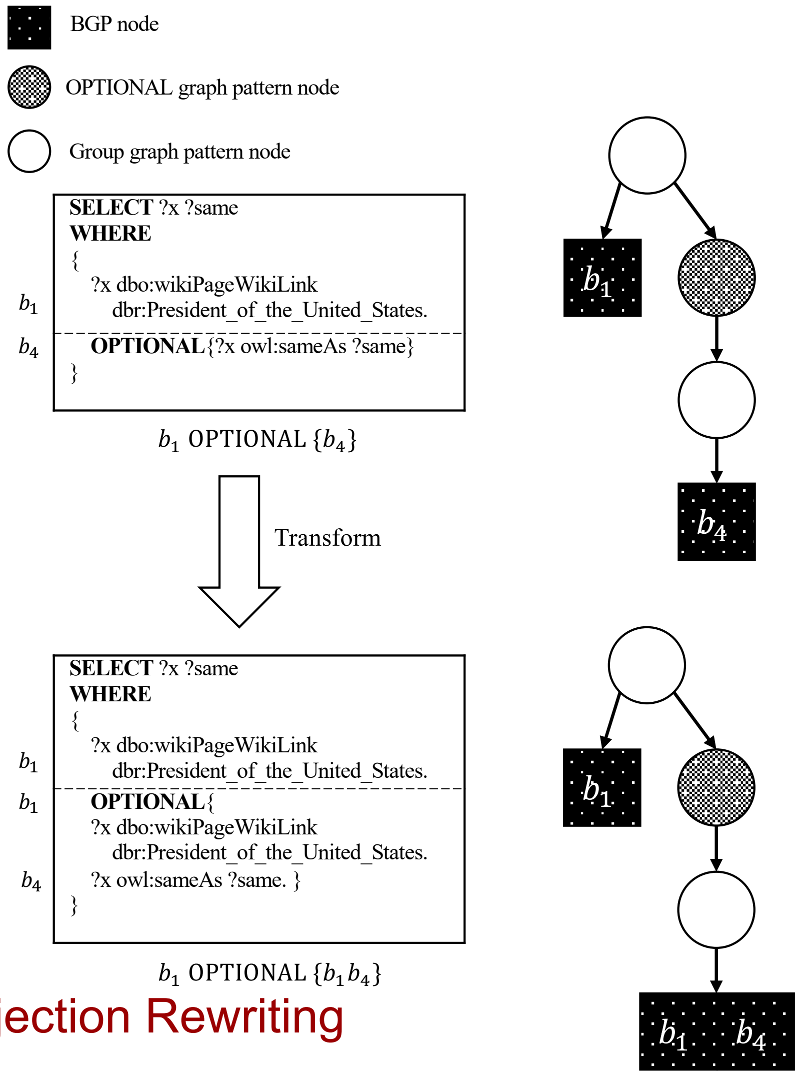
[[$P1$ AND { {$P2$ } UNION {$P3$ } }]] = [[{$P1$ AND $P2$ } UNION {$P1$ AND $P3$ }]]



Merge Rewriting

Query Re-Writing: **Aims to reduce the intermediate result sizes.**

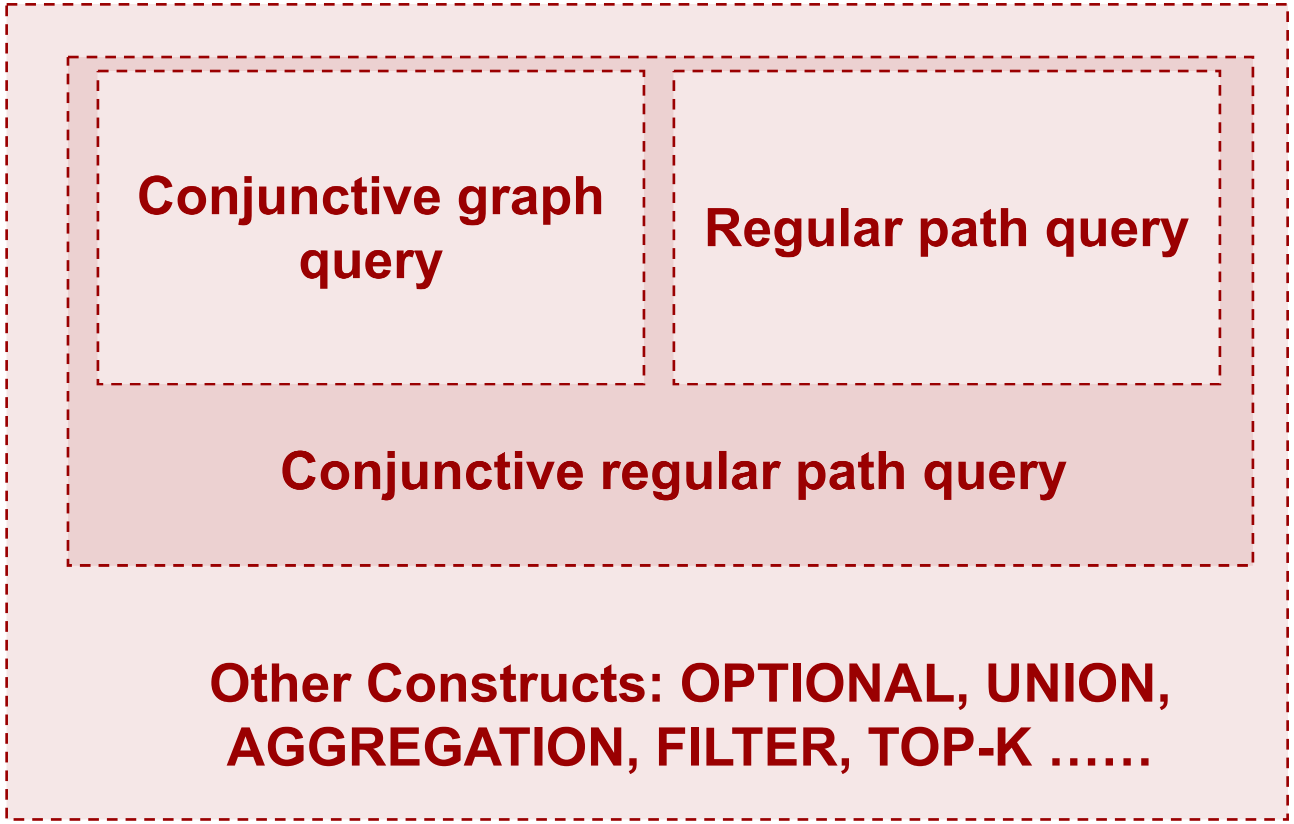
[[$P1$ OPTIONAL {$P2$ }]] = [[$P1$ OPTIONAL {$P1$ AND $P2$ }]]



Injection Rewriting

3. Cardinality Estimation

Cardinality estimators for each component



A Uniform Graph Query Engine

3. Cardinality Estimation : Subgraph Query

- **Largely based on the same frameworks as relational databases:** synopses, sampling, and machine learning
- **More challenging to estimate than relational join queries due to richer joins.** To address:
  - Building synopses based on subgraph patterns [14]
  - Devising sampling strategies that reduce the sampling space [14]
  - Using graph neural networks (GNN) to capture the graph structure [15]
  - …

Cost estimation

- **Essentially distinct from its relational counterpart** due to the different join methods and storage schemes

[15] K. Kim, H. Kim, G. Fletcher, and W. Han, “Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation,” SIGMOD 2021.
[16] T. Schwabe and M. Acosta, “Cardinality Estimation over Knowledge Graphs with Embeddings and Graph Neural Networks,” Proc. ACM Manag. Data 2, 1, Article 44 (March 2024).

3. Cardinality Estimation : Regular Path Query

PEKING UNIVERSITY

- [8] proposes a statistic-based cost & cardinality estimation framework for their proposed hybrid RPQ plans
- [7] proposes a cost & cardinality estimation framework for multi-query RPQ plans based on α-relational-algebra, using both statistics and online sampling to enhance scalability
- [5] proposes a statistic-based cost & cardinality estimation framework for finite-automata-based plans

All the methods above **assume the maximum length of regular paths with Kleene closures** are known in certain cases, which is usually not the case

How to **effectively remove this assumption** is an important **open problem**

[8] Nikolay Yakovets, Parke Godfrey, and Jarek Gryz. 2016. Query Planning for Evaluating SPARQL Property Paths. In Proceedings of the 2016 International Conference on Management of Data.

[7] Y. Pang, L. Zou, J. X. Yu, and L. Yang, "Materialized View Selection & View-Based Query Planning for Regular Path Queries," Proc. ACM Manag. Data 2, 3, Article 152 (June 2024).

[5] Van-Quyet Nguyen, Quyet-Thang Huynh, and Kyungbaek Kim. 2022. Estimating Searching Cost of Regular Path Queries on Large Graphs by Exploiting Unit-Subqueries. Journal of Heuristics 28, 2 (April 2022), 149–169.

3. Cardinality Estimation : Other Graph Query Constructs

北京大学 PEKING UNIVERSITY

UNION operator's cost: $$f_U(c_i) = \sum_{c_j \in \mathrm{child}(c_i)} |c_j|$$

OPTIONAL operator's cost: $f_O(c_i) = f_G(c_j)$, where c_j is c_i's child GGP

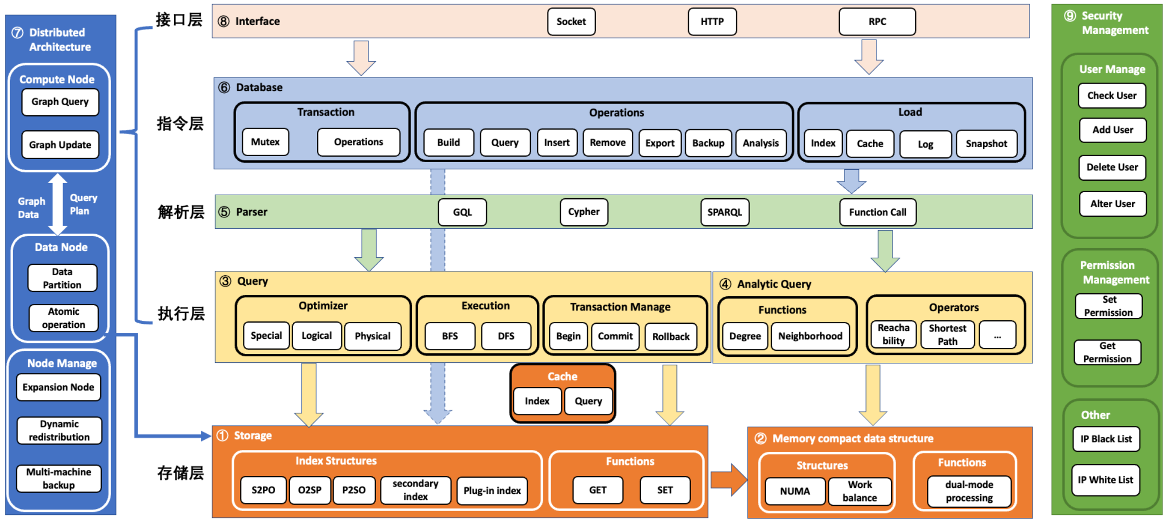
Since these query construct have **direct counterparts in relational queries**, it is still an open problem whether their cost & cardinality

- Should be estimated differently on graphs?
- Can be estimated more efficiently using novel techniques on graphs?

[12] Y. Pang, L. Yang, L. Zou, and M. T. Özsu, "gFOV: A Full-Stack SPARQL Query Optimizer & Plan Visualizer," CIKM 2023.

gStore: Our Open-Source RDF Graph Database

Starred 777



✓ **Native graph storage**

✓ **Uniform query plan representation** (binary + worst-case-optimal joins) for conjunctive graph queries and regular path query.

✓ Supports **billion-edge RDF graphs** on a single server

In the near future:

Support both RDF & property graphs

- Same parsing, planning, and evaluation workflow
- Unified query plan representation
- Adaptive storage

【Lei Zou, et al, gStore: Answering SPARQL Queries Via Subgraph Matching, in Proceedings of 37th International Conference on very Large Databases (VLDB), 2011】

Take Home Message

北京大學 PEKING UNIVERSITY

- **A uniform graph query representation** lays the basis for graph query evaluation and optimization, which requires more extensive research.

- **Cardinality Estimator** for different graph query components are vital in optimizing graph planning.

- **The interactions and interdependence** between the query layer and the storage layer in a graph database are worthy of further investigation..

References

[1] A. Mhedhbi and S. Salihoğlu, “Modern techniques for querying graph-structured relations: foundations, system implementations, and open challenges,” Proc. VLDB Endow., 2022.
[2] A. Mhedhbi, and S. Salihoğlu, "Optimizing subgraph queries by combining binary and worst-case optimal joins,” Proc. VLDB Endow., 2019.
[3] N. Yakovets, P. Godfrey, and J. Gryz, “Query Planning for Evaluating SPARQL Property Paths,” in Proceedings of the 2016 International Conference on Management of Data.
[4] Y. Pang, L. Zou, J. X. Yu, and L. Yang, “Materialized View Selection & View-Based Query Planning for Regular Path Queries,” Proc. ACM Manag. Data 2, 3, Article 152 (June 2024).
[5] Y. Pang, L. Yang, L. Zou, and M. T. Özsu, “gFOV: A Full-Stack SPARQL Query Optimizer & Plan Visualizer,” CIKM 2023.
[6] L. Yang, Y. Zhou, Y. Pang, and L. Zou, “Efficient Pruned Top-K Subgraph Matching with Topology-Aware Bounds,” CIKM 2024.
[7] K. Kim, H. Kim, G. Fletcher, and W. Han, “Combining Sampling and Synopses with Worst-Case Optimal Runtime and Quality Guarantees for Graph Pattern Cardinality Estimation,” SIGMOD 2021.
[8] T. Schwabe and M. Acosta, “Cardinality Estimation over Knowledge Graphs with Embeddings and Graph Neural Networks,” Proc. ACM Manag. Data 2, 1, Article 44 (March 2024).
[9] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” SIAM J. Sci. Comput. 20(1), 1998.
[10] P. Peng, M. T. Özsu, L. Zou, C. Yan, and C. Liu, “MPC: Minimum property-cut RDF graph partitioning,” ICDE 2022.
[11] P. Peng, S. Ji, M. T. Özsu, and L. Zou, “Minimum motif-cut: a workload-aware RDF graph partitioning strategy,” The VLDB Journal (2024).
[12] R. Angles, C. B. Aranda, A. Hogan, C. Rojas, and D. Vrgoč, “WDBench: A Wikidata Graph Query Benchmark,” ISWC 2022.
[13] M. Morsey, J. Lehmann, S. Auer, A.-C. Ngonga Ngomo, “DBpedia SPARQL benchmark – performance assessment with real queries on real data,”ISWC 2011.
[14]Pacaci A, Bonifati A, Özsu M T. Evaluating complex queries on streaming graphs[C]//2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, 2022: 272-285.
[15] Y. Guo, Z. Pan, and J. Heflin, “LUBM: A benchmark for OWL knowledge base systems,” J. Web Semantics 3(2-3), 158–182 (2005).
[16] G. Bagan, A. Bonifati, R. Ciucanu, G. H. L. Fletcher, A. Lemay, and N. Advokaat, “gMark: Schema-Driven Generation of Graphs and Queries,” IEEE Trans. Knowl. Data Eng., vol. 29, no. 4, pp. 856–869, Apr. 2017.
[17] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee, “Diversified Stress Testing of RDF Data Management Systems,” ISWC 2014.
[18] O. Erling et al., “The LDBC Social Network Benchmark: Interactive Workload,” SIGMOD 2015.
[19] Ldbc. (n.d.). Ldbc/LDBC_FINBENCH_DOCS: Specification of the LDBC financial benchmark. GitHub. https://github.com/ldbc/ldbc_finbench_docs.

北京大學
PEKING UNIVERSITY

Thank you for listening!