DATALAB
@Northeastern

Efficient Ranked Access over Joins

VLDB 2023 PhD Workshop

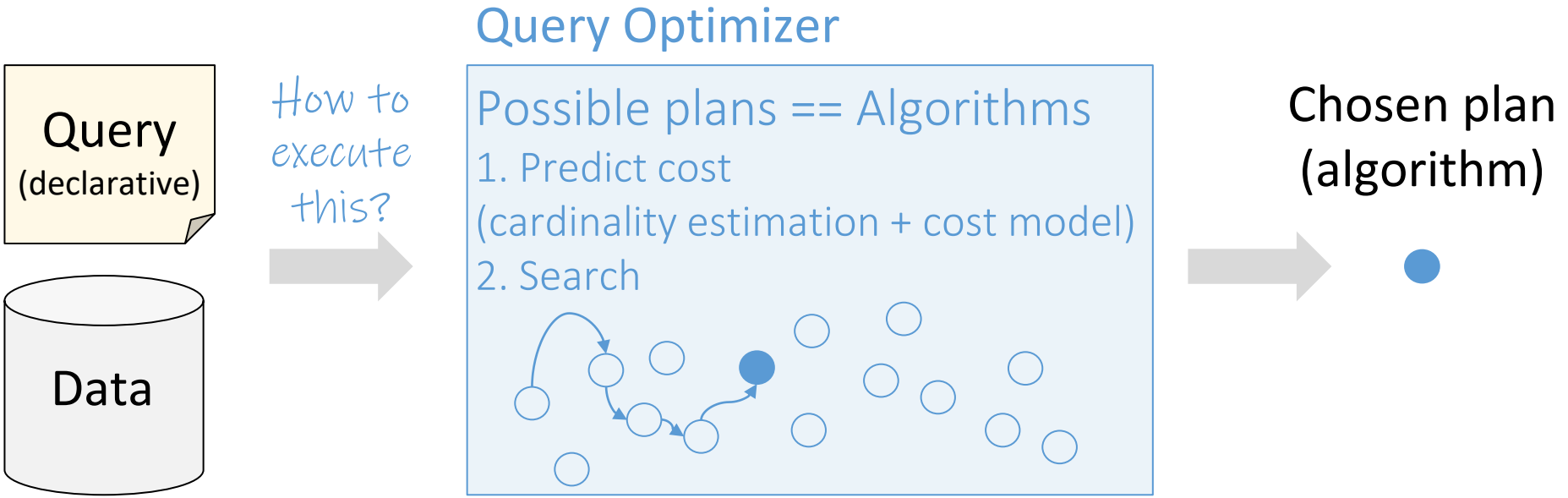
Nikolaos Tziavelis

Advisors: Wolfgang Gatterbauer, Mirek Riedewald

Northeastern University, Boston

Website: https://northeastern-datalab.github.io/anyk/

Query Processing & Optimization in Data Systems



… but which algorithms does the query optimizer know?

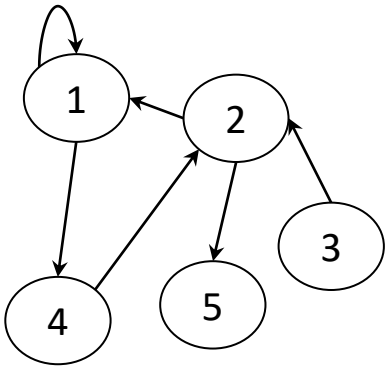
- Worst-case optimal joins not known until 2012 [N+12]
- Factorization techniques for aggregates only recently starting to be adopted [G+21]
- For complex queries, only naïve algorithms available

[N+12] Ngo, Porat, Ré, Rudra. Worst-case optimal join algorithms. PODS'12 https://doi.org/10.1145/2213556.2213565

[G+21] Gupta, Mhedhbi, Salihoglu. Columnar Storage and List-based Processing for Graph Database Management Systems. VLDB'21 https://doi.org/10.14778/3476249.3476297

Many-to-Many Joins

- Many-to-many joins are challenging because the output can be huge.
- Ex: Subgraph queries (2-hop paths, tree patterns, triangles, etc.)



2-hop paths

TO=FROM ⋈

EDGES

FROM	TO
1	1
1	4
2	1
2	5
3	2
4	2

EDGES

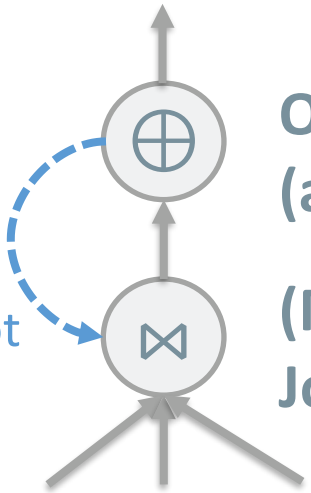
FROM	TO
1	1
1	4
2	1
2	5
3	2
4	2

Relations of size $n \Rightarrow$ Join output $O(n^2)$

… and even larger with more joins

Avoiding Join Materialization

- Materializing the join output may be infeasible (regardless of the join algorithm) because of its size.
- Often, we may not be interested in the join itself but in some operation on top of the join.
- Can we then avoid materializing it?



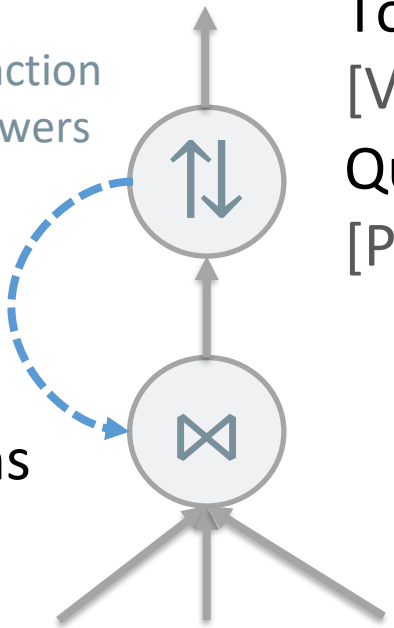
Operator with small output (aggregation, top-k, median, …)

(Multi-way) Join

"Push down"?
→ combined algorithm whose running time does not depend on the join output

Research Overview

"Ranked Access":
1) Order by ranking function
2) Retrieve specific answers according to ranking



No knowledge of k
(~incremental sorting)

Top-k / Any-k
[VLDB'20, arXiv'22]

Quantiles / Direct Access
[PODS'21, TODS'23, PODS'23]

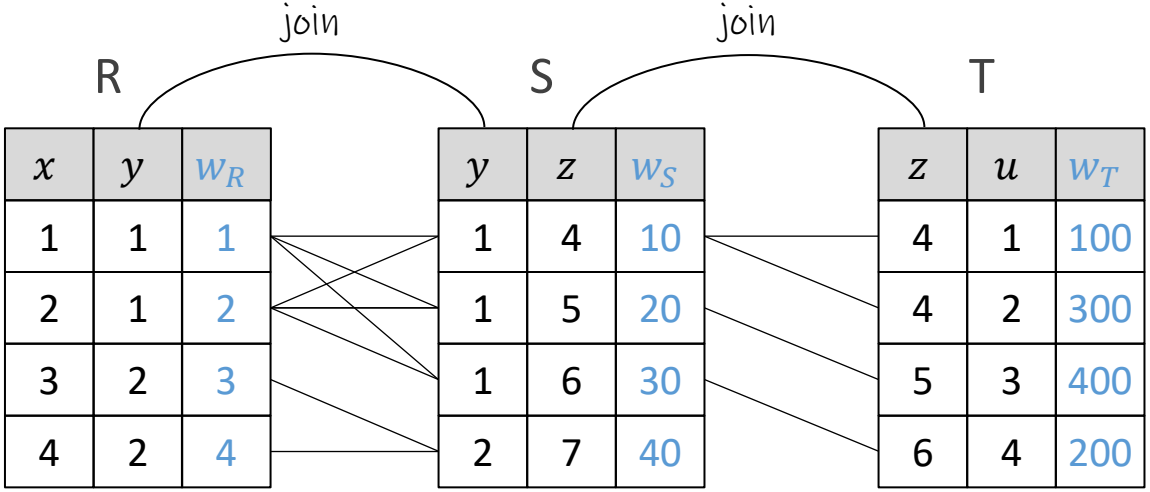
More general inequality joins
[VLDB'21]

[VLDB'20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB'20 https://doi.org/10.14778/3397230.3397250
[VLDB'21] Tziavelis, Gatterbauer, Riedewald. Beyond Equi-joins: Ranking, Enumeration and Factorization. PVLDB'21 https://doi.org/10.14778/3476249.3476306
[PODS'21] Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald. Tractable Orders for Direct Access to Ranked Answers of Conjunctive Queries. PODS'21 https://doi.org/10.1145/3452021.3458331
[arXiv'22] Tziavelis, Gatterbauer, Riedewald. Any-k Algorithms for Enumerating Ranked Answers to Conjunctive Queries. arXiv'22 https://arxiv.org/abs/2205.05649
[TODS'23] Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald. Tractable Orders for Direct Access to Ranked Answers of Conjunctive Queries. TODS'23 https://doi.org/10.1145/3578517
[PODS'23] Tziavelis, Carmeli, Gatterbauer, Kimelfeld, Riedewald. Efficient Computation of Quantiles over Joins. PODS'23 https://doi.org/10.1145/3584372.3588670

Outline

- Motivation & Research Overview
- Highlights of Results
  - Any-k
  - Quantile Queries
- Future Directions

Any-k for Joins: Example



Ranking function

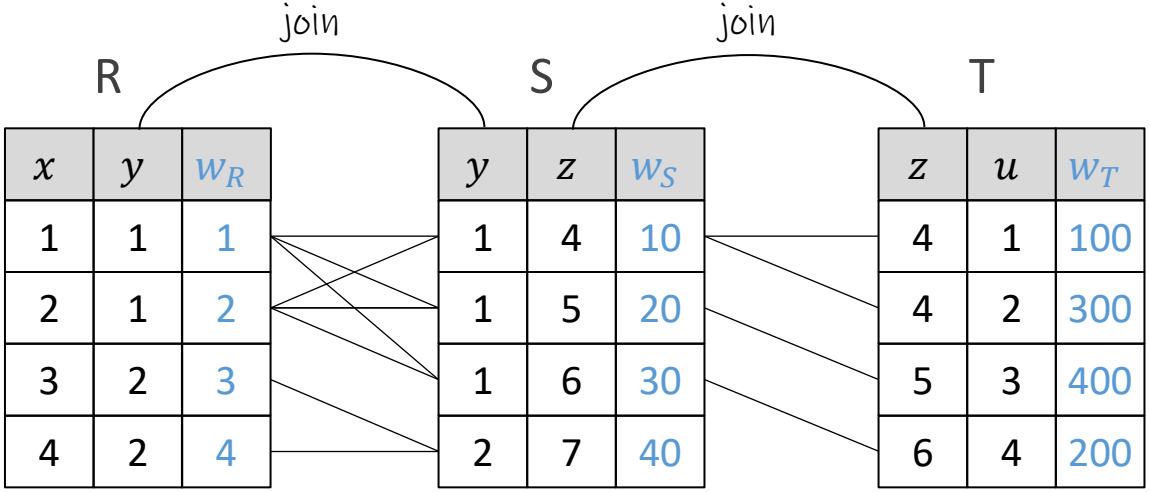
```
select  x, y, z, u,
        w_R + w_S + w_T as weight
from    R, S, T
where   R.y=S.y and S.z=T.z
order by weight ASC
limit k  any-k
```

Increasing sum of weights

(1, 1, 4, 1, 111) → (2, 1, 4, 1, 112) → (1, 1, 6, 4, 231) → …

[VLDB'20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB'20
https://doi.org/10.14778/3397230.3397250

Any-k for Joins: Ranking Functions and Complexity



x	y	w_R
1	1	1
2	1	2
3	2	3
4	2	4

y	z	w_S
1	4	10
1	5	20
1	6	30
2	7	40

z	u	w_T
4	1	100
4	2	300
5	3	400
6	4	200

Ranking functions

SUM: 1+10+100

MAX: max(1,10,100)

LEX: first w_R, then w_R, then w_T

+ acyclic join →

Time-to-kth answer

$O(|\mathrm{DB}| + k \log k)$

independent of join output size

[VLDB'20] Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB'20
https://doi.org/10.14778/3397230.3397250

Any-k and Shortest Paths

- Any-k = generalization of k-shortest paths on a Directed Acyclic Graph
- We adapt and improve:

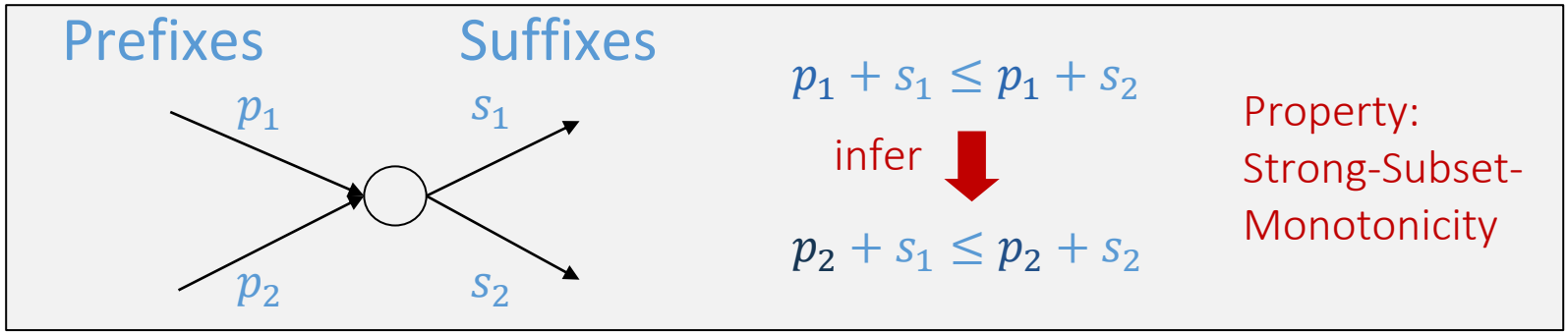
path length

#nodes

$$O(|G| + k(\log k + \ell)) \longrightarrow O(|G| + k(\log N + \ell))$$

Previously best known [E98]

Improvement for large k (k can be N^ℓ)
(for small k, $O(|G|)$ dominates)
Faster than sorting for entire sorted output



[arXiv'22] Tziavelis, Gatterbauer, Riedewald. Any-k Algorithms for Enumerating Ranked Answers to Conjunctive Queries. arXiv'22 https://arxiv.org/abs/2205.05649
[E98] Eppstein. Finding the k shortest paths. SIAM Journal on computing 1998. https://doi.org/10.1137/S0097539795290477

Any-k Implementation

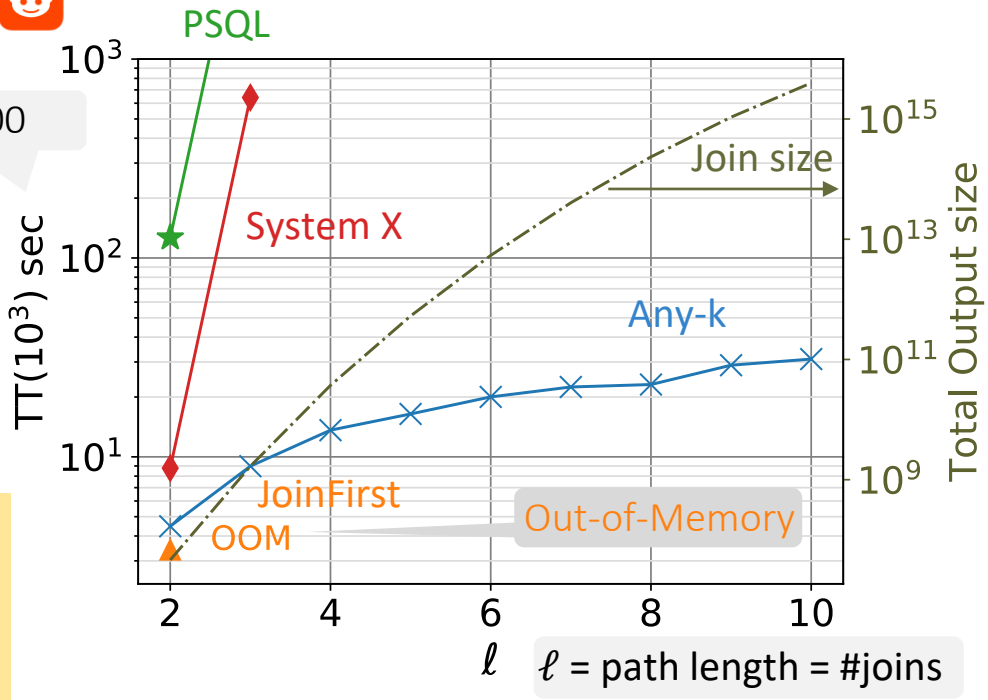
Query: Negative-sentiment paths on Reddit

- 572k edges
- length-ℓ paths
- timestamps in increasing order
- sentiment in decreasing order
- top results by sum of readability

Top-1000

SQL for $\ell = 2$:

```
select *, R1.Readability + R2.Readability as weight
from Reddit R1, Reddit R2
where  R2.Source = R1.Target
       AND R2.Timestamp > R1.Timestamp
       AND R2.Sentiment < R1.Sentiment
order by weight desc
limit 1000
```



[VLDB'21] Tziavelis, Gatterbauer, Riedewald. Beyond Equi-joins: Ranking, Enumeration and Factorization. PVLDB'21 https://doi.org/10.14778/3476249.3476306

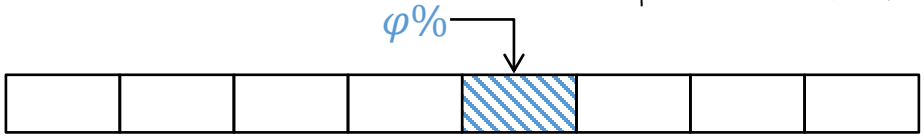
Outline

- Motivation & Research Overview
- Highlights of Results
  - Any-k
  - Quantile Queries
- Future Directions

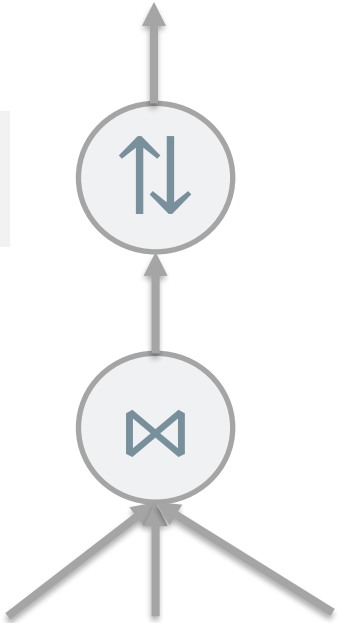
Quantile Join Queries

relative position 50%

e.g., pair of employees from the same department with median total salary



SUM, MAX, MIN, LEX, …



When can we find the quantile without computing the join?

==

Achieve time

$O(|\mathrm{DB}|\ \mathrm{polylog}|\mathrm{DB}|)$

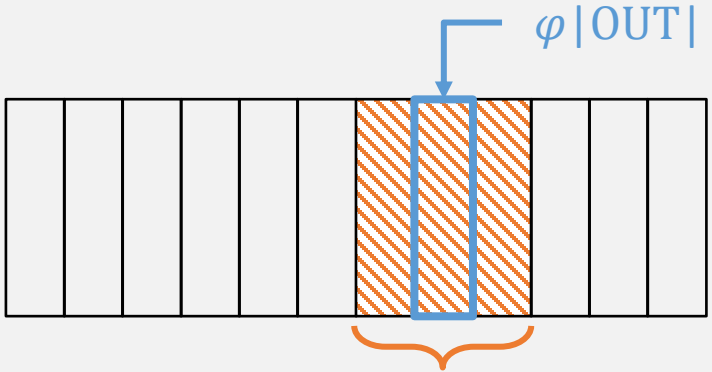
[PODS'23] Tziavelis, Carmeli, Gatterbauer, Kimelfeld, Riedewald. Efficient Computation of Quantiles over Joins. PODS'23 https://doi.org/10.1145/3584372.3588670

Quantile Join Queries: Examples

$R(x,y) \bowtie S(y,z) \bowtie T(z,u)$

- LEX
  - $x \to y \to z \to u$ ✓
  - $z \to x \to u$ ✓
- MIN
  - $\min(x,y,z,u)$ ✓
  - $\max(x,u)$ ✓
- SUM
  - $x+y+z+u$ ✗ (reduction from triangle detection)
  - $x+y+z$ ✓

But can be approximated efficiently!



[PODS'23] Tziavelis, Carmeli, Gatterbauer, Kimelfeld, Riedewald. Efficient Computation of Quantiles over Joins. PODS'23 https://doi.org/10.1145/3584372.3588670

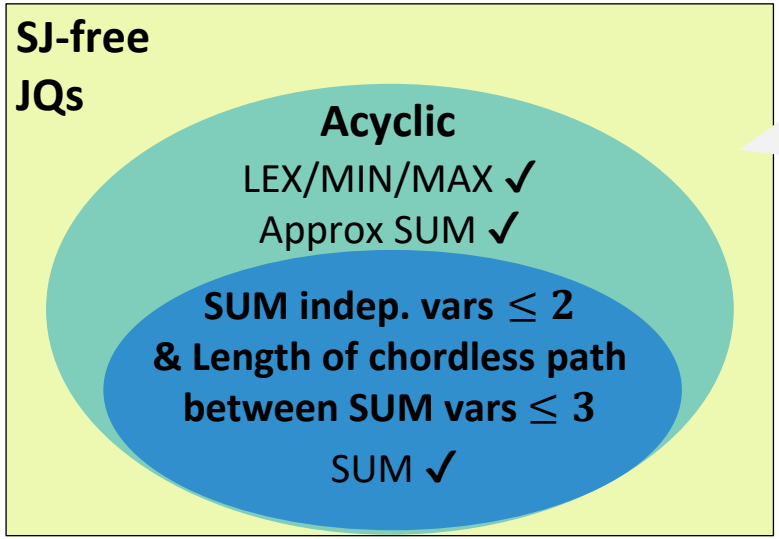
Dichotomy for Quantile Join Queries

Dichotomy: Characterize the tractability of every query and common ranking function

$R(A, B) \bowtie S(B, C) \bowtie T(C, A)$

$R(A, B) \bowtie S(B, C) \bowtie T(C, D)$

$R(A, B) \bowtie S(B, C)$



Lower bounds only for self-join-free queries and conditional on hardness hypotheses

[TODS'23] Carmeli, Tziavelis, Gatterbauer, Kimelfeld, Riedewald. Tractable Orders for Direct Access to Ranked Answers of Conjunctive Queries. TODS'23 https://doi.org/10.1145/3578517
[PODS'23] Tziavelis, Carmeli, Gatterbauer, Kimelfeld, Riedewald. Efficient Computation of Quantiles over Joins. PODS'23 https://doi.org/10.1145/3584372.3588670

Outline

- Motivation & Research Overview
- Highlights of Results
  - Any-k
  - Quantile Queries
- Future Directions

Future Directions

- What other operations can we support over joins without paying the cost of the join?
- What other types of joins can we support?
- Adopting these algorithms in practice:
  1. Implementation for quantile algorithms
  2. Any-k integration with a DBMS
- Stronger guarantees (instance-optimality)
- Algorithms for distributed computation

Thank you!

Website: https://northeastern-datalab.github.io/anyk/