ICDE 2022 tutorial



Toward Responsive DBMS: Optimal Join Algorithms, Enumeration, Factorization, Ranking, and Dynamic Programming

Nikolaos Tziavelis, Wolfgang Gatterbauer, Mirek Riedewald

Northeastern University, Boston

Part 7 : Beyond Equi-Joins, Conclusions

Northeastern University
Khoury College of Computer Sciences

Slides: https://northeastern-datalab.github.io/responsive-dbms-tutorial
DOI: https://doi.org/10.1109/ICDE53745.2022.00299
Data Lab: https://db.khoury.northeastern.edu

DATALAB @Northeastern

Outline tutorial

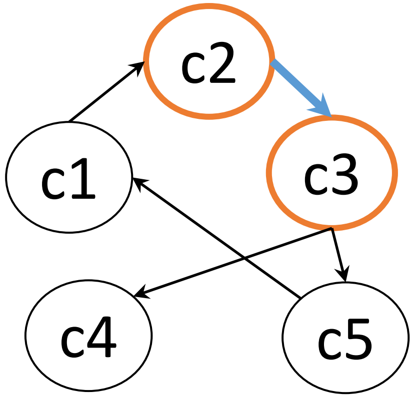
1: Introduction (Nikos) ~40min
2: Tree Decompositions (Mirek) ~20min
3: Acyclic Queries & Enumeration (Wolfgang) ~25min

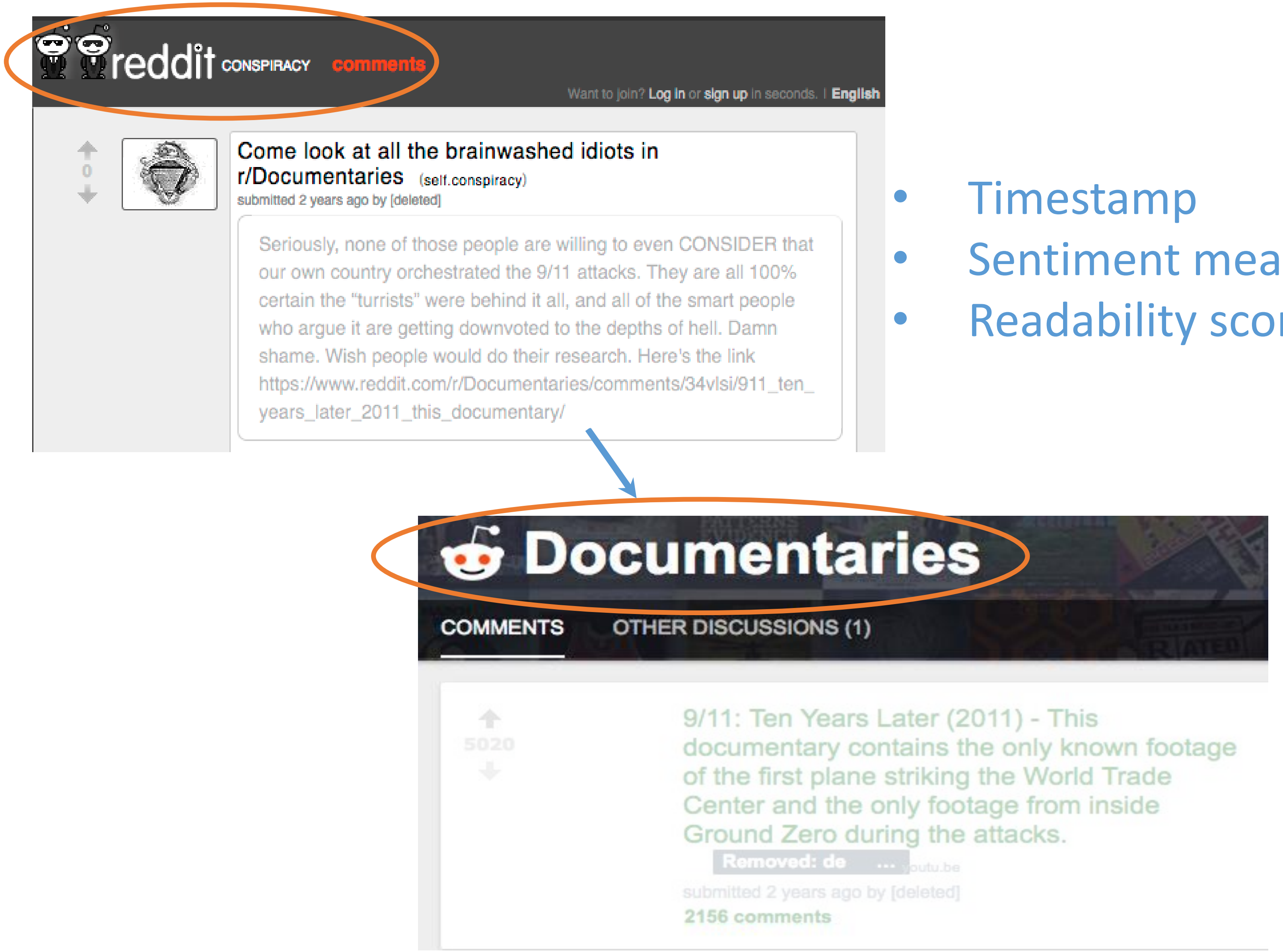
BREAK

4: Factorization (Nikos) ~10min
5: Dynamic Programming & Semirings (Wolfgang) ~20min
6: Any-k or Ranked Enumeration (Nikos) ~35min
7. Decomposition of Comparison Predicates (Mirek) ~10min
8. Conclusion (Mirek) ~10min

Motivating Example

Reddit Network



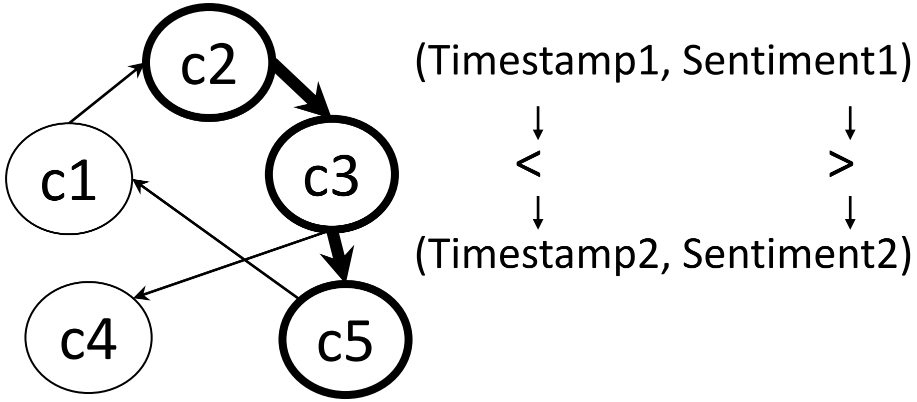


- Timestamp
- Sentiment measure
- Readability score

[K+ 18] Kumar, Hamilton, Leskovec, Jurafsky. Community Interaction and Conflict on the Web. WWW'18. https://doi.org/10.1145/3178876.3186141

Motivating Example

Reddit Network



Join in SQL:

```
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
```

Equality

Inequalities

Ranking

Q: - length-2 paths
- timestamps in increasing order
- sentiment in decreasing order
- top results by sum of readability

Naïve plan:
1. Compute all $\mathrm{O}(n^2)$ join results
2. Sort them

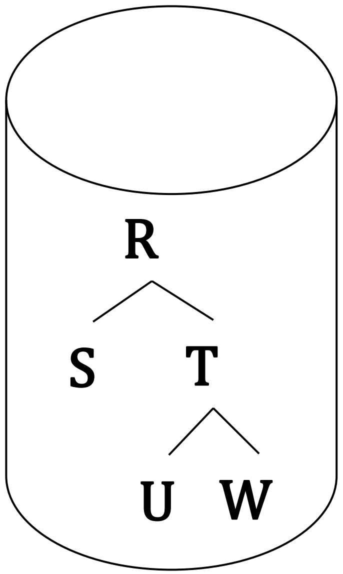
Any-k with factorized representation:
$\mathrm{TT}(k) = \widetilde{\mathbf{O}}(\boldsymbol{n} + \boldsymbol{k})$ (ignoring log factors)

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

Ranked Enumeration for Full Acyclic Join Queries

DB

n tuples



Join

ℓ relations

Lower bound: $\mathrm{TT}(k) = \Omega(n + k)$

Equi-joins [T+20]: $\mathrm{TT}(k) = O(n + k \log k)$

Join-then-rank: $\mathrm{TT}(k) = O\left(n^{\ell} + k \log n\right)$

Any-k applied to DNF of (in)equalities [T+21]:

$$\mathrm{TT}(k) = O(n \operatorname{polylog} n + k \log k)$$

Any-k applied to theta-join [T+21]:

$$\mathrm{TT}(k) = O(n^2 + k \log k)$$

Assumptions

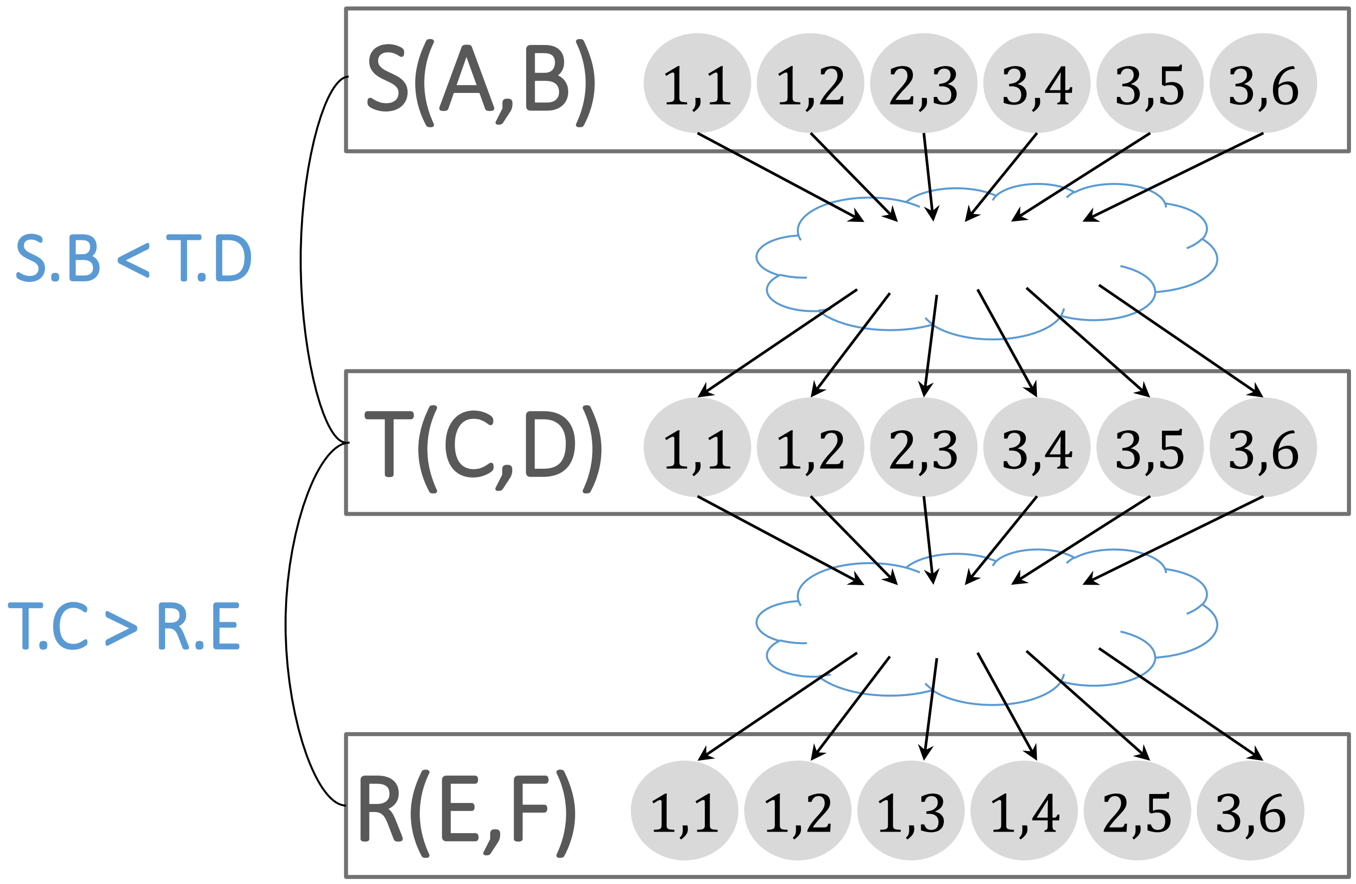
- Data complexity (ℓ, #attributes constant)
- Indexes must be built on-the-fly
- In-memory computation

[T+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

[T+21] Tziavelis, Gatterbauer, Riedewald. Beyond Equi-joins: Ranking, Enumeration and Factorization. PVLDB'21 https://doi.org/10.14778/3476249.3476306

Factorized Representation for (Acyclic) Theta-Joins

Enumeration Graph



relation layers (tuples)

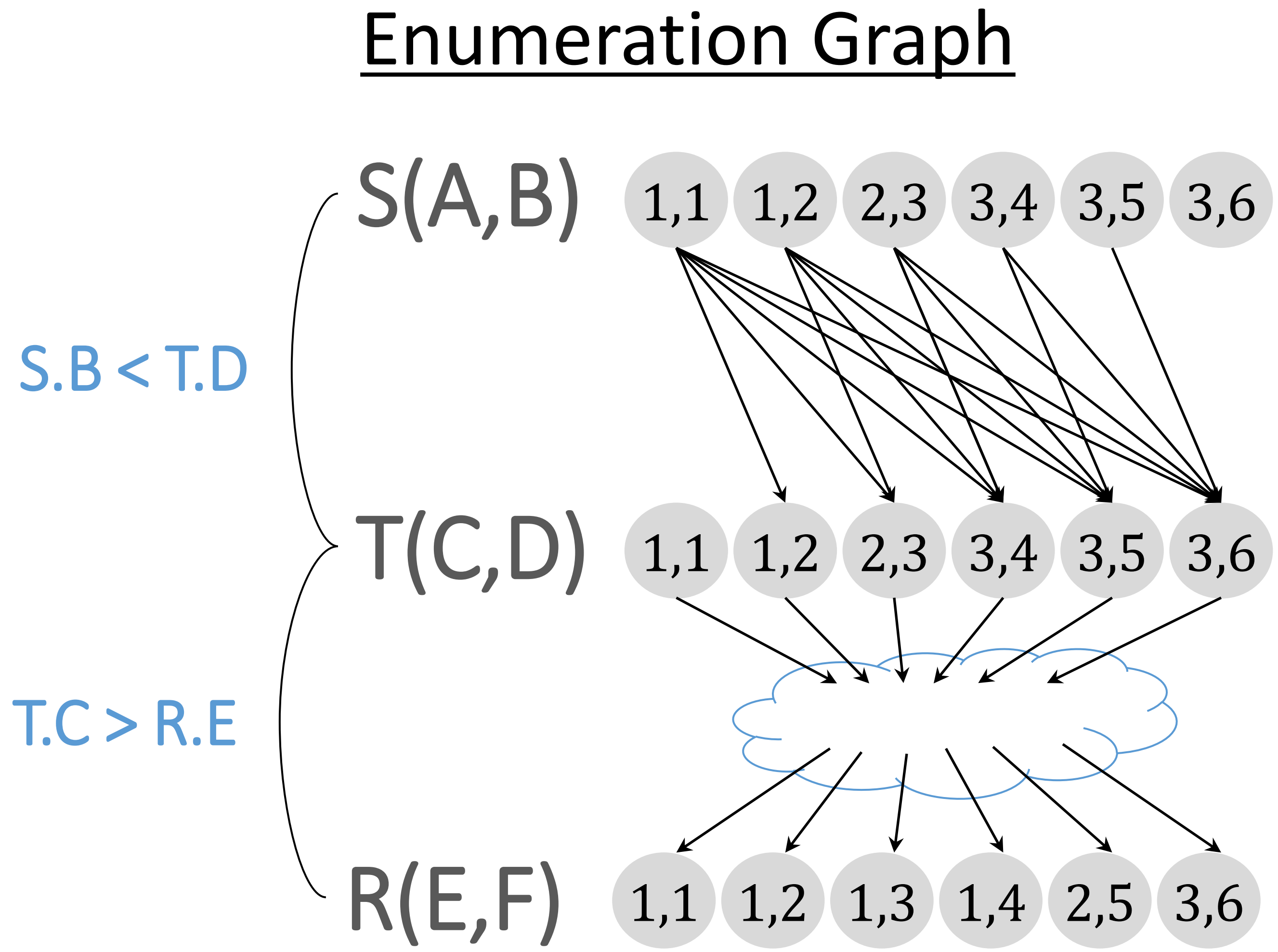
Tuple-Level Factorization Graph (TLFG)

- DAG between 2 relation layers
- Path from S tuple to T tuple

  $\Leftrightarrow$

  Valid join pair
- Ranked enumeration for any TLFG
  - Size affects preprocessing time
  - Depth (longest path) affects delay

Direct TLFGs

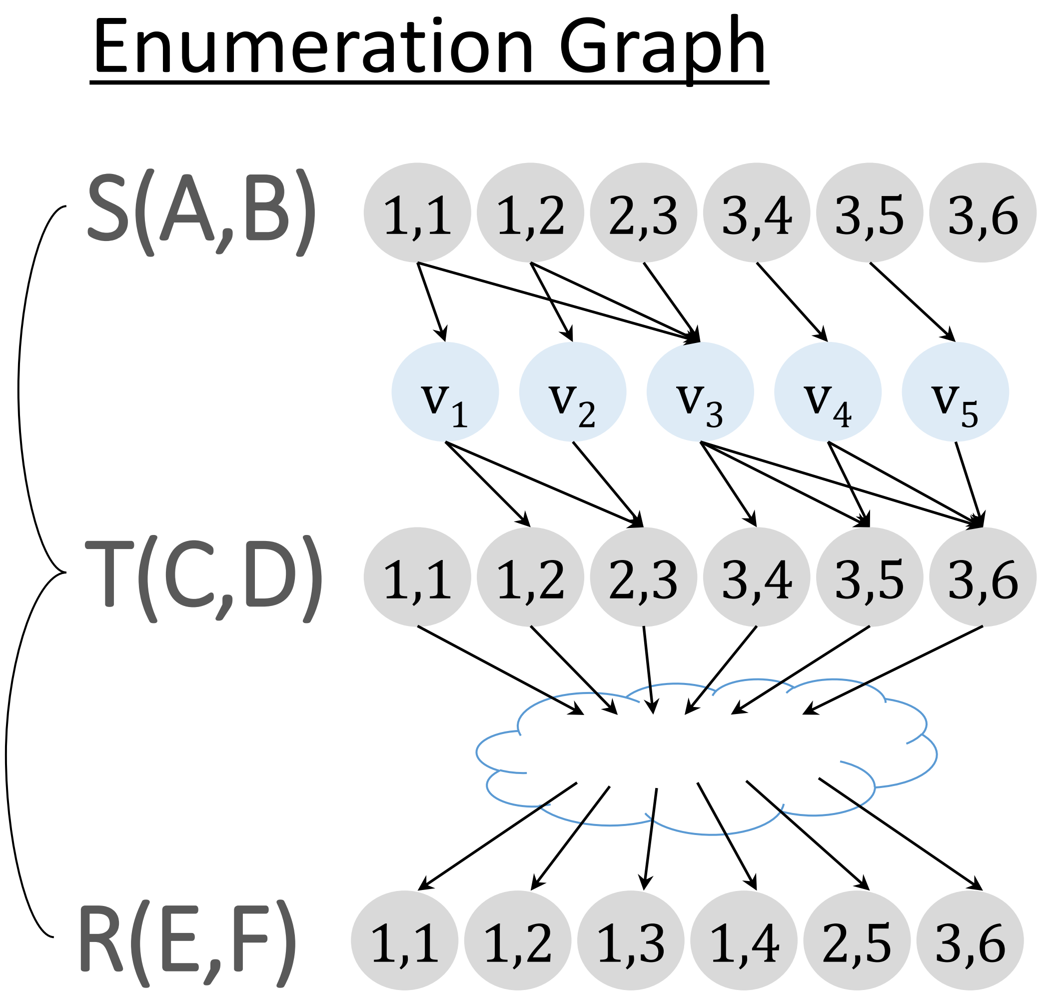


Direct TLFG

- $O(n^2)$ edges
- Depth = 1
- Works for any join condition

$$TT(k) = O(n^2 + k \log k)$$

Binary Partitioning for Inequality Predicate



Binary Partitioning Method

- $O(n \log n)$ size
- Depth = 2
- For 1 inequality predicate

$$\mathrm{TT}(k) = O(n \log n + k \log k)$$ ☺

$\bowtie_{<}$ → $\bowtie_{=}$

$O(n)$-size relations → $O(n \log n)$-size relations

Factorization Generalization and Extensions

- Band predicates ($|S.A - T.B| < \varepsilon$)
- Non-equality predicates ($S.A \neq T.B$)
- Conjunctions/Disjunctions of predicates

- Optimizations for improved memory consumption

Experiments

METHOD	DETAILS
Factorized	• Our method
QuadEqui	• Direct TLFG (materializes auxiliary relations of size $O(n^2)$ to reduce theta-join to equi-join) • Uses ranked enumeration for equi-joins • Time measured **after** materialization
Batch	• Time to rank all results with a Priority Queue • Time for join **not** measured
PSQL	• Prebuilt indexes • Limit clause
System X	• Commercial DBMS • In-memory optimized

Thus, idealized compared to real implementation!

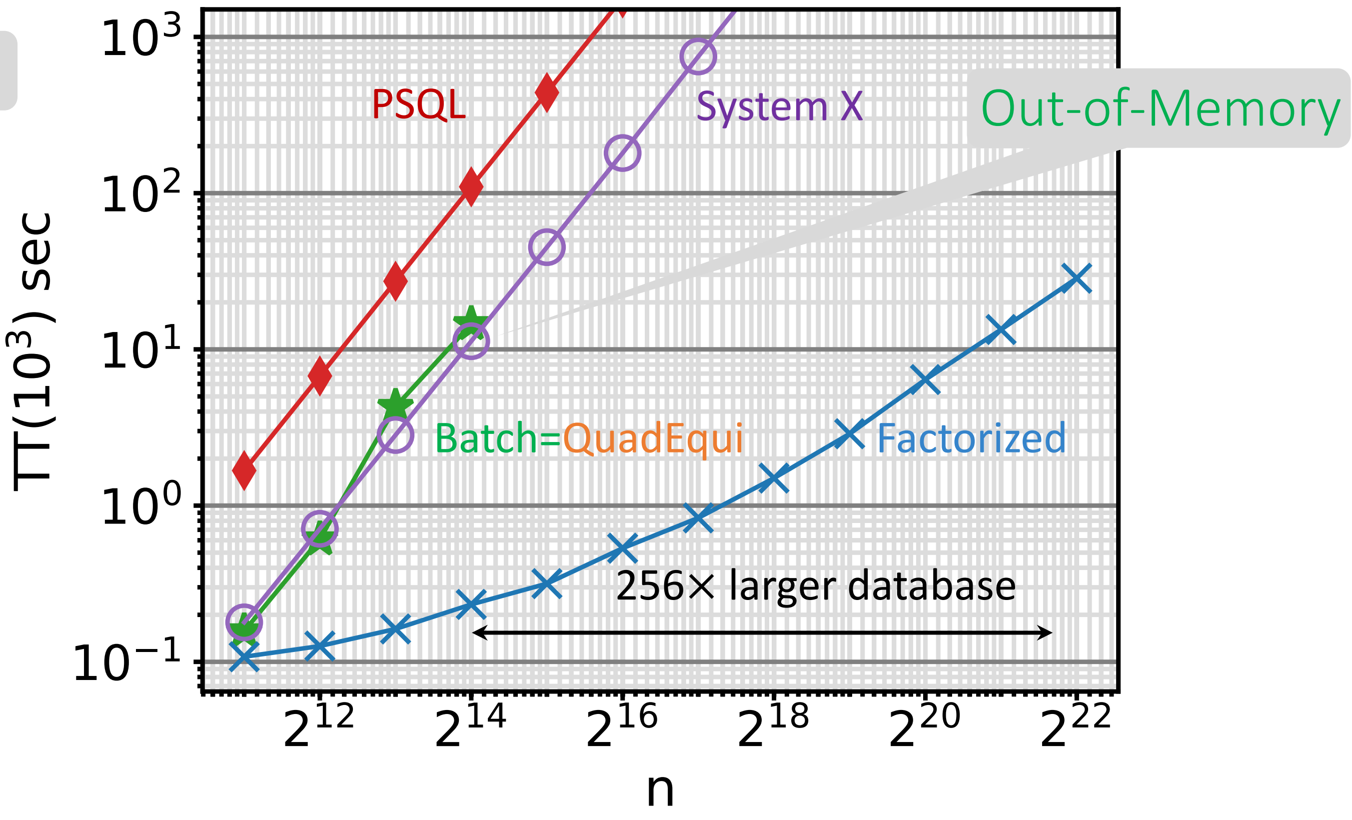
DBMSs (PSQL, System X)

Exp1: Synthetic Data

$S_i(A_i, A_{i+1}, W)$

- Tuples values drawn randomly from integer domain
- Binary join with one inequality predicate

Top-1000




```
select *, S1.W + S2.W as weight
from S1, S2
where S1.A2 < S2.A3
order by weight asc
```

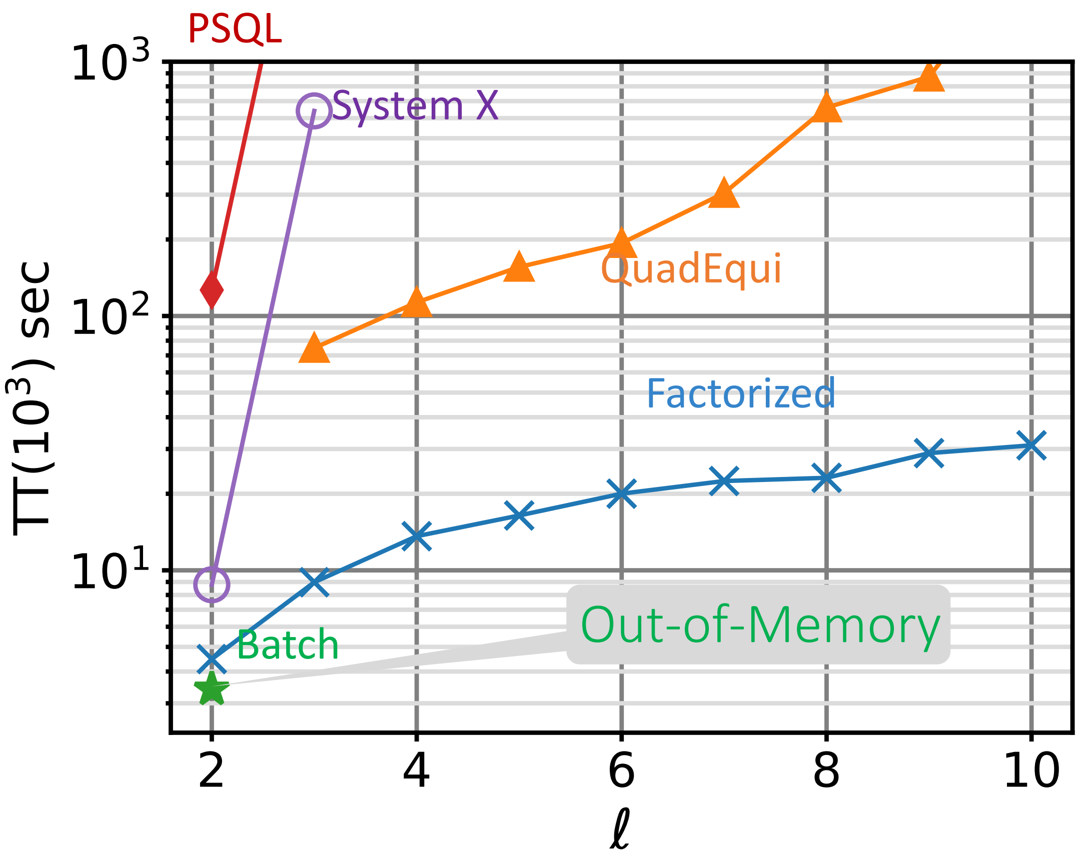
Other methods face memory problems as n increases.

n = relation size

Exp2: Paths on Reddit

Q: - length-ℓ paths ~286k edges

- timestamps in increasing order
- sentiment in decreasing order
- top results by sum of readability



select *
from Reddit R1, Reddit R2
where R2.Source = R1.Target
AND R2.Timestamp > R1.Timestamp
AND R2.Sentiment < R1.Sentiment
order by weight desc

Our method is robust to different query sizes and complicated join conditions.

ℓ = #relations

Summary

- DBMSs typically struggle with complex join predicates like inequalities.
- The any-k factorized algorithm can return the top join results (e.g., top-1000) in time comparable to sorting the input

  Even for $O(n^{\ell})$ join results!

- For (full) acyclic queries with DNFs of equalities and inequalities:

$$\mathrm{TT}(k) = O(n \operatorname{polylog} n + k \log k)$$

- This factorization also applies to the other query types (e.g., unranked enumeration, aggregation) with analogous time complexity guarantees

Website: https://northeastern-datalab.github.io/anyk/
Code available online!

Conclusions

Queries Discussed

- Focused on conjunctive queries
  - SELECT-FROM-WHERE with only AND-connected constraints
- Also explored extensions using union and aggregation
  - DNF of inequality join conditions
  - ORDER BY
- Covers a large spectrum of real-world SQL queries for
  - Unranked enumeration
  - Ranked enumeration
  - Direct access to specific output position

Complexity of Join Computation

- A join query of ℓ relations of size n has output size $r = O(n^{\ell})$
  - AGM bound gives tighter upper bound specific to the given query, e.g., $O(n^{1.5})$ instead of $O(n^{3})$ for the 3-cycle query
- Lower bound to compute the join: $\Omega(n + r)$
- Matching it requires avoiding intermediate results greater than r
  - Yannakakis algorithm achieves $O(n + r)$ for full acyclic queries
    - Bottom-up and top-down sweeps of semi-join reduction on join tree
    - Then bottom-up join of relation leftovers
  - Generally not achievable for cyclic queries
    - Best possible so far is $O(n^{d} + r)$, where $d \geq 1$ is a width parameter
    - Approach uses tree decompositions and worst-cast optimal join algorithms

Getting Faster Responses (Full Acyclic Queries)

- …even when join output size is n^{ℓ}
- Enumeration: output join answers (unordered)
  - $\mathrm{TT}(k) = \mathrm{O}(n + k)$
- Ranked enumeration: output in order of "monotonic" ranking function
  - $\mathrm{TT}(k) = \mathrm{O}(n + k \log k)$
- Direct access (for fragment of acyclic queries and limited class of monotonic ranking functions)
  - $\mathrm{TT}(k) = \mathrm{O}(n \operatorname{polylog} n + k \operatorname{polylog} k)$

Extended Results

- Selection: easy
  - Apply in pre-processing
- Projection: can be hard!
  - Intuition: it creates “duplicates” that cause large delays until next returned answer
  - In SQL only an issue for SELECT DISTINCT queries
  - $\mathrm{TT}(k) = \mathrm{O}(n + k)$ achievable for (and only for) free-connex acyclic queries
- Joins with conditions other than equality: ranked enumeration
  - DNF of inequalities: $\mathrm{TT}(k) = \mathrm{O}(n \operatorname{polylog} n + k \log k)$
  - Theta-join: $\mathrm{TT}(k) = \mathrm{O}(n^2 + k \log k)$

The "Secret Sauce"

- Semi-join reductions (Yannakakis algorithm)
- Dynamic programming
  - Graph of related sub-problems to efficiently compute top-1 answer
- Generalization to compute top-2, top-3,… efficiently
- Factorization
  - Represent a binary join of size $\mathrm{O}(n^2)$ in space $\mathrm{O}(n)$ (equi-join) or $\mathrm{O}(n \text{ polylog } n)$ (DNF of inequality conditions)
- Semi-rings
  - Enabler of factorization: $ac + ad + bc + bd = (a + b)(c + d)$

What is Next?

- Support for more general types of predicates
  - E.g., $R.A + S.B < T.C$
- Efficient incremental maintenance under updates
  - Some results for enumeration and aggregates (e.g., triangle count)
- ML over factorized data

- General question: What can we compute quickly/efficiently without having to materialize the entire join result?

Thank you!

Slides: https://northeastern-datalab.github.io/responsive-dbms-tutorial
DOI: https://doi.org/10.1109/ICDE53745.2022.00299
Data Lab: https://db.khoury.northeastern.edu

This work was supported in part by the Office of Naval Research (Grant#: N00014-21-C-1111), the National Institutes of Health (NIH) under award number R01 NS091421, and by the National Science Foundation (NSF) under award numbers CAREER IIS-1762268and IIS-1956096. Any opinions, findings, and conclusions or recommendations expressed in this presentation are those of the authors and do not necessarily reflect the views of the funding agencies.