Asymptotically Better Query Optimization Using Indexed Algebra

Philipp Fent
Technical University of Munich
fent@in.tum.de

Motivation

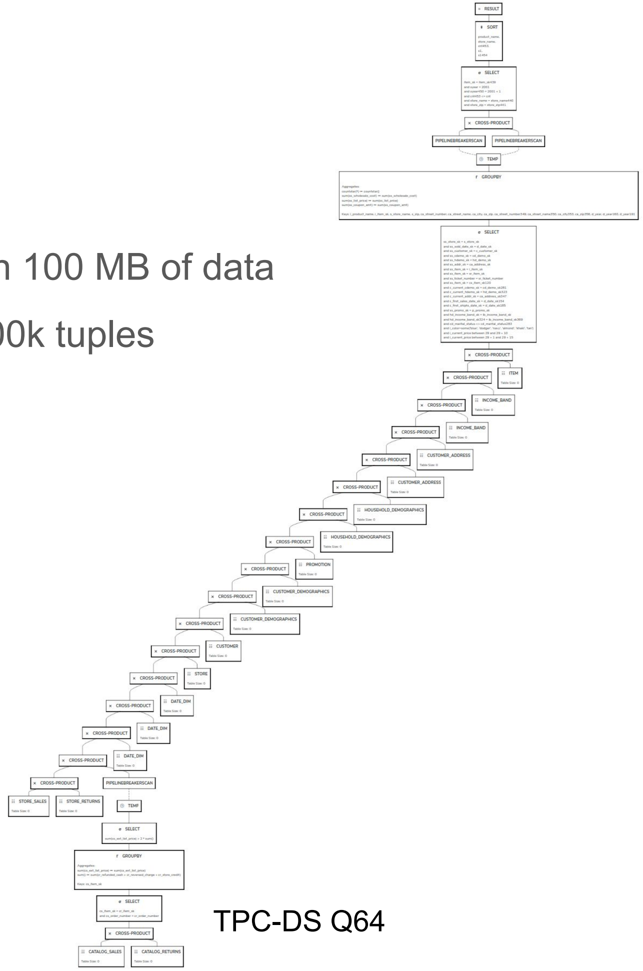
- Complex queries on small workloads
  - BigQuery: 90% of queries processed less than 100 MB of data
  - Tableau Public: 90% of workbooks are less than 100k tuples

Motivation

- Complex queries on small workloads
  - BigQuery: 90% of queries processed less than 100 MB of data
  - Tableau Public: 90% of workbooks are less than 100k tuples
- TPC-H
  - Scale 1: 0.8 ms optimization, 20 ms execution
  - Scale 0.01: 0.8 ms optimization, 0.2 ms execution

Motivation

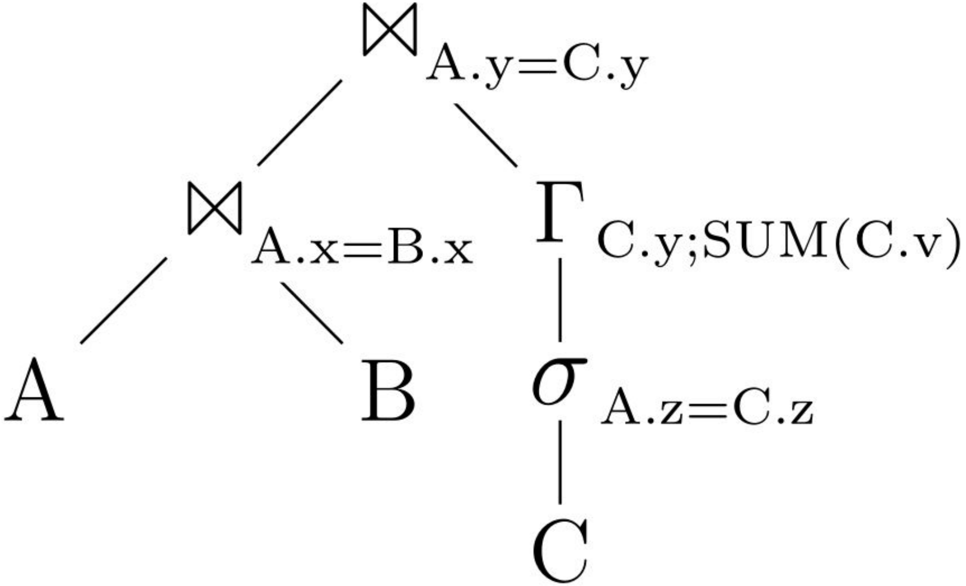
- Complex queries on small workloads
  - BigQuery: 90% of queries processed less than 100 MB of data
  - Tableau Public: 90% of workbooks are less than 100k tuples
- TPC-H
  - Scale 1: 0.8 ms optimization, 20 ms execution
  - Scale 0.01: 0.8 ms optimization, 0.2 ms execution
- And: Optimization time scales super-linear with query complexity



TPC-DS Q64

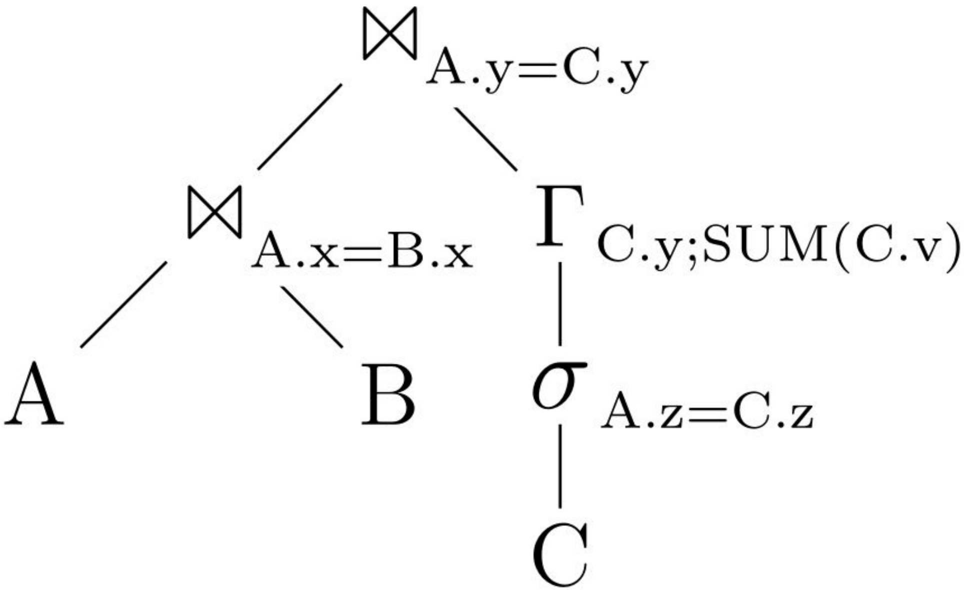
Algebra

- Relational algebra trees
  - Operators
  - Expressions
  - Columns / IUs



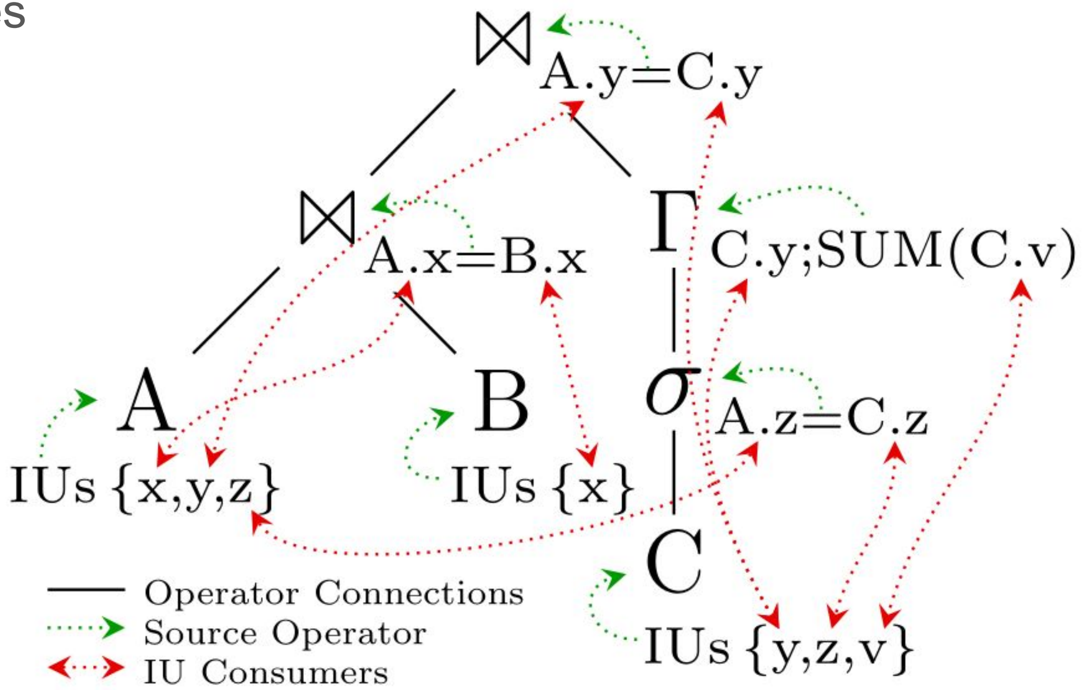
Algebra

- Relational algebra trees
  - Operators
  - Expressions
  - Columns / IUs
- Analyze data-flow for optimization
  - Which path?
  - Modifications?
  - Materialized?



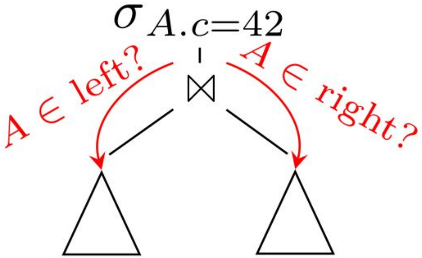
Algebra

- Relational algebra trees
  - Operators
  - Expressions
  - Columns / IUs
- Analyze data-flow for optimization
  - Which path?
  - Modifications?
  - Materialized?
- Interconnected



Optimization

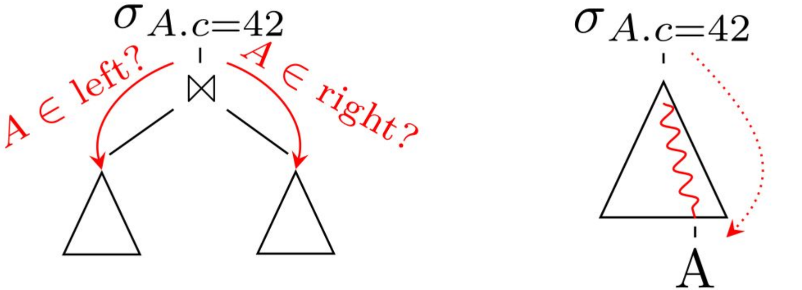
- Reason about the algebra to derive optimization possibilities
- Top-down, operator at a time
  - Needs $O(n^2)$ column sets



(a) Operator-centric

Optimization

- Reason about the algebra to derive optimization possibilities
- Top-down, operator at a time
  - Needs $O(n^2)$ column sets
- Path-centric
  - Still $O(n)$ length
  - With indexing: $O(\log n)$



(a) Operator-centric (b) Path-centric

Indexing Algebra

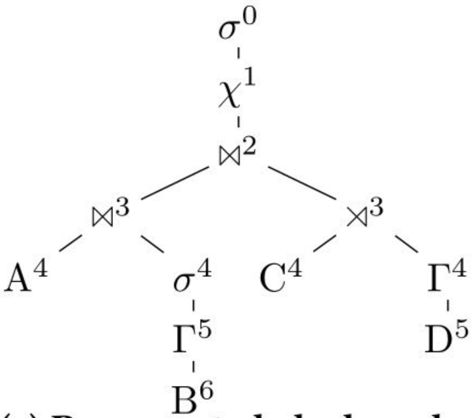
- Index paths through the algebra

➡ Faster path traversal

Indexing Algebra

- Index paths through the algebra

➡ Faster path traversal



(a) Represented algebra plan

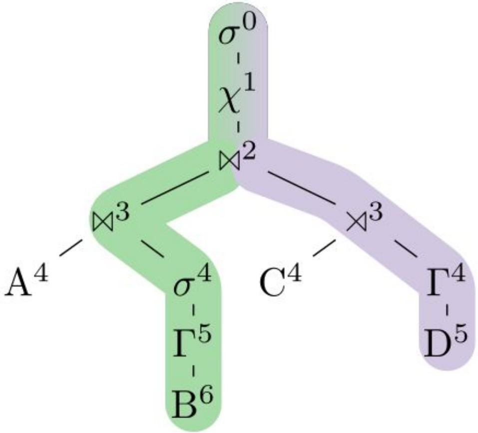
Indexing Algebra

- Index paths through the algebra

  ➡ Faster path traversal

- Binary search trees

  on path depth



(a) Represented algebra plan

Indexing Algebra

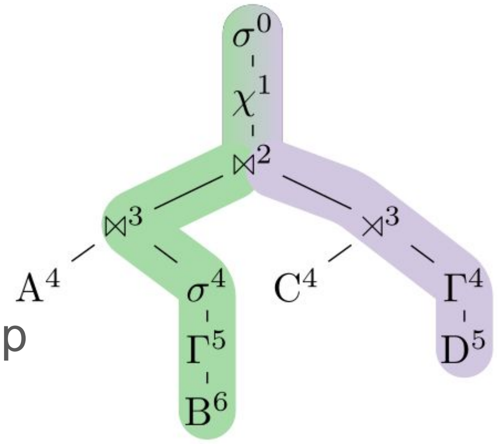
- Index paths through the algebra

  ➡ Faster path traversal

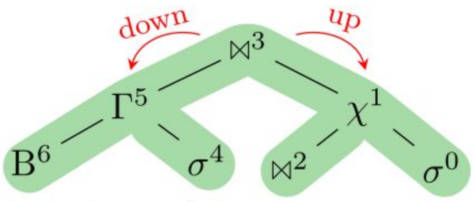
- Binary search trees

  on path depth

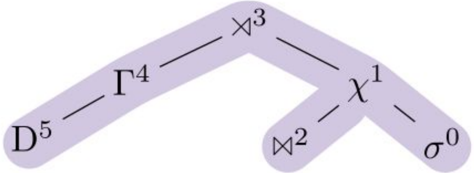
- Paths from root might overlap



(a) Represented algebra plan



(b) Balanced binary index of the path from B^6 to the root



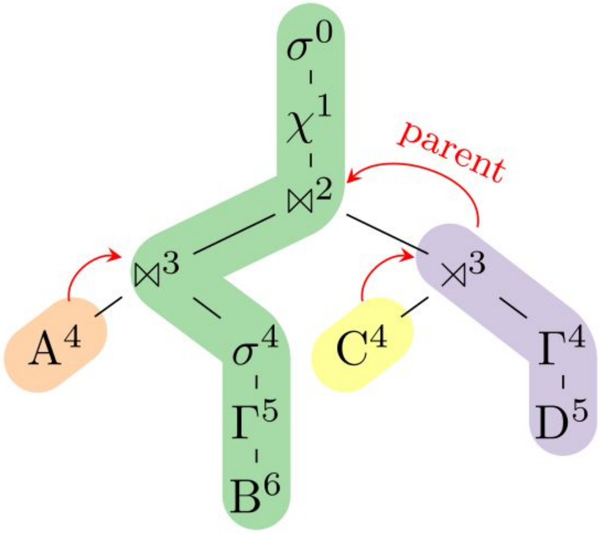
(c) Index from D^5 to the root

Indexing Algebra

- Index paths through the algebra

  ➡ Faster path traversal

- Binary search trees on path depth
- Paths from root might overlap
- **Link/cut trees** support that efficiently



Indexing Algebra

- Index paths through the algebra

  ➡ Faster path traversal

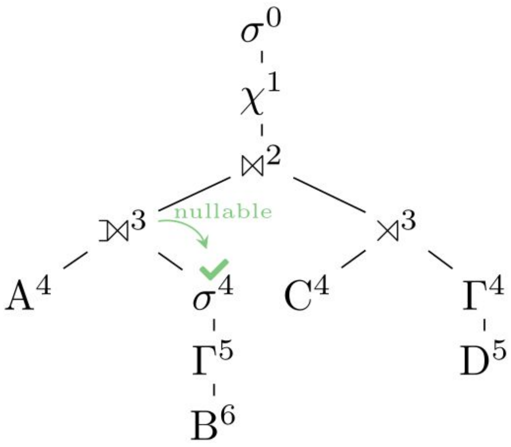
- Binary search trees on path depth
- Paths from root might overlap
- **Link/cut trees** support that efficiently

Rel. Algebra	Transformation	Traversal
w/o index	$\mathcal{O}(1)$	$\mathcal{O}(n)$
static index	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$
path labeling	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Indexed Algebra	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

Path-centric optimization

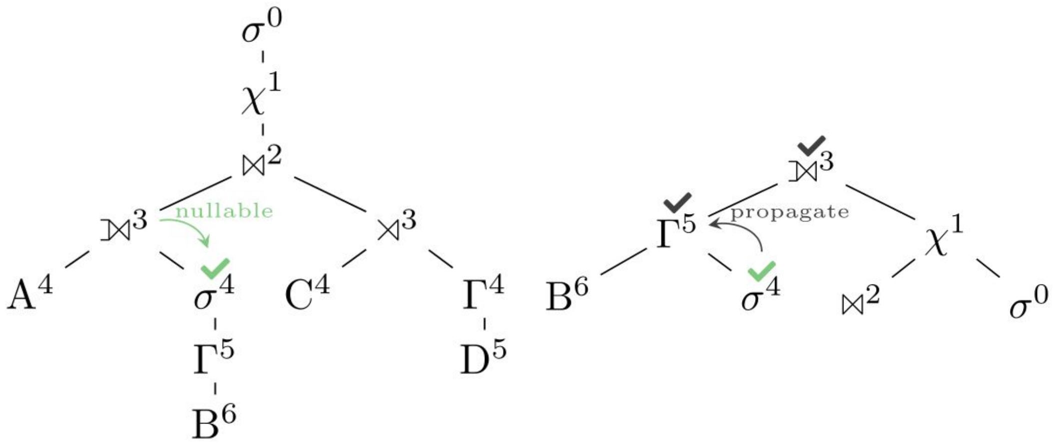
Path-centric optimization

- Detect outer-joins on path
  - Mark children nullable



Path-centric optimization

- Detect outer-joins on path
  - Mark children nullable
  - Propagate marker in index

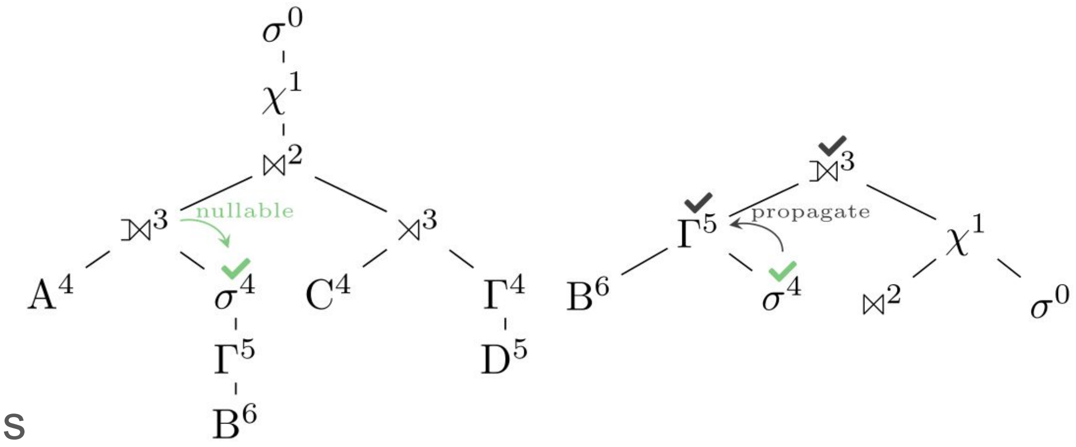


Path-centric optimization

- Detect outer-joins on path
  - Mark children nullable
  - Propagate marker in index
- Check null bit in index
  - Cut away subtrees for partial path queries
- Predicate pushdown

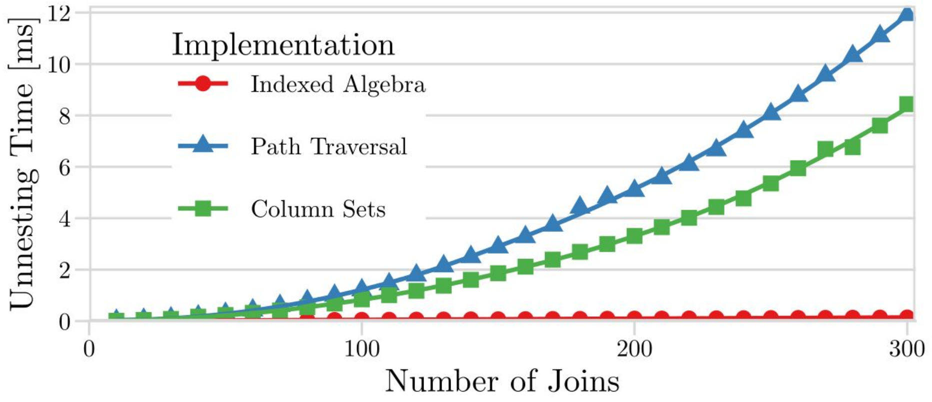
  Constant propagation upwards

  ➡ O(log n)



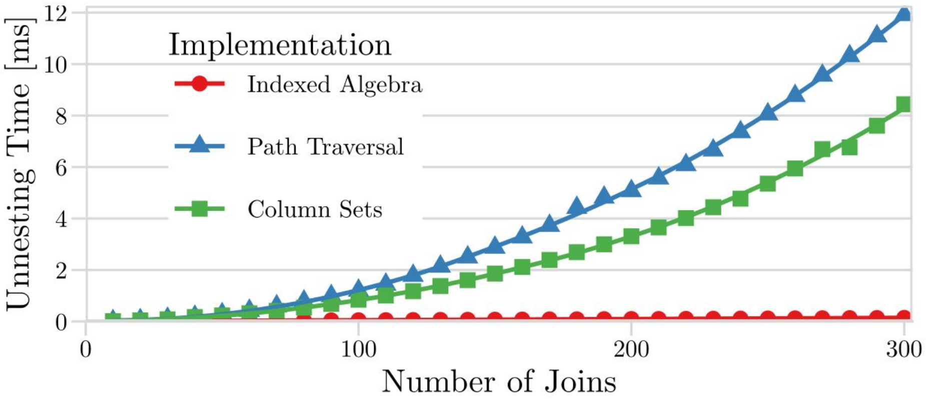
Performance

- Clearly better asymptotics



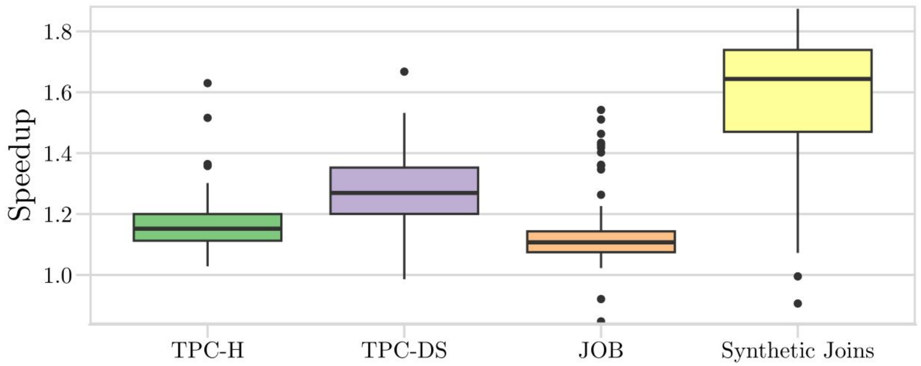
Performance

- Clearly better asymptotics
  - Query unnesting is read-only
  - Path-labels would be even better but need $O(n^2)$ construction



Performance

- Significant overall improvements
- 10 - 30% faster optimization
- 8% better *end-to-end* latency in Tableau Public



Asymptotically Better Query Optimization Using Indexed Algebra

- Asymptotically better query optimization
- Allows elegant and concise formulations for *data flow* questions
- But needs effort to reengineer the optimizer

Philipp Fent
Technical University of Munich
fent@in.tum.de