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
Algebra

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

\[
\begin{align*}
A \bowtie & \ A.y = C.y \\
A \bowtie & \ A.x = B.x \\
B \quad & \quad \Gamma C.y; \text{SUM}(C.v) \\
\sigma & \ A.z = C.z
\end{align*}
\]
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
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

<table>
<thead>
<tr>
<th>Rel. Algebra</th>
<th>Transformation</th>
<th>Traversal</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o index</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>static index</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>path labeling</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Indexed Algebra</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>
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
  \( \Rightarrow O(\log n) \)
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