A Practical Approach to Groupjoin and Nested Aggregates

Philipp Fent, Thomas Neumann

Technische Universität München

VLDB 2021
Groupjoin
Combines Join and Group-By
Groupjoin
Combines Join and Group-By
Groupjoin

The MD-join: An Operator for Complex OLAP

Damianos Chatziantoniou
Panacea Software Inc.
damianos@panacea.com

Theodore Johnson
Database Research Center
AT&T Labs - Research
johnsonnt@research.att.com

Michael Akinde
Dept. of Computer Science
Aalborg University
strategy@cs.auc.dk

Samuel Kim
Dept. of Computer Science
Stevens Institute of Tech.
skim@cs.stevens-tech.edu

Main Memory Implementations for Binary Grouping

Norman Man and Guido Moerkotte
University of Mannheim
D-68
68131 Mannheim, Germany
{norman.man}@cs.informatik.uni-mannheim.de

Accelerating Queries with Group-By and Join by Groupjoin

Guido Moerkotte
Universität Mannheim
Mannheim, Germany
moerkotte@informatik.uni-mannheim.de

Thomas Neumann
Technische Universität München
Munich, Germany
neumann@in.tum.de

Getting Swole: Generating Access-Aware Code with Predicate Pullups

Andrew Croft
Brown University
amcroft@cs.brown.edu

Alex Galiatsatos
Brown University
agaliats@cs.brown.edu

Tim Kraska
MIT CSAIL
kraska@csail.mit.edu
Groupjoin

The MD-join : An Operator for Complex OLAP

Damianos Chatziantoniou
Panakea Software Inc.
damianos@panakea.com

Theodore Johnson
Database Research Center
AT&T Labs - Research
johnson@research.att.com

Michael Akinde
Dept. of Computer Science*
Aalborg University
strategy@cs.auc.dk

Samuel Kim
Dept. of Computer Science
Stevens Institute of Tech.
skim@cs.stevens-tech.edu

Main Memory Implementations for Binary Grouping

Norman May and Guido Moerkotte
University of Mannheim
D-68
68131 Mannheim, Germany

Accelerating Queries with Group-By and Join by Groupjoin

Guido Moerkotte
Universität Mannheim
Mannheim, Germany
moerkotte@informatik.uni-mannheim.de

Thomas Neumann
Technische Universität München
Munich, Germany
neumann@in.tum.de

Getting Swole: Generating Access-Aware Code with Predicate Pullups

Andrew Croft
Brown University
acroft@cs.brown.edu

Alex Galabov
Brown University
agabov@cs.brown.edu

Tim Kraska
MIT CSAIL
kraska@csail.mit.edu

Combined Hashtable

Key Aggregates

Key Aggregates

Key Aggregates
Groupjoin

The MD-join: An Operator for Complex OLAP

Michael Akinde
Dept. of Computer Science
Aalborg University
strategy@cs.auc.dk

Theodore Johnson
Database Research Center
AT&T Labs - Research
johnsonot@research.att.com

Damianos Chatziantoniou
Panana Software Inc.
damianos@panana.com

Samuel Kim
Dept. of Computer Science
Stevens Institute of Tech.
skim@cs.stevens-tech.edu

Main Memory Implementations for Binary Grouping

Norman Mao and Guido Moerkotte
University of Mannheim
D-68131 Mannheim, Germany
{norman-mao,guido}@informatik.uni-mannheim.de

Accelerating Queries with Group-By and Join by Groupjoin

Guido Moerkotte
Universität Mannheim
Mannheim, Germany
moerkotte@informatik.uni-mannheim.de

Thomas Neumann
Technische Universität München
Munich, Germany
neumann@in.tum.de

Getting Swole: Generating Access-Aware Code with Predicate Pullups

Andrew Croft
Brown University
croft@cs.brown.edu

Alex Galkovsky
Brown University
galex@cs.brown.edu

Tim Kraska
MIT CSAIL
timk@csail.mit.edu

Combined Hashtable

Key Aggregates

Key Aggregates

Key Aggregates
Groupjoin
Avoiding Contention with Eager Right Execution

- Re-uses thread-local $\Gamma$ logic
- Works well for $\sigma_S \approx 1$
Groupjoin
Memoizing Groupjoin

Global Hashtable

<table>
<thead>
<tr>
<th>Key</th>
<th>T₀</th>
<th>Aggregates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Thread-local Hashtables

- T₁
  \[ \Gamma_1 = \{\} \]
- T₂
  \[ \Gamma_2 = \{\} \]
- T₃
  \[ \Gamma_3 = \{\} \]
Groupjoin
Memoizing Groupjoin

Global Hashtable

<table>
<thead>
<tr>
<th>Key</th>
<th>T₀</th>
<th>Aggregates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Thread-local Hashtables

\[
\Gamma_1 = \{\} \\
\Gamma_2 = \{\} \\
\Gamma_3 = \{\}
\]
Groupjoin

Memoizing Groupjoin

Global Hashtable

- Key | $T_2$ | Aggregates
- Key | $T_1$ | Aggregates
- Key | $T_3$ | Aggregates

Thread-local Hashtables

- $T_1$
  - $\Gamma_1 = \{\}$
- $T_2$
  - $\Gamma_2 = \{\}$
- $T_3$
  - $\Gamma_3 = \{\}$
Groupjoin
Memoizing Groupjoin

Global Hashtable

<table>
<thead>
<tr>
<th>Key</th>
<th>T2</th>
<th>Aggregates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>T1</th>
<th>Aggregates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>T3</th>
<th>Aggregates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Gamma_3 = \{ \} \]

\[ \Gamma_1 = \{ 3 : \gamma_1 \} \]

\[ \Gamma_2 = \{ 3 : \gamma_2 \} \]

Thread-local Hashtables

- \( T_1 \)
- \( T_2 \)
- \( T_3 \)
Groupjoin

Comparison

Method
- eager
- memoizing
- separate

Used Memory [MB]

Progress

0%  25%  50%  75%  100%
Nested Aggregates

- Decorrelation supported by Groupjoin
- HAVING predicates are hard to estimate
Nested Aggregates

- Decorrelation supported by Groupjoin
- HAVING predicates are hard to estimate

\[ \Gamma \]
\[ \text{lineitem} \star \text{customer} \]
\[ \sigma \Gamma \]
\[ \text{lineitem} \]

a) Selective $\sigma$-Predicate

\[ \Gamma \]
\[ \text{lineitem} \star \text{customer} \]
\[ \sigma \Gamma \]
\[ \text{lineitem} \]

b) Unselective $\sigma$-Predicate

\[ \Gamma \]
\[ \text{lineitem} \star \text{customer} \]
\[ \sigma \Gamma \]
\[ \text{lineitem} \]
Estimating Aggregates

- Numerical columns $\sim \mathcal{N}(\mu, \sigma^2)$
- Cheap and generalizes nicely . . . but inherently symmetric
Estimating Aggregates

- Numerical columns $\sim \mathcal{N}(\mu, \sigma^2)$
- Cheap and generalizes nicely ... but inherently symmetric

IMDb movie rating

TPC-H customer balance

Observed Data
Calculated skew-normal fit
Estimating Aggregates

Using a skew-normal distribution

- $\Pr[x \leq c] = \Phi_{sn}(c)$
- Also approximates $(x \circ y)$, AVG/SUM/MIN/MAX
Estimating Aggregates

Using a skew-normal distribution

- \( \Pr[x \leq c] = \Phi_{sn}(c) \)
- Also approximates \((x \circ y), \text{AVG/SUM/MIN/MAX}\)

Input Data

<table>
<thead>
<tr>
<th>SUM(X)</th>
<th>AVG(X)</th>
<th>COUNT(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

Density

- MIN(X)
- MAX(X)

SUM(X)

Density

Base Column X  Group Size  Simulated  Calculated Estimate
Practical Groupjoins and Nested Aggregates

Overall impact on TPC-H and TPC-DS

- Big improvements with cost model advised Groupjoin
- Slightly better query plans with HAVING estimates

Live queries: https://umbra-db.com/interface