REVISITING REUSE IN (MAIN-MEMORY) DATABASES

KAYHAN DURSUN*, CARSTEN BINNIG, UGUR CETINTEMEL, TIM KRASKA
REUSE IN THE REAL WORLD … MIGHT HELP!
... BUT MIGHT ALSO HURT!
REUSE IN DATABASES MIGHT HURT …

1. **Reason about reuse** potential of intermediate results

2. **Add additional materialization** operations to query plan

3. Analyze if subsequent query can **reuse cached intermediates**

Unclear if extra costs pay off in future
Main idea: Reuse internal structures
• Keep internal data structures
• Reuse for subsequent queries

Savings are two-fold
• **No** additional **materialization costs**
• **No** need to **re-create internal structures**

More robust towards different reuse-potentials

This talk: reuse hash tables for joins + aggregations
CASE I: EXACT REUSE

**Initial Query**

```
SELECT c.age, c.sex, SUM(o.total)
FROM C, O, L
WHERE <join-conditions>
AND l.shipdate >= 2015-02-01
GROUP BY c.age, c.sex
```

**Reuse Query**

```
SELECT c.age, c.sex, SUM(o.total)
FROM C, O, L
WHERE <join-conditions>
AND l.shipdate >= 2015-01-01
GROUP BY c.age, c.sex
```

**Exact-Reuse HT2**
CASE II: PARTIAL REUSE

**Initial Query**
```
SELECT c.age, c.sex, SUM(o.total)
FROM C, O, L
WHERE <join-conditions>
AND l.shipdate ≥ 2015-02-01
GROUP BY c.age, c.sex
```

**Reuse Query**
```
SELECT c.age, c.sex, SUM(o.total)
FROM C, O, L
WHERE <join-conditions>
AND l.shipdate ≥ 2015-01-01
GROUP BY c.age, c.sex
```

**Partial-Reuse HT3:**
Add missing tuples
CASE III: SUBSUMING REUSE

Initial Query

SELECT c.age, c.sex, SUM(o.total) 
FROM C, O, L 
WHERE <join-conditions> 
AND l.shipdate ≥ 2015-02-01 
GROUP BY c.age, c.sex 

Reuse Query

SELECT c.age, c.sex, SUM(o.total) 
FROM C, O, L 
WHERE <join-conditions> 
AND l.shipdate ≥ 2015-02-01 
AND o.orderdate < 2015-01-01 
GROUP BY c.age, c.sex 

Subsuming-Reuse HT2: Filter false positives
SELECT o.total
FROM L JOIN O
WHERE o.orderdate ≥ 2015-01-01

Which HT to reuse for the join?

Rewrite query and execute it

Executor

DBMS Runtime

Hash Table
Manager (HTM)

Hash Tables
Lineage +
Statistics
Locks
Usage

Garbage
Collection

Hash Table
Cache

1. get_candidates
(lineage)

2. use
(hash_tables)

3. execute
(reuse_plan)

4. free(hash_tables)
COST-MODEL FOR REUSE IN JOINS

Cost of a Hash-⨝: \( c_{\text{build}}(HT) + c_{\text{probe}}(HT) \)

How is that different for a RHJ?

\( c_{RHJ} = c_{\text{resize}}^R(HT) + c_{\text{build}}^R(HT) + c_{\text{probe}}^R(HT) \)

1. add only missing tuples
2. higher per-tuple cost (due to add. tuples)
3. post-filter false positives (i.e. \( a=0 \))

Reuse Query:

\[ R(a,b) \land S(c,d) \land b = d \land 1 \leq a \leq 2 \]

<table>
<thead>
<tr>
<th>Table R</th>
<th>HT: ( 0 \leq a \leq 1 )</th>
<th>Table S</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b )</td>
<td>( b )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
JOIN: REUSE OR NOT REUSE?

Given: one fixed size HT for reuse

- Always Share
- Never Share
- Cost Model

Run Time (sec)

contribution-ratio

0% missing 100% missing
AGGREGATION: REUSE OR NOT REUSE?

Given: one fixed size HT for reuse

- Always Share
- Never Share
- Cost Model

0% missing

100% missing
ADDITIONAL OPTIMIZATIONS

Store additional attributes in HTs
• Enables post-filtering false positives
• Only if extension does not exceed cache line

Aggregate Optimizations
• Always store COUNT and SUM instead of AVG
• Add additional aggregate functions

Speculative Optimizations
• Join re-ordering to produce “better” hash-tables
• Store aggregate hash table on finer granularity
RESULTS: REUSE EFFICIENCY

Workloads: 64 queries with different reuse potentials

- **Low**: 1% of the cached data is reused on average
- **Medium**: 10% of the cached data is reused on average
- **High**: 50% of the cached data is reused on average
RESULTS: EFFECT OF MULTI-THREADING

Workload: Varying the degree of parallelism

- **Medium-reuse:** 64 queries
- **Thread:** 2-20 threads
RESULTS: ACCURACY OF COST MODEL

Workload:

- 5-way join: Customer, Orders, Lineitem, Supplier, Part
- For each sub-plan: enumerated all HTs that qualify
SUMMARY AND OUTLOOK

Main idea:
• Reuse internal data structures of operators
• More robust towards different workloads

Other topics not covered in this talk
• Reuse-aware hash aggregate
• Top-down plan enumeration algorithm
• Query batch interface

Full paper at SIGMOD 2017