Finding the Pitfalls in Query Performance

M.L. Kersten
P. Koutsourakis
Y. Zhang

CWI, MonetDB Solutions

EU H2020 project ACTiCLOUD
Which system is relatively better?
The Solution, TPC-H?

Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
Q16
Q17
Q18
Q19
Q20
Q21
Q22

MonetDB Mar-18
Postgres 10.6
MonetDB Jul-17
Postgres 11
SQLite
TPC-H is a collection of ‘random points’?
TPC-H may miss clarifying queries

Query tells a better story for MonetDB
TPC-H may miss discriminative queries

Postgres is relatively better than MonetDB
TPC-H may miss discriminative queries

MonetDB Degrades on A query
TPC-H may miss discriminative queries

SQLite is superior
The Challenge

• Which system is better on a benchmark?

• What queries perform relatively better on system A than system B?
The Challenge

Find the discriminative queries

SQL scalpel

MonetDB Mar-18
Postgres 10.6
MonetDB Jul-17
Postgres 11
SQLite
Assumptions

• The database schema and data distribution are given
• A collection of business inspired queries are available
• No direct access to the Database/DBMS/Platform
The Solution, TPC-H as a start

MonetDB Mar-18
Postgres 10.6
MonetDB Jul-17
Postgres 11
SQLite
The Solution, TPC-H as a start

-- Query 6

```
select
  sum(l_extendedprice * l_discount) as revenue
from
  lineitem
where
  l_shipdate >= date '1994-01-01' and
  l_shipdate < date '1994-01-01' + interval '1' year and
  l_discount between .06 - 0.01 and .06 + 0.01 and
  l_quantity < 24
```
The Solution, TPC-H as a start

-- Query 6

select 
  sum(l_extendedprice * l_discount) as revenue 
from 
  lineitem 
where 
  l_shipdate >= date '1994-01-01'
The Solution, TPC-H as a start

-- Query 6

select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_discount between .06 - 0.01 and .06 + 0.01 and
    l_quantity < 24
The Solution, TPC-H as a start

-- Query 6

select sum(l_extendedprice * l_discount) as revenue
from lineitem
where l_shipdate < date '1994-01-01' + interval '1' year
The Solution, TPC-H as a start

-- Query 6

```
select 
    sum(l_extendedprice * l_discount) as revenue
from 
    lineitem
where
    l_shipdate >= date '1994-01-01'
    and
    l_shipdate < date '1994-01-01' + interval '1' year
    and
    l_discount between .06 - 0.01 and .06 + 0.01
    and
    l_quantity < 24
```
SQLscalpel compiles the query into a grammar

-- Query 6

select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1994-01-01' and
    l_shipdate < date '1994-01-01' + interval '1' year and
    l_discount between .06 - 0.01 and .06 + 0.01 and
    l_quantity < 24

tpch_q6:
SELECT ${projection} FROM ${table}
WHERE ${pred} ${predlist}*

projection:
    sum(l_extendedprice * l_discount) as revenue

table:
    lineitem

pred:
   l_shipdate >= date '1994-01-01'
   l_shipdate < date '1994-01-01' + interval '1' year
   l_discount between 0.06 - 0.01 and 0.06 + 0.01
   l_quantity < 24

predlist:
   AND ${pred}
SQLscalpel enumerates templates

SELECT @{$projection} FROM @{$table} WHERE @{$pred}

SELECT @{$projection} FROM @{$table} WHERE @{$pred} AND @{$pred}

SELECT @{$projection} FROM @{$table} WHERE @{$pred} AND @{$pred} AND @{$pred}

SELECT @{$projection} FROM @{$table} WHERE @{$pred} AND @{$pred} AND @{$pred} AND @{$pred}

\[
\text{tpch\_q6:} \\
\text{SELECT @{$projection} FROM @{$table}} \\
\text{WHERE @{$pred} @{$predlist}\ast}
\]

projection:
sum(l_extendedprice * l_discount) as revenue

table:
lineitem

pred:
l_shipdate >= date '1994-01-01'
l_shipdate < date '1994-01-01' + interval '1' year
l_discount between 0.06 - 0.01 and 0.06 + 0.01
l_quantity < 24

predlist:
\text{AND @{$pred}}
<table>
<thead>
<tr>
<th>tag</th>
<th>templates</th>
<th>space</th>
<th>tag</th>
<th>templates</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>40</td>
<td>9207</td>
<td>Q12</td>
<td>8484</td>
<td>162918</td>
</tr>
<tr>
<td>Q2</td>
<td>58160</td>
<td>6354837405</td>
<td>Q13</td>
<td>16</td>
<td>81</td>
</tr>
<tr>
<td>Q3</td>
<td>240</td>
<td>29295</td>
<td>Q14</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>Q4</td>
<td>28</td>
<td>81</td>
<td>Q15</td>
<td>40</td>
<td>372</td>
</tr>
<tr>
<td>Q5</td>
<td>108</td>
<td>96579</td>
<td>Q16</td>
<td>608</td>
<td>25515</td>
</tr>
<tr>
<td>Q6</td>
<td>4</td>
<td>15</td>
<td>Q17</td>
<td>26</td>
<td>81</td>
</tr>
<tr>
<td>Q7</td>
<td>&gt;100K</td>
<td>–</td>
<td>Q18</td>
<td>576</td>
<td>43659</td>
</tr>
<tr>
<td>Q8</td>
<td>480</td>
<td>5478165</td>
<td>Q19</td>
<td>&gt;100K</td>
<td>–</td>
</tr>
<tr>
<td>Q9</td>
<td>1512</td>
<td>3528441</td>
<td>Q20</td>
<td>320</td>
<td>3339.0</td>
</tr>
<tr>
<td>Q10</td>
<td>384</td>
<td>722925</td>
<td>Q21</td>
<td>18464</td>
<td>4255065</td>
</tr>
<tr>
<td>Q11</td>
<td>162</td>
<td>7203</td>
<td>Q22</td>
<td>156</td>
<td>777</td>
</tr>
</tbody>
</table>

**Figure 6: TPC-H query space**
Maintain a query pool

Pick promising candidates

Keep a workflow database

Support analysis
Maintain a query pool
Pick promising candidates
Keep a workflow database
Support analysis
SQLscalpel architecture

Maintain a query pool
Pick promising candidates
Keep a workflow database
Support analysis

Strategies:
• Baseline
• Random
• Alter a lexical term
• Expand a query
• Prune a query
• Ditch manually
SQLscalpel architecture

Maintain a query pool

Pick promising candidates

Keep a workflow database

Support analysis

Strategies:
• FCFS
• Manual steering
• Simulated annealing
• Biased terms
The Scalpel grammar

A Scalpel grammar is a concise description of a collection of test cases. It is described using a grammar composed of rules identified by an identifier following by a colon. Each rule is followed by a series of alternative text snippets to construct the test case. Each snippet in the grammar is only used once in a single test case. The grammar rules can be embedded as references $\{\text{name}\}$ or $\{\text{name}\}$ in the snippets.

```
query:
  select $\{l\_select\}$ $\{select\_expr\}*$ $\{l\_from\}$ $\{l\_where\}$ group by $\{l\_group\_by\_term\}$ $\{group\_by\_expr\}$ order by $\{l\_order\_by\_term\}$

  $\{l\_group\_by\_term\}$
  select_expr:

  $\{l\_select\}$
  order_by_expr:

  $\{l\_from\}$ from lineitem

  $\{l\_where\}$

  where _shipdate <= date '1998-12-01' - interval '90' day (3)

  $\{l\_group\_by\_term\}$

  _returnflag

  _linestatus

  _select_term:

  _returnflag

  _linestatus

  sum ( _quantity ) as sum_qty

  sum ( _extendedprice ) as sum_base_price

  sum ( _extendedprice * (1 - _discount) ) as sum_disc_price

  sum ( _extendedprice * (1 - _discount) * (1 + _tax) ) as sum_charge

  avg ( _quantity ) as avg_qty

  avg ( _extendedprice ) as avg_price

  avg ( _discount ) as avg_disc

  count ( * ) as count_order

  $\{l\_order\_by\_term\}$

  _returnflag

  _linestatus
```
Scalpel experiment lineage

The exploration steps through the query space are visualized as a map. It simply shows the performance of the queries in milliseconds and the lineage relationships of the queries with each other.

Highlight results for just a few terms.
Scalpel experiment lineage

The exploration steps through the query space are visualized as a map. It simply shows the performance of the queries in milliseconds and the ops.

Highlight results for just a few terms.

Response time lineage sf1/MonetDB_Mar18/localhost

- Random
- Alter
- Expand
- Prune
- Error

Tag: 103 (82)
Times: 180.166 ms
Ratio: 0.978

Dropped: sum (L_quantity) as sum_qty
Query: SELECT L_returnflag, sum (L_extendedprice * (1 - L_discount)) as sum_disc_price, avg (L_extendedprice) as avg_price, avg (L_discount) as avg_disc, L_linenumber, sum (L_quantity) as sum_qty FROM lineitem WHERE L_shipdate <= date '1998-12-01' - interval '90' day (3) GROUP BY L_returnflag, L_linenumber ORDER BY L_returnflag
Scalpel terms analysis

The lexical terms impact on a query is visualized below. It is calculated by taking two queries which differ only in a single term, then statistically calculate the mean and stddev, and also show the outliers.

The measured performance can drop below 0 due to caching effects of tables over a sequence of similar queries.

Pre-filter the result table

```
sum (l_quantity) as sum_qt
sum (l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge
sum (l_extendedprice) as sum_base_price
avg (l_extendedprice) as avg_price
sum (l_extendedprice * (1 - l_discount)) as sum_disc_price
l_returnflag
l_linestatus
count (*) as count_order
avg (l_discount) as avg_disc
avg (l_quantity) as avg_qt
```
Scalpel scatter analysis

Compare all queries executed against two database/dbms/platform combinations.

Pre-filter the result table  Set dimensions
### SQLscalpel pitfalls

**tpch_q6:**

```
SELECT ${projection} FROM ${table} WHERE ${pred} ${predlist}*
```

**projection:**

```
sum(l_extendedprice * l_discount) as revenue
```

**table:**

```
lineitem
lineitem, regions
```

**pred:**

```
l_shipdate >= date '1994-01-01'
l_shipdate < date '1994-01-01' + interval '1' year
l_discount between 0.06 - 0.01 and 0.06 + 0.01
l_quantity < 24
```

**predlist:**

```
AND ${pred}
```
- SQLscalpel prototype is up and running:
  - Full-stack infrastructure
  - Drivers for MonetDB, MonetDB Lite, SQLite, Postgres

- Functional enhancements planned:
  - Multi-party public/private projects
  - Built-in forum to share the story
Divergence

Q1: SELECT count(*) FROM nation WHERE nation.n_name='BRAZIL'
Q2: SELECT count(*) FROM nation WHERE nation.n_name='BRAZIL' AND nation.n_regionkey=1

\[
\begin{align*}
&\frac{T_A(Q_2)}{T_B(Q_2)} \\
&\frac{T_A(Q_1)}{T_B(Q_1)}
\end{align*}
\]

\[
\begin{align*}
&\frac{T_A(Q_2)}{T_B(Q_2)} \cdot \frac{T_B(Q_1)}{T_A(Q_1)}
\end{align*}
\]
Q1: SELECT count(*) FROM nation WHERE nation.n_name='BRAZIL'
Q2: SELECT count(*) FROM nation WHERE nation.n_name='BRAZIL' AND nation.n_regionkey=1

\[
\frac{T_A(Q_2)}{T_B(Q_2)} < 1 \quad \text{System A is better than B on Q}_2 \text{ against Q}_1
\]

\[
\frac{T_B(Q_1)}{T_A(Q_1)} = 1
\]

\[
> 1 \quad \text{System B is better than A on Q}_2 \text{ against Q}_1
\]