Online Estimation For Subset-Based SQL Queries

Chris Jermaine, Alin Dobra, Abhi Pol, Shantanu Joshi
University of Florida
Computer and Information Sciences and Engineering Department
Online, Approximate Computation

- DB performance for large-scale, ad-hoc analytic processing completely stinks
  - TPC-H: Can spend $millions on warehouse
  - And still wait minutes, hours, even days

- Can we use randomized, *online* algorithms?
  - More time $\rightarrow$ better answer
  - Bounds: “With 98% conf., the answer is 45.5 ± 3.5”

- “Classic” work: HHW SIGMOD 1997

No hists, wavelets, sketches...
What is the state of the art?

• Existing work: selection, join, grouping
• But no work applicable to “subset-based” queries:

```sql
SELECT SUM (EMP.SALARY) 
FROM EMP 
WHERE EMP.START.DATE < 'Jan 1, 1999' AND 
 NOT EXISTS (SELECT * 
 FROM SALES 
 WHERE EMP.ID = SALES.ID AND 
 SALES.DATE > 'Jan 1, 2002')
```

“What is the total salary for employees who started before 1999 and have not had a sale after 2001?”
What is the state of the art?

- Existing work: selection, join, grouping
- But no work applicable to “subset-based” queries:

```
SELECT SUM (EMP.SALARY)
FROM EMP
WHERE EMP.START.DATE < 'Jan 1, 1999' AND
NOT EXISTS (SELECT *
            FROM SALES
            WHERE EMP.ID = SALES.ID AND
            SALES.DATE > 'Jan 1, 2002')
```

outer aggregate query linked to a correlated subquery via a “subset” operator
What is the state of the art?

- Existing work: selection, join, grouping
- But no work applicable to “subset-based” queries:

```sql
SELECT SUM (EMP.SALARY)
FROM EMP
WHERE EMP.START.DATE < 'Jan 1, 1999' AND
    NOT EXISTS (SELECT *
                FROM SALES
                WHERE EMP.ID = SALES.ID AND
                SALES.DATE > 'Jan 1, 2002')
```

Covers SQL (NOT) IN, (NOT) EXISTS, EXCEPT, DISTINCT, and others... we will focus on NOT EXISTS
Can We Handle Such Queries Now?

• Not too hard using HHW’s original framework
• Sample repeatedly from EMP
• Rely on an index on SALES
The Problem

• But could be terrible solution:
  • Need around three seeks per tuple in EMP
  • Only 1000 tuples per minute

• Might we forgo the index on SALES?
Joins Are Relatively Easy

- Ripple Join (HH SIGMOD 1999): Compute “mini-join” and then just scale to unbias

\[
\begin{array}{cccccc}
\text{SALES} \\
\text{(Jan)} & \text{(Tim)} & \text{(Sue)} & \text{(Joe)} & \text{(Jon)} & \text{(Nat)} \\
\hline
\text{Tom} & 12 & \text{Sam} & 5 & \text{Jon} & 21 & \text{Joe} & 8 & \text{Tim} & 4 \\
\end{array}
\]

\[
\text{EMP} \\
\hline
(\text{Jan}, 3) & (\text{Sam}, 5) & (\text{Jon}, 21) & (\text{Joe}, 8) & (\text{Tim}, 4) \\
\end{array}
\]

\[
\begin{align*}
\text{SELECT} & \quad \text{SUM EMP.b} \\
\text{FROM} & \quad \text{EMP, SALES} \\
\text{WHERE} & \quad \text{EMP.a = SALES.a}
\end{align*}
\]

\[
\frac{6}{1} \times \frac{6}{1} \times (0) = 0
\]

Must be clustered w. true statistical randomness
Joins Are Relatively Easy

- Ripple Join (HH SIGMOD 1999): Compute “mini-join” and then just scale to unbiased

<table>
<thead>
<tr>
<th>EMP</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Jan)</td>
<td>(Tom, 12)</td>
</tr>
<tr>
<td>(Tim)</td>
<td>(Jan, 3)</td>
</tr>
<tr>
<td>(Sue)</td>
<td>(Sam, 5)</td>
</tr>
<tr>
<td>(Joe)</td>
<td>(Jon, 21)</td>
</tr>
<tr>
<td>(Jon)</td>
<td>(Joe, 8)</td>
</tr>
<tr>
<td>(Nat)</td>
<td>(Tim, 4)</td>
</tr>
</tbody>
</table>

\[
\frac{6}{2} \times \frac{6}{2} \times (3) = 27
\]

```sql
SELECT SUM EMP.b
FROM EMP, SALES
WHERE EMP.a = SALES.a
```
Joins Are Relatively Easy

- Ripple Join (HH SIGMOD 1999): Compute “mini-join” and then just scale to unbias

\[ \frac{6}{4} \times \frac{6}{4} \times (3 + 12) = 33.75 \]

```
SELECT SUM EMP.b
FROM EMP, SALES
WHERE EMP.a = SALES.a
```
Joins Are Relatively Easy

- Ripple Join (HH SIGMOD 1999): Compute “mini-join” and then just scale to unbiased

\[
\text{SELECT SUM EMP.b} \\
\text{FROM EMP, SALES} \\
\text{WHERE EMP.a = SALES.a}
\]

\[
\frac{6}{5} \times \frac{6}{5} \times (3 + 12 + 10) = 36
\]
Joins Are Relatively Easy

- Ripple Join (HH SIGMOD 1999): Compute “mini-join” and then just scale to unbias

```
SALES
<table>
<thead>
<tr>
<th>(Jan)</th>
<th>(Tim)</th>
<th>(Tom)</th>
<th>(Joe)</th>
<th>(Jon)</th>
<th>(Sue)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EMP
<table>
<thead>
<tr>
<th>(Tom, 12)</th>
<th>(Jan, 3)</th>
<th>(Sam, 5)</th>
<th>(Jon, 10)</th>
<th>(Joe, 8)</th>
<th>(Tim, 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SELECT SUM EMP.b FROM EMP, SALES WHERE EMP.a = SALES.a

Final answer

\[
\frac{6}{6} \times \frac{6}{6} \times (3 + 12 + 10) = \frac{25}{6}
\]
Subset-Based Not So Easy

- Can scale up to deal with sampling from outer relation
- But how do you scale for inner? Additional tuples decrease the running total (assuming \texttt{NOT EXISTS} query)
- Hence bias is a fact of life
Subset-Based Not So Easy

- Obvious extension to ripple join likely to over-estimate total

\[
\frac{6}{1} \times (12) = 60
\]

```
SELECT SUM EMP.b
FROM EMP WHERE NOT EXISTS (SELECT * FROM SALES
WHERE EMP.a = SALES.a)
```

![Diagram of SALES and EMP sets]
• Obvious extension to ripple join likely to over-estimate total

\[
\frac{6}{2} \times (12 + 3) = 45
\]

SELECT SUM EMP.b
FROM EMP WHERE NOT EXISTS
EXISTS (SELECT * FROM SALES
WHERE EMP.a = SALES.a)
Subset-Based Not So Easy

- Obvious extension to ripple join likely to over-estimate total

\[
\frac{6}{4} \times (12 + 3 + 5) = 30
\]

SELECT SUM EMP.b
FROM EMP
WHERE NOT EXISTS (SELECT * FROM SALES
WHERE EMP.a = SALES.a)
Subset-Based Not So Easy

• Obvious extension to ripple join likely to over-estimate total

\[
\frac{6}{5} \times (12 + 5) = 20.4
\]

SELECT SUM EMP.b
FROM EMP WHERE NOT EXISTS
EXISTS (SELECT * FROM SALES
WHERE EMP.a = SALES.a)
Subset-Based Not So Easy

- Obvious extension to ripple join likely to over-estimate total

Not an accident that first estimate was 6 times too large

\[
\frac{6}{6} \times (5 + 4) = 9
\]

**SELECT SUM EMP.b FROM EMP WHERE NOT EXISTS (SELECT * FROM SALES WHERE EMP.a = SALES.a)\]**
Save The Concurrent Sample?

• Typical method: quantify bias and correct
• In this case, the bias is:

\[
\sum_{e \in \text{EMP}} f(e)(\varphi(e) - \text{one}(e, \text{SALES}))
\]

• Problem: \(\varphi(e)\) hard to obtain
  • Need exact count of \(e\) in \(\text{SALES}\)
Solution: Combine W. Indexed Sol’n

• We need an estimate for

\[ \sum_{e \in EMP} f(e)(\varphi(e) - one(e, SALES)) \]

• So, repeatedly:
  • Sample e from EMP
  • Use index to look for matches in SALES
Solution: Combine W. Indexed Sol’n

• We need an estimate for

\[ \sum_{e \in EMP} f(e)(\varphi(e) - one(e, SALES)) \]

• So, repeatedly:
  • Sample e from EMP
  • Use index to look for matches in SALES
  • Compute this for e
Combined Solution

• This gives us an estimator:

\[ N + U \]

From concurrent sample \hspace{1cm} From index
Combined Solution (cont)

While (true)

1) Read some more blocks from EMP
   Read some more blocks from SALES
   Look for matches and update \( N \)
2) Read some random tuples from EMP
   Use index to look for matches and update \( U \)
3) Output \( N + U \) as current estimate
Why Do We Like This?

- Unbiased (unlike concurrent sample)
- Paper has derivation of $\sigma^2(N + U)$
- Less vulnerable to high seek costs than plain indexed solution. Why?
  - Bias of $N$ is usually less than query answer
  - So terms that are added when you use index to get $U$ must be smaller than if you used same info to estimate answer directly
  - So variance of $U$ tends to be smaller than if you used same info to estimate answer directly
  - So less indexed samples needed
Can Do Even Better

• Can we reduce the variance even more?
• Almost all variance of $N + U$ is $U$’s fault
  • So paper describes a modified est. $wN + U(w)$
  • Now have a family of unbiased estimators
  • Find a “sweet spot” that minimizes variance
• 2nd issue: how to choose three sampling speeds? (we use grad. descent)
Results

• Depending on data properties
  • Conf. interval width of combined solution might be almost identical to simple indexed solution
  • Maybe even a bit larger
  • Or it might be 100 times narrower
  • See the paper
  • Particularly good when distribution is skewed
Future Work

• Can we eliminate index entirely?
• We now have an unbiased (very complicated) version of $U$ that does not use an index
• But it has poor variance
• Forgo unbiasedness and minimize MSE?