Accelerating Queries with Group-By and Join by Groupjoin

Guido Moerkotte and Thomas Neumann

Sept. 2011

A Week at Max Data

Setting

- Max Data: startup company implementing main memory DBMS
- Joe: head of query processing team
- Max: founder and boss (Max is a true manager)

Monday

After 6 month of intensive tuning of the QEE,
Joe has to report performance numbers for TPC-H to his boss.

Joe's Slide

Query	time(ms)
21	500
13	278
9	192
3	104
10	74
5	68
16	49
20	37
17	34
...	...
total	1932

Max' Favorite Management Tool

Max Not bad.
However, I've the feeling we can do better.

Max turns to the Performance Tuning Dart Board (PTDB).


50%
30%
20%
10%
Der Abzug

The Management Decision

The dart says

30% off

Max: Make it 30% faster.

Still Monday

This night Joe goes to his favorite Bar.


MAXBAR
Café
MAXBAR
Heineken
Café
Espresso

. . . where he meets Jack

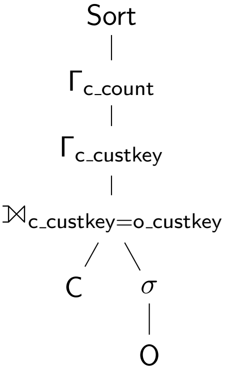
- Jack is an retired DBMS veteran.
- He likes telling old DBMS war stories.
- Joe tells Jack about the 30%.

Jack: Give me an example of a simple, long running query.

Query 13

```
select    c_count, count(*) as custdist
from      (select   c_custkey, count(o_orderkey) as c_count
           from     customer left outer join
                    orders on c_custkey = o_custkey
           and      o_comment not like
                    '%special%requests%'
           group by  c_custkey
          ) as c_orders (c_custkey, c_count)
group by  c_count
order by  custdist desc, c_count desc
```

Plan For Query 13



Jack: Are you sure this is the optimal plan?

Jack's Story

Jack In the 80s, I was working on Kardamon.
Kardamon was a main memory DBMS ...
[We cut out most of Jack's story.]
In Kardamom, we had the outer aggregation operator.
Do you know outer aggregation?

Joe No.

Jack I explain it to you.

Outer Aggregation

Jack explains outer aggregation.

R

a
1
2
3
3

S

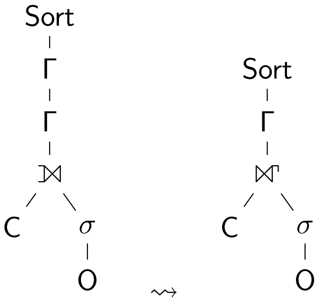
b	c
2	2
2	3
3	4
3	5
3	6
4	7

$R \Join\!\Gamma_{a=b;s:\text{count}(*),t:\text{sum}(c)}\ S$

a	s	t
1	0	null
2	2	5
3	3	15
3	3	15

Then, Jack says that "$⟕ + \Gamma = \Join\!\Gamma$".

The Napkin



Jack Groupjoin maybe a better name, since $⟕ + \Gamma = \bowtie^{\Gamma}$.

Joe This may give us a factor of two. At most.

Jack Wrong, it can give you more!

Tuesday

Back in office, Joe explains the groupjoin to his team, introduces them to "$⟕ + \Gamma = ⋈^{\Gamma}$", and shows why this is useful.

Implementing the Groupjoin

Joe and his collegue Abraham implement the groupjoin. They reuse a large deal of their main memory hash join. Their approach to implement $R \bowtie\!\!\ulcorner_{a=b;F}\, S$ is

1. Build a hashtable on $R.a$.
   It holds attributes from R and initialized aggregates from F.
2. Probe the hashtable with $S.b$.
   Aggregates are advanced for every tuple from S having a join partner.
3. Scan the hashtable, finalize the aggregates, and push the result to the next operator.

Then, they handcraft a plan for Query 13 and let it run.

Tuesday Evening

Joe and Abraham present the results.

Joe: The old runtime for Q 13 was 278 ms.

Abraham: The runtime for Q 13 now is 84 ms!

Bernie: Wow! A factor of 3.3!

Lucy: What about the other queries?

Joe: They don't benefit from the groupjoin. Q 13 is the only one with a left outerjoin.

They went home.

Wednesday

Lucy and Bernie meet early in the morning.

Lucy: It seems worthwhile.

Bernie: Yes.

Lucy: You know what you have to do?

Bernie: Yes.

And off they went.

Bernie on Wednesday

Bernie is responsible for cardinality and cost estimation.
Cardinality estimation for the groupjoin is very simple:

$$|R \bowtie^{\Gamma} S| = |R|$$

Cost Function for the Groupjoin

Bernie takes a look at the implementation and sees that the cost of the groupjoin is linear in its inputs:

$$\text{Cost}(R \bowtie^{\Gamma} S) = a + b|R| + c|S|$$

He generates a few test instances for R and S, measures the execution time of the groupjoin, approximates the results by the linear function under l_q, and gets

$$2.3 + 0.04 * |R| + 0.006 * |S| \quad (\mu s)$$

for his machine and a maximal q-error of 1.7.
[The next day, he refines the cost model to get a maximal q-error of 1.4.]

The Equivalence

Lucy comes up with

$$\Gamma_{G;F}(e_1 ⟕_{J_1=J_2} e_2) \equiv \Pi_C(e_1 \bowtie\!\!\Gamma_{J_1=J_2;F}\, e_2)$$

The most important conditions are

1. $G \to G_2^+$ and $G_1, G_2^+ \to \mathrm{TID}(e_1)$ hold in $e_1 ⟕_{J_1=J_2} e_2$,
2. $J_2 \to G_2^+$ holds in e_2, and
3. $\mathcal{F}(F) \subseteq \mathcal{A}(e_2)$.

Lucy sends this to Bernie together with the following question:

Can the groupjoin be worse than a left outerjoin followed by a group-by?

Lucy does the proof of the equivalence and then goes home.

Thursday

Early Thursday morning, while brushing her teeth, Lucy thinks that there is no difference between a join and a left outerjoin.
Well, almost.
She is all excited and writes (using lipstick of course) onto the mirror in her bathroom

$$\Gamma_{G;F}(e_1 \bowtie_{J_1=J_2} e_2) \equiv \Pi_C(\sigma_{c_2>0}(e_1 ⟕_{J_1=J_2;F\circ(c_2:\text{count}(*))} e_2))$$

(same conditions)

Thursday

- In the morning, Bernie assures Lucy that introducing the groupjoin never makes the plan worse.
- Lucy tells the other team members about the non-difference between join and left-outerjoin.
- Some time later, Joe comes back telling everybody that there are plenty of opportunities (in TPC-H) to apply Lucy's *mirror equivalence*.
- Lucy goes to her office and starts extending their query optimizer.

Lucy's Long Thursday

- Lucy plans the additions to her plan generator $\texttt{DP}_{\text{hype}}$.
- She implements a transformation such that any plan particle that consists of an (outer) join followed by a grouping is replaced by a groupjoin if the conditions are fulfilled.
- This unconditional transformation is possible since the transformation never results in a performance penalty.

Lucy has to work long hours this day ...

Friday Morning

Lucy is done.

They run TPC-H.

Joe prepares the slide for Max.

Friday Afternoon

Presentation time. In Max' office.

Joe's slide

Query	time(ms) with $\ltimes\!\urcorner$	time(ms) old
9	192	192
21	127	500
13	84	278
3	70	104
10	51	74
5	59	68
16	45	49
20	37	37
17	33	34
...	...	...
total	1295	1932

Joe Overall, the improvement is about 33%.

Max I knew you could do it.

THE END

Q-Error

Q-Paranorm:

$$||x||_Q := \max(x, 1/x)$$

Q-Error:

$$\text{q-error}(f, \hat{f}) := ||\hat{f}/f||_Q$$

Refined Cost Function

$$\text{Cost}(e_1 \bowtie\!\!\Gamma_{q;F}\; e_2) := a + b|e_1| + c|e_2| + ds$$

where

$$s = |e_2 \ltimes_q e_1| / |e_2|$$

$J_2 \not\to G_2$ in S

R_1
a
1

S		
c	d	e
1	8	1
1	9	2

$R_1 \bowtie_{a=c} S$			
a	c	d	e
1	1	8	1
1	1	9	2

$\Gamma_{a,e;\text{sum}(d)}(R_1 \bowtie_{a=c} S)$		
a	e	sum(d)
1	1	8
1	2	9

$R_1 \bowtie^{\Gamma}_{a=c;\text{sum}(d)} S$	
a	sum(d)
1	17

	G	G_1	G_2	J_2	G_2^+
1	$\{a,e\}$	$\{a\}$	$\{e\}$	$\{c\}$	$\{c,e\}$

$G_1, G_2^+ \not\to \mathrm{TID}(e_1)$ in $R \Join_{a=c} S$

R_2

a
1
1

S

c	d	e
1	8	1
1	9	2

$R_2 \Join_{a=c} S$

a	c	d	e
1	1	8	1
1	1	9	2
1	1	8	1
1	1	9	2

$\Gamma_{a;\mathrm{sum}(d)}(R_2 \Join_{a=c} S)$

a	sum(d)
1	34

$R_2 \Join^{\Gamma}_{a=c;\mathrm{sum}(d)} S$

a	sum(d)
1	17
1	17

	G	G_1	G_2	J_2	G_2^+
2	$\{a\}$	$\{a\}$	$\emptyset$	$\{c\}$	$\{c\}$

$G \not\to G_2^+$ in $R \ltimes_{a=c} S$

R_3

a	b
1	1
1	2

S

c	d	e
1	8	1
1	9	2

$R_3 \ltimes_{b=e} S$

a	b	c	d	e
1	1	1	8	1
1	2	1	9	2

$\Gamma_{a;\text{sum}(d)}(R_3 \ltimes_{b=e} S)$

a	sum(d)
1	17

$R_3 \ltimes\Gamma_{b=e;\text{sum}(d)} S$

a	b	sum(d)
1	1	8
1	2	9

	G	G_1	G_2	J_2	G_2^+
3	$\{a\}$	$\{a\}$	$\emptyset$	$\{e\}$	$\{e\}$