Research track – Query Processing and Optimization I
Cardinality Estimation for Having-Clauses

G. Moerkotte

Problem to solve

Estimate the result cardinality of

```
select    A,...
from      R
[ where   p ]
group by  A
having    aggr(B) [ = b | between l and u]
```

- Approach: extend simple profile (eSP)
- Motivation: simple profile is default in many DBMSs

The Simple Profile (System R)

consists of

- cardinality $|R|$
- for every attribute X: $\min_X$, $\max_X$, d_X (number of distinct values)

Cardinality estimation and propagation over algebraic operators require assumptions:

- uniform distribution assumption (UDA)
  (of attribute values in *base* relations)
- independence assumption (IA)
  (if not obviously violated)
- ...

Query for Count

We start with the aggregate function count and consider the following example query:

```
select    ...
from      Lineitem
group by  l_orderkey
having    count(*) [ = b | between l and u]
```

Note:

- TPC-H attribute values are uniformly distributed
- TPC-H is well-known

count(∗): data analysis

Query Q_c		Result of Q_c	
		C	F_c
select	C, count(*) as F_c	1	214'172
from	(select count(*) as C	2	214'434
	from Lineitem	3	214'379
	group by l_orderkey)	4	213'728
group by	C	5	214'217
order by	C	6	214'449
		7	214'621

We observe that

- the values of C are all in $[1, 7]$, and
- the values of F_c are all about equal.

An estimate for F_c is denoted by $\hat{F}_c$. If we assume the counts C to be uniformly distributed, then all F_c ($\hat{F}_c$) are equal, we use F ($\hat{F}$) to denote that number.

Extended Simple Profile (eSP)

relations and attributes	
R	relation in the `from` clause
A	attribute(s) of R in the `group by` clause
B	(derived) attribute of R in some aggregate function in the `having` clause
C	defined as `count(*) as C` (see Query Q_E)
simple profile for R, $X \in \{A, B\}$	
$\|R\|$	cardinality of relation R
$\min_X$	minimum value of attribute X
$\max_X$	maximum value of attribute X
d_X	number of distinct values of attribute X
extension	
$\min_C$	minimum value of `count(*)`
$\max_C$	maximum value of `count(*)`
d_C	number of distinct values for `count(*)`

Calculation of Extensions using DuckDB

Query Q_E:

```
select min(C), max(C), count(distinct C)
from  (select count(*) as C
       from R
       group by A)
```

Thus, no big extension to DBMS necessary.

count: Cardinality Estimation Alternatives

blind Use some constant for the selectivity (e.g. 0.3).

one eyed guess some distribution for C and its moments (without looking at the result of Query Q_c)

eSP Store the result of Query Q_E (minimum, maximum, number of distinct values of C) and assume a uniform distribution.

cmp Compactify the result of Query Q_c using

- a histogram,
- some standard approximation techniques, or
- some parameterized distribution (preferable: finite support, discrete)

all Completely store the result of Q_c (if it is small).

count: Cardinality Estimation with UDA (eSP)

The estimates for the number of result tuples of our query template for

- `having count(*) = c` or
- `having count(*) between l and u`

are then produced by

$$\hat{E}[\text{cnt}](c) = \frac{d_A}{\max_C - \min_C + 1} \quad // \text{ independent of } c$$

$$\hat{E}[\text{cnt}](l,u) = \sum_{k=l}^{u} \hat{E}[\text{cnt}](k) = (u - l + 1)\hat{F}$$

count: Cardinality Estimation Precision (q-error)

`having count(*) = c`				
c	White	Fent	β-D	eSP
0	inf	inf	1	1
1	3.678	3.677	1.389	1.001
2	1.182	1.182	1.001	1.001
3	1.222	1.222	1.321	1
4	1.385	1.385	1.407	1.003
5	1.223	1.223	1.320	1
6	1.181	1.181	1.001	1.001
7	2.180	2.180	1.386	1.002
8	inf	inf	1	1

White 2017 (SqlServer); Fent, Neumann PVLDB 2019; eSP: extended simple profile (UDA for C).

Query for Sum

select ...
from Lineitem
group by l_orderkey
having sum(l_quantity) [= b | between l and u]

sum(B): data analysis: distribution of `l_quantity`

Query Q_q

```
select    l_quantity,
          count(*)
from      Lineitem
group by  l_quantity
order by  l_quantity
```

Result of Q_q

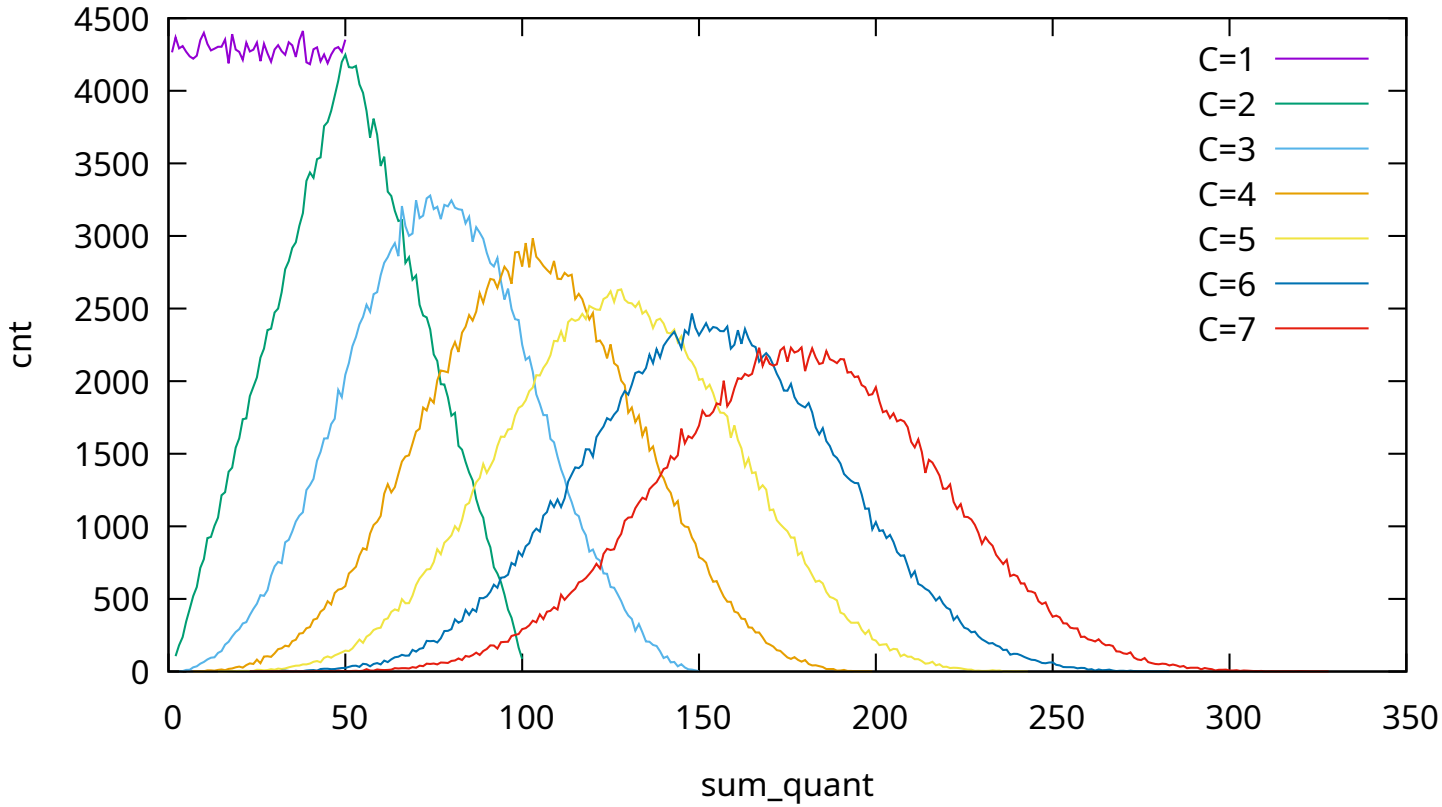
l_quantity	count(*)
1	120'401
2	119'460
3	120'047
...	...
48	120'191
49	119'624
50	119'846

We observe that the values of `l_quantity` are all in $[1, 50]$. Further, they are uniformly distributed.

sum(B): distribution of `sum(l_quantity)`

```
select    C, sum_quant, count(*) as cnt
from      (select l_orderkey,
                  count(*) as C,
                  sum(l_quantity) as sum_quant
           from Lineitem
           group by l_orderkey)
group by C, sum_quant
order by C, sum_quant;
```

sum(B): distribution of `sum(l_quantity)`



Observation: except for small C: sum is normally distributed

sum(B): solutions

- for $C = 0$ use uniform distribution to produce estimate
- for $C > 0$ we have a choice:
  - for $C > 0$ use normal distribution (eSP)
  - for $C > 0$ use integer compositions (IC)

sum(B): evaluation

having sum(l_quantity) = b:

`having sum(l_quantity) = b` maximum q-error for *b*-ranges					
b-range	Fent	β-D	eSP(1)	eSP(2)	IC
$[1, 200]$	1.53	6.02	1.30	1.30	1.04
$[200, 249]$	2.96	4.03	1.36	1.29	1.07
$[250, 300]$	104.6	179.9	2.33	2.33	1.96

Outlook

- other aggregate functions: avg, min, max
- having-clause with and, or
- where-clause