Cardinality Estimation for Having-Clauses

Guido Moerkotte
University of Mannheim
Mannheim, Germany
moerkotte@uni-mannheim.de

ABSTRACT

We present several methods for estimating the result cardinality of single table queries with a `having` clause. More specifically, we provide cardinality estimates for predicates using the aggregate functions `count(*)`, `sum(B)`, `avg(B)`, `min(B)`, and `max(B)`. We do so for queries with and without a `where`-clause. Finally, we show how to handle conjunctions and disjunctions in the `having`-clause.

PVLDB Reference Format:
Guido Moerkotte. Cardinality Estimation for Having-Clauses. PVLDB, 18(1): 28 - 41, 2024.
doi:10.14778/3696435.3696438

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at github.com/moerkotte/having.

1 INTRODUCTION

The goal is to discuss methods to estimate the result cardinality of group-by-having (GBH) queries. More specifically, we consider cardinality estimates for the following GBH-query pattern:

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

Predicates of the form aggr$(B)\theta b$ for $\theta \in \{\neq, <, \leq, >, \geq\}$ can be reduced to equality for $\neq$ and to `between` queries for the other θ.

The relevance of considering cardinality estimation for GBH-queries becomes clear if we consider them occurring nested in some larger query. Examples thereof are TPC-H Q18 [67] and TPC-DS Q8, Q14, Q23, Q44, and Q64 [68]. Some of these contain one or more GBH-queries nested in the `from`-clause together with other relations. In these cases, the optimal join order and the choice of the optimal join implementation highly depends on the cardinality of the GBH-queries. Besides these benchmark queries, we assure the reader that real customer queries with nested GBH-queries exist[1].

In their seminal paper Selinger et al. introduced what we call the *simple profile* (SP) [63]. The main idea of the simple profile is to store only a few numbers per relation and per attribute thereof and use the uniform distribution assumption and the independence assumption to come up with cardinality estimates. Here, we introduce the *extended simple profile* (eSP) by adding a few more numbers to the simple profile and discuss existing and new methods to produce estimates for the result cardinality of GBH-queries. The eSP-based estimation methods assume a uniform distribution of the values in attributes A and B. The estimates produced by eSP can be seen as a default in the lack of better alternatives (which mostly have to be developed in the future).

Table 1: Supported SQL-Constructs

	White	Fent+	here
count	+	+	+
sum/avg		+	+
min/max		(+)	+
and/or			+
where			+

Note that almost nothing has been published on estimating the selectivity of *having* clauses. In fact, during a literature and an internet seach we only found two descriptions of cardinality estimation methods for `having` clauses. The first is a blog entry by White describing the selectivity estimation in SQL-Server for having predicates in `count(*)` [74]. The second is a paper by Fent and Neumann [21]. We will discuss both methods in detail. Table 1 gives an overview of supported SQL-constructs. Fent and Neumann's approach only supports `min/max` for numerical attributes.

Since the main focus is on uniformly distributed values of A and B, we use the data from the TPC-H benchmark [67] (with scale factor SF=1) and modifications of Query 18 to illustrate the different approaches.

The rest of the paper is organized as follows. Section 2 provides a superset of the statistics required for all approaches. It also contains SQL-queries to derive these statistics. Further, the according numbers for the TPC-H `Lineitem` table are provided for further reference. Section 3 deals with predicates in `count(*)`. Section 4 deals with predicates in `sum(B)` and `avg(B)`. Section 5 deals with predicates in `min(B)` and `max(B)`. The problem of multiple predicates in the `having`-clause either conjunctively or discjunctively connected is handled in Section 6. So far, all methods discussed have ignored a possible `where`-clause with an according selection predicate. Therefor, Section 7 discusses estimation methods in case the query contains a `where`-clause. Section 8 contains related work. Section 9 concludes the paper.

2 PRELIMINARIES

Table 2 contains the notation used in the paper. We assume a relation R with attributes A and B is given. In our query pattern,

Proceedings of the VLDB Endowment, Vol. 18, No. 1 ISSN 2150-8097.
doi:10.14778/3696435.3696438

[1]There even are blogposts on it: oracle-randolf.blogspot.com/2013/01/having-cardinality.html [last accessed 01.09.24]

Table 2: Notation

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)
standard statistics for $X \in \{A, B\}$	
μ_X	mean of attribute X
σ_X	standard deviation of attribute X
γ_X	skewness of attribute X
simple profile for R, $X \in \{A, B\}$	
$\lvert R\rvert$	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(*)`
μ_C	mean of `count(*)`
σ_C	standard deviation of `count(*)`
γ_C	skewness of `count(*)`

Table 3: Numbers for TPC-H (SF1)

standard statistics		used by White	Fent+	β-D	eSP
$\mu_{\texttt{l\_orderkey}}$	3'000'279.60			x	d
$\sigma_{\texttt{l\_orderkey}}$	1'732'187.87			x	d
$\gamma_{\texttt{l\_orderkey}}$	-0.000'178'9		o		
$\mu_{\texttt{l\_quantity}}$	25.508		x	x	d
$\sigma_{\texttt{l\_quantity}}$	14.426		x	x	d
$\gamma_{\texttt{l\_quantity}}$	-0.001		x		
simple profile					
`|Lineitem|`	6'001'215	x	x		x
$\min_{\texttt{l\_orderkey}}$	1			x	x
$\max_{\texttt{l\_orderkey}}$	6'000'000			x	x
$d_{\texttt{l\_orderkey}}$	1'500'000	x	x		x
$\min_{\texttt{l\_quantity}}$	1			x	x
$\max_{\texttt{l\_quantity}}$	50			x	x
$d_{\texttt{l\_quantity}}$	50				x
extension					
$\min_C$	1			x	x
$\max_C$	7			x	x
d_C	7				x
μ_C	4.00	d	d	x	d
σ_C	2.00	d	d	x	d
γ_C	0.00		d		

x: used as input; d: derived; o: optional, default if unavailable

A is the grouping attribute and B is the argument of some aggregate function. The parameters used throughout are clustered into three parts. The first part contains *standard statistics* for numerical columns, like mean, standard deviation, and skewness. The *simple profile* stores the cardinality of relation R. Further, for all attributes the minimum and maximum value as well as the number of distinct values is stored. The idea then is to use the uniform distribution assumption and the independence assumption to derive cardinality estimates [63]. The simple profile is also described in detail in [50, Sec. 24.3]. Typically, the standard statistics is not part of the simple profile. The *extension part* contains some numbers on `count(*)` values. The eSP then contains the SP numbers as well as the first three numbers in the extension part. The numbers in the extension part can be calculated via Query Q_E (e.g. using DuckDB [61]):

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

Under certain assumptions, some of these numbers can be determined without evaluating Q_E, as we will see later. It is important to note that not all approaches presented will rely on all the numbers contained in Table 2. Thus, for some approaches Table 2 contains a strict superset of their required input. Table 3 contains the details about the usage of the numbers for different estimation methods presented below. An x means that the number must be available. A d means that the value is derived from the inputs marked by x. An o (optional) means that the value is used if it is avaible and a default value is used otherwise.

For our examples, we use the data of TPC-H (SF=1) [67] and the following query pattern, which covers the nested query block of Query 18:

```
select l_orderkey,...
from   Lineitem
[where p]
group by l_orderkey
having aggr(l_quantity) [ = b | between l and u]
```

The according numbers for R = `Lineitem`, A = `l_orderkey`, B = `l_quantity` are given in Table 3.

For completeness, Figure 1 contains Query 18 of the TPCH-H benchmark [67]. We see that the optimal query plan depends on the result cardinality of the nested query block. The runtimes of different plan alternatives for different cardinalities of the nested query block have been investigated by Fent and Neumann [21].

Finally, in all our evaluations, we use the q-error [53]. If $x \geq 0$ is some number and $e \geq 0$ is an estimate thereof, the q-error(e,x) is defined as

$$\text{q-error}(e, x) = \begin{cases} 1 & \text{if } e = 0 \land x = 0 \\ \max(x/e, e/x) & \text{if } e \neq 0 \land x \neq 0 \\ \infty & \text{else} \end{cases}$$

3 PREDICATES IN COUNT

The query pattern we consider in this section is

select	...
from	R
group by	A
having	count(*) [= b | between l and u]

```
select   c_name, c_custkey, o_orderkey, o_orderdate,
         o_totalprice, sum(l_quantity)
from     customer, orders, lineitem
where    o_orderkey in (select l_orderkey
                        from  lineitem
                        group by  l_orderkey
                       having sum(l_quantity) > PARAM)
         and  c_custkey = o_custkey
         and  o_orderkey = l_orderkey
group by c_name, c_custkey, o_orderkey,
         o_orderdate, o_totalprice
order by o_totalprice desc, o_orderdate
```

Figure 1: Query 18 of TPC-H

First note that the value of `count(*)` is always an integer and that 1 is a lower bound. This even holds if R is empty, as then there is no tuple which could violate the lower bound. The upper bound can, in general, be quite large but is of course bounded by $|R|$, a case which occurs if there is only a single value in A.

3.1 Data Analysis

The true numbers for `count(*)` for `Lineitem` can be derived by the following SQL query, which returns the different number of lineitems per `l_orderkey` (C) and the frequency (F_c) of their occurrence:

Query 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

(Result of Q_c: columns C and F_c.)

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.

3.2 The Alternatives

The following enumeration lists alternative approaches to deal with the different values for `count(*)` aka C:

(1) Guess some distribution for C and its moments without looking at the result of Query Q_c (Sec. 3.3 and 3.4).
(2) Take a look at the result of Query Q_C and store minimum, maximum, number of distinct values of C and assume a uniform distribution (Sec. 3.5).
(3) Compactify the result of Query Q_c using
    (a) a histogram,
    (b) some standard approximation techniques, or
    (c) some parameterized distribution (preferable: finite support, discrete) (Sec. 3.6)
(4) Completely store the result of Q_c, if it is as small as it is for `l_orderkey`.

Note that all but the first solution require evaluating Q_c. We discuss two existing approaches of Alternative 1 in Sections 3.3 and 3.4. A new approach for Alternative 2 is presented in Section 3.5. A new approach for Alternative 3 is discussed in Sec. 3.6. Concerning Alternative 4, observe that the result size of Q_c is less than $\min(\sqrt{2|R|}, d_A)$, which should make it a viable approach in many cases. However, since this approach is trivial (and precise), we will not discuss it any further.

3.3 Normal Distribution (White: Alt. 1)

White describes the SQL-Server approach[74]. In short, a normal distribution for the values of `count(*)` is assumed. Define d_A as the number of distinct values contained in attribute `A`. By using a single scan over the relation, d_A can either be determined precisely or approximated using a sketch [8, 20, 22]. Remember that using sampling to determine the number of distinct values is not a good idea [13].

By using d_A, we can determine the mean

$$\mu_C = \frac{|\mathrm{R}|}{d_A} \; [= \frac{6'001'215}{1'500'000} \approx 4].$$

where the numbers in brackets are for `Lineitem` and `l_orderkey`. Looking at Table 3 for the true value of μ_C, we can see that the above estimate is precise. This is of course no surprise, as the values in `l_orderkey` are uniformly distributed.

Since nothing is known about the standard deviation, a standard deviation of

$$\sigma_C = \sqrt{\mu_C} \; [\approx 2]$$

is pretty arbitrarily assumed. Again, the number in brackets is for `l_orderkey`. Comparing this number with the precise value $\sigma_C = 2.00$ from Table 3, we see that it matches.

Then, the cumulative distribution function Φ of the normal distribution is used to calculate the estimate. The estimated selectivity for `count(*) = c` is

$$\hat{s}(c) = \begin{cases} \Phi_{\mu'_C,\sqrt{\mu_C}}(1.5) - \Phi_{\mu'_C,\sqrt{\mu_C}}(1) & \text{if } c = 1 \\ \Phi_{\mu'_C,\sqrt{\mu_C}}(c+0.5) - \Phi_{\mu'_C,\sqrt{\mu_C}}(c-0.5) & \text{else} \end{cases} \quad (1)$$

where $\mu'_C = ((d_A - 1)/d_A)\mu_C$. The final cardinality estimate is then produced by

$$\hat{E}_N[cnt](c) = \hat{s}(c) d_A \quad (2)$$

For `count(*) between l and u`, the selectivity is estimated as

$$\hat{s}(l,u) = \begin{cases} \Phi_{\mu'_C,\sqrt{\mu_C}}(u+0.5) - \Phi_{\mu'_C,\sqrt{\mu_C}}(1) & \text{if } l = 1 \\ \Phi_{\mu'_C,\sqrt{\mu_C}}(u+0.5) - \Phi_{\mu'_C,\sqrt{\mu_C}}(l-0.5) & \text{else} \end{cases} \quad (3)$$

The final cardinality estimate is then produced by

$$\hat{E}_N[cnt](l,u) = \hat{s}(l,u) d_A \quad (4)$$

Using the SQL Server method to estimate the cardinality for `having count(*) = c`, we get for different c in Table 4. We observe that

- the maximal q-error of $\hat{E}$ (SQL Server's estimate) is ∞ for $c > 7$ and 3.68 for $c = 1$

Although this does not look too bad for this example, there comes a couple of obvious deficiencies with this approach:

Table 4: SQL-Server q-Errors

c	true cardinality	$\hat{E}_N[cnt](c)$	q-error
1	214'172	58'237	3.68
2	214'434	181'394	1.18
3	214'379	261'928	1.22
4	213'728	296'090	1.39
5	214'217	262'032	1.22
6	214'449	181'538	1.18
7	214'621	98'456	2.18
8	0	41'797	inf

(1) the counts may not be normally distributed,
(2) the arbitrary guess for the standard deviation may be good or bad, and
(3) the normal distribution is open ended to $\pm\infty$, whereas the values of count C definitely have a lower bound of 1 and an upper bound $|R| - d_A + 1$.

3.4 Skew-Normal Distribution (Fent: Alt. 1)

In the approach of Fent and Neumann[21], the mean μ_C is calculated the same way as in the SQL-Server approach. The standard deviation is assumed to be

$$\sigma_C = \sqrt{|R| * p_A * (1 - p_A)} \tag{5}$$

where $p_A = 1/d_A$. For A = l_orderkey, this formula results in an estimate of $\sqrt{6001215 * (1/1500000) * (1 - (1/1500000))} \approx 2$ which is almost equal to the true value of 2.00 (see Table 3). Note that these numbers are the same as in the SQL-Server approach discussed in the previous subsection.

Finally, Fent and Neumann calculate the skewness γ_C by the following formula:

$$\gamma_C = \begin{cases} \gamma_A & \text{if known} \\ 0 & \text{else} \end{cases} \tag{6}$$

For A = l_orderkey, γ_C is estimated as $\gamma_{\text{l\_orderkey}} \approx 0$ (see Table 3) in both cases. This perfectly matches the true value of $\gamma_C = 0$.

With these three parameters, a skew-normal distribution [6] $\mathrm{SN}(\mu_C, \sigma_C, \gamma_C)$ is used to produce the estimates, which we also call $\mathrm{SN}_C(\mu_C, \sigma_C, \gamma_C)$ for further reference. Note that this is a slight abuse of notation to which we stick for simplicity. In reality, the skew-normal distribution has three parameters ξ, η, λ [6], which must be calculated from μ_C, σ_C, and γ_C using the method of moments [29, 59, 62]. For convenience, Appendix C contains the necessary formulas.

Let $\Phi_{\mathrm{SN}(\mu_C,\sigma_C,\gamma_C)}$ be the cumulative distribution function of SN_C. Since the description [21] lacks any details, we can assume that we calculate the estimates the same way as SQL-Server (using Eqns. 1 to 4), except that $\Phi_{\mathrm{SN}(\mu_C,\sigma_C,\gamma_C)}$ is used instead of $\Phi_{\mu'_C,\sqrt{\mu_C}}$.

3.5 Uniform Distribution (eSP: Alt. 2)

We now extend the simple profile by three numbers: d_C, $\min_C$, and $\max_C$.

Since the counts are always integers, the simple profile applies the *discrete* uniform distribution. 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

$$\hat{E}[\mathrm{cnt}](c) = \frac{d_A}{\max_C - \min_C + 1} \text{ // independent of } c \tag{7}$$

We introduce the following abbreviation:

$$\hat{F}_k = \hat{E}[\mathrm{cnt}](k) \tag{8}$$

Since for the uniform distribution, $\hat{E}[\mathrm{cnt}](k)$ are all equal, we use simply $\hat{F}$ in this case, which of course is the same $\hat{F}$ as in Sec. 3.1. Then, for range queries we have

$$\hat{E}[\mathrm{cnt}](l, u) = \sum_{k=l}^{u} \hat{E}[\mathrm{cnt}](k) \;=\; (u - l + 1)\hat{F}. \tag{9}$$

As a side remark observe that for the discrete uniform distribution on some interval $[a, b]$ we have that

$$\mu = \frac{a+b}{2} \tag{10}$$

$$\sigma = \sqrt{\frac{(b-a+1)^2 - 1}{12}} \tag{11}$$

$$\gamma = 0 \tag{12}$$

When using these formulas we see that $\mu_C = 4$, $\sigma_C = 2$, and $\gamma_C = 0$ are pretty close to the true values in Table 3.

Note that contrary to the original simple profile, where only the *schema* determines which numbers are stored (for each relation and attribute), the queries of interest now determine what information we have to calculate and store. That is, for every attribute combination occurring in the `group by` clause of some *query* of interest, we need to calculate and store $\min_C$, $\max_C$, and if we cannot assume the counts to be dense we also need to calculate and store d_C. The latter can of course be approximated by some sketch [8, 20, 22].

Sampling can be used to calculate an approximation of the result of Q_c in order to reduce the amount of data to be scanned or to be processed in order to evaluate Q_c. However, we must be careful to make sure that for any orderkey all the lineitems belonging to it are really counted. If there is an index on `Lineitem.l_orderkey`, the index can be sampled for a sample of `l_orderkey` values. If there is no index, a complete scan is required. The amount of storage to be scanned can possibly be reduced by using small materialized aggregates (SMAs) if they exist [49]. The amount of storage used during the query evaluation can be reduced via sampling only a fraction of the `l_orderkey` values by, e.g., applying some selection predicate on the hash values of `l_orderkey`.

3.6 Beta-Distribution (Alt. 3)

The β-distribution has a finite support and is quite flexible. Thus, although it is a continous distribution, it fits quite well in case the counts C are not uniformly distributed. From the values $\min_C$, $\max_C$, μ_C, σ_C the parameters α and β of the β-distribution can again be derived using the method of moments. Then, the estimation procedure is exactly the same as in Eqns. 1 to 4 except that the cumulative distribution function of the β-distribution is used.

3.7 Evaluation

The q-errors for different values of c are given in the following table:

having count(*) = c				
c	White	Fent	β-D	eSP
	Sec. 3.3	Sec. 3.4	Sec. 3.6	Sec. 3.5
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

We observe that there is virtually no difference between the estimation method of SQL-Server as described by White [74] and the one proposed by Fent and Neumann [21]. This comes as no surprise as the mean and standard deviation calculated by both approaches coincide and the skewness is calculated to be zero by the approach of Fent and Neumann, in which case there is no difference between the normal and the skew-normal distribution. Apart from that, the β-distribution is slightly better and the clear winner is the simple profile.

It should be obvious that the more the real distribution deviates from the assumed distribution, the larger the q-error becomes. To illustrate this, we generated a relation with 10^6 tuples where the values of the grouping attribute A are Zipf distributed in $[1, 1000]$ with $z = 0 \ldots 1$. A Zipf distribution with parameter $z = 0$ corresponds to a uniform distribution and for $z = 1$ the distribution is highly skewed. As the having predicate we use `count(*) > 1000`. The resulting q-errors are

z	White	Fent	β-D	eSP
0	1	1	1	1.1
0.2	1.5	1.5	1.2	2.6
0.4	1.7	1.6	1.1	3.3
0.6	2.1	1.9	1.1	4.1
0.8	2.7	2.3	1.4	5.4
1	3.7	3	1.8	7.5

4 PREDICATES IN SUM(B) AND AVG(B)

Some estimation procedures require to sum up partial estimates over the possible `count(*)` values. For these, we define for all aggregate functions agg $\in$ {sum, avg, min, max}

$$\hat{E}[\text{agg}](b) = \sum_{k=\min_C}^{\max_C} \hat{E}_k[\text{agg}](b) \tag{13}$$

$$\hat{E}[\text{agg}](l,u) = \sum_{k=\min_C}^{\max_C} \hat{E}_k[\text{agg}](l,u) \tag{14}$$

where the first case covers `having agg(B) = b` and the second case `having agg(B) between l and u`.

In this section, we start with agg = sum. Thus, our query pattern becomes

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

We illustrate the approach using the following query, which we call Q'_{18} as it corresponds to the nested query block of Query 18 (see Fig. 1) of the TPC-H benchmark:

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

This section is organized as follows. First we perform some data analysis where we investigate the distributions of `l_quantity` and `sum(l_quantity)` (Sec. 4.1). Then we discuss the approach of Fent and Neumann [21], who propose to use a skew-normal distribution for `sum(B)` (Sec. 4.2). Next, we discuss the use of the β-distribution for `sum(B)` (Sec. 4.3). Then, we make a short excursion to k-compositions for an integer n and estimation methods using it (Sec. 4.4). Next, we present another new estimation procedure for the extended simple profile, which uses a normal distribution for `sum(B)` (Sec. 4.5). Finally, we show how to apply the estimation procedures for `sum(B)` to `avg(B)` (Sec. 4.6).

4.1 Data Analysis

4.1.1 Distribution of l-quantity. We first take a look at the distinct values of `l_quantity` and their frequencies, which are derived by Query Q_q:

Query Q_q		Result of Q_q	
		l_quantity	count(*)
select	l_quantity, count(*)	1	120'401
from	Lineitem	2	119'460
group by	l_quantity	3	120'047
order by	l_quantity	...	...
		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.

4.1.2 Distribution of sum(l_quantity). Let us turn to the distribution of `sum(l_quantity)`. The minimal theoretically possible value of `sum(B)` can be determined as $\min_C * \min_B$. The maximal theoretically possible value of `sum(B)` can be determined as $\max_C * \max_B$. From the above and the result of query Q_c, we can conclude that

- the minimum possible value for `sum(l_quantity)` is 1 (= $1 * 1$) and
- the maximum possible value of the sum is 350 (= $7 * 50$).

The minimum value of 1 really occurs, the actual maximum value is 328.

What we are interested in now is the distribution of the values for `sum(l_quantity)`. This distribution can be calculated via Query Q_d:

```
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;
```

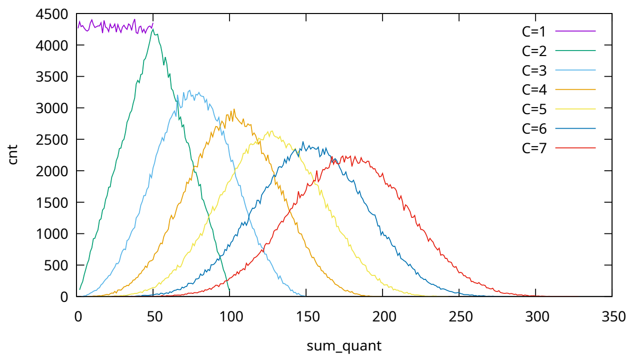

Figure 2: Distribution of frequencies of sum(l_quantity) values

Fig. 2 shows the resulting distributions. The x-axis represents `sum_quant` and the y-axis `cnt`. There is one curve for every possible value for $C \in [1, 7]$.

We observe the following:

- for $C = 1$, the data is uniformly distributed between 1 and 50.
- for $C = 2$, the curve can be approximated by two lines.
- for $C > 2$, an approximation via a normal distribution seems feasible.

Further, since there are very few values in the result of Query Q_d (only 1247 values), it could be materialized or approximated, e.g., via histograms [17] or some approximation technique with error guarantees [64].

4.2 Skew-Normal Distribution (Fent)

Again, Fent and Neumann propose to use the skew-normal distribution for `having`-predicates in `sum(B)` [21]. To derive this distribution, first the distribution of the values of B is modelled via a skew-normal distribution $\mathrm{SN}_B = \mathrm{SN}(\mu_B, \sigma_B, \gamma_B)$. Then, $\mathrm{SN}_C(\mu_C, \sigma_C, \gamma_C)$ from Sec. 3.4, is used to get the distribution $\mathrm{SN}_{\mathrm{sum}} = \mathrm{SN}(\mu_s, \sigma_s, \gamma_s) = \mathrm{SN}_C * \mathrm{SN}_B$, which approximates the distribution of `sum(B)`. If independence is assumed between C and B, the parameters of this product of two distributions can be determined by the usual multiplication rule.

For convenience, we provide the multiplication rule (extended by some boundary checks):

$$\mu_s = \mu_C * \mu_B \tag{15}$$

$$\sigma_s = \sqrt{\max(0, m_{2,s} - \mu_s)} \tag{16}$$

$$\gamma_s = \begin{cases} 0 & \text{if } \sigma_s \leq 0.001 \\ \frac{m_{3,C} * m_{3,B} - 3\mu_s * \sigma_s + 2 * \mu_s^3}{\sigma_s^3} & \text{else} \end{cases} \tag{17}$$

where

$$\begin{aligned} m_{2,C} &= \sigma_C^2 + \mu_C^2 \\ m_{2,B} &= \sigma_B^2 + \mu_B^2 \\ m_{2,s} &= m_{2,C} * m_{2,B} \\ m_{3,C} &= \gamma_C * \sigma_C^3 - 2\mu_C^3 + 3\mu_C * m_{2,C} \\ m_{3,B} &= \gamma_B * \sigma_B^3 - 2\mu_B^3 + 3\mu_B * m_{2,B} \end{aligned}$$

Let us denote by Φ_{SN} the cumulative distribution function of the thus derived skew-normal distribution. Then, estimates are produced as follows:

$$E_{\mathrm{SN}}[sum](b) = d_A * (\Phi_{\mathrm{SN}}(b + 0.5) - \Phi_{\mathrm{SN}}(b - 0.5)) \tag{18}$$

$$E_{\mathrm{SN}}[sum](l, u) = d_A * (\Phi_{\mathrm{SN}}(u) - \Phi_{\mathrm{SN}}(l)) \tag{19}$$

4.3 Beta-Distribution (Beta-D)

This approach works exactly the same way as in the approach in the previous section. First, the distribution parameters μ_s, σ_s, and γ_s for $\sum(B)$ are calculated. Then, the cumulative distribution function of the β-distribution is used in Eqns. 18 and 19.

4.4 Integer B: Counting Integer Compositions (IC)

Next, we discuss another possible approach relying on counting integer compositions. The reason is that for small group sizes C, it would be nice to have a more precise approximation than the normal distribution. In this section, we assume that the attribute B is of type integer. Remember that `l_quantity` is also of type integer.

We start with the case $C = 1$. Thus, we consider all groups with exactly one tuple. Then, the distribution of the $\mathrm{sum}(B)$ values is the same as the distribution of the B value themselves. Let us consider the estimate $\hat{E}_1[sum](b)$. Clearly, `sum(l_quantity)` is equal to the value of `l_quantity` in this single tuple. Thus, we can conclude that for any b the estimate is $\hat{E}_1 = (1/50)\hat{F}_1$, or in general

$$\hat{E}_1[sum](b) = (1/d_B)\hat{F}_1. \tag{20}$$

under the uniform distribution assumption for B. This estimate can be used independently of the type of B. Further, if more specific information about B is available (e.g., histograms), we can use it to add precision to this estimate in case the uniformity assumption does not hold.

Next is the case $C = 2$, that is, we consider groups of size 2 and here the integers come into play. We start with illustrating our approach with the very specific having predicate sum(l_quantity) = 77. Since we consider the special case $C = 2$, we must have two quantities a_1 and a_2 which sum up to 77. Thus, we have the following possibilities for a_1 and a_2 both ≥ 1:

$$77 = 1 + 76 = 2 + 75 = 3 + 74 = \ldots = 76 + 1$$

Since in our case $a_1, a_2 \in [1, 50]$, we assign the probability of $\mathrm{P}(a_i = z), z \notin [1, 50]$ to zero. The probability that $a_1 + a_2 = 77$ can thus be calculated as

$$\mathrm{P}(a_1 + a_2 = 77) = \sum_{i=1}^{50} \mathrm{P}(a_1 = i) * \mathrm{P}(a_2 = 77 - i).$$

If we assume a uniform distribution for the a_i values, then for all $x, y \in [1, 50]$ we have that $\mathrm{P}(a_1 = x) = \mathrm{P}(a_2 = y)$, and we know from Q_q that this probability is $p = 1/50$. Using this, the above becomes

$$\begin{aligned} \mathrm{P}(a_1 + a_2 = 77) &= \sum_{i=27}^{50} \mathrm{P}(a_1 = i) * \mathrm{P}(a_2 = 77 - i) \\ &= \sum_{i=27}^{50} p * p \\ &= p * p * \sum_{i=27}^{50} 1 \\ &= p * p * 24. \end{aligned}$$

The estimate is then $(1/50) * (1/50) * 24 * F_2 = (1/50) * (1/50) * 24 * 214434 = 2058$. The true value is 1979. The scheme here is that we have to count the number of possibilites we can compose the number n (here it was 77) from two numbers a_1 and a_2 and then multiply with the probabilities.

In general, we need to calculate the number of *integer compositions*. For a positive integer $n > 0$, a tuple $(a_1, \dots, a_k)$ of positive integers $a_i > 0$ with

$$\sum_{i=1}^{k} a_i = n$$

is called a *k-composition* of n [65, p. 14]. Every a_i is called a *part*.

The number of k-compositions is

$$\binom{n-1}{k-1} \tag{21}$$

and the total number of compositions is 2^{n-1} (for all possible k) (see [65, p. 14]). In this definition, the parts a_i can be arbitrarily large, only bounded by n. Unfortunately, this is not the case in our estimation task: the parts a_i must be in $[1, 50]$. Denote by m the maximum possible value for any a_i. Then, we can apply the above formula (Formula 21) to count the number of compositions if and only if the condition

$$n \leq m \tag{22}$$

holds. The reason is that Formula 21 counts *all* possibilities. Given $n = 77$ and $m = 50$, it would count, e.g., the possibilites

$$77 = 76 + 1 = 75 + 2 = 74 + 3 \dots$$

which cannot occur since $76, 75, 74 > 50 = m$.

An S-restricted k-composition requires $a_i \in S$ for some set of positive integers S. Providing a closed form expression for the number of S-restricted k-compositions is not possible [79]. However, some special cases (especially $k = 1, 2, 3$) still result in closed formulas (see Appendix A).

We resume producing cardinality estimates for our query pattern. Denote by $\hat{F}_k$ an estimate of the count frequencies (cf. query Q_c), $p_B = 1/d_B$ for d_B equals the number of distinct values of attribute B, and by $N(k, [\min_B, \max_B], n)$ the number of k-compositions of a positive integer n where each part a_i satisfies $\min_B \leq a_i \leq \max_B$. Then, the estimate for `having sum(B) = b` is

$$\hat{E}_{\mathrm{IC}}[sum](b) = \sum_{k=1}^{\max_C} p_B^k N(k, [\min_B, \max_B], b) \hat{F}_k \tag{23}$$

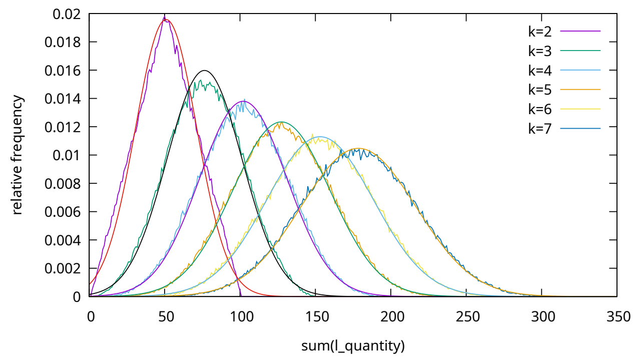

Figure 3: Relative frequency distribution of different `sum(l_quantity)` values and their approximation via normal distributions

For having `sum(B) between l and u`

$$\hat{E}_{\mathrm{IC}}[sum](l, u) = \sum_{b=l}^{u} \sum_{k=1}^{\max_C} p_B^k N(k, [\min_B, \max_B], b) \hat{F}_k \tag{24}$$

4.5 Normal Distribution (eSP)

In this subsection, we extend the simple profile.

4.5.1 Introduction. Since we already know that for a group size of $k = 1$, we have a uniform distribution (or in general, the same distribution as the B itself), we treat this case as in the previous section. For $k > 1$, an approximation of the probability distribution via the normal distribution is not too far off. In fact, the *central limit theorem* gives us the following (e.g. [25, Chap. 7], [29, p. 207]): If $X_1, \dots, X_n$ are random variables with any (!) fixed distribution, then for $k \to \infty$ the sum $X_1 + \dots + X_n$ is normally distributed with its mean being k times the mean of the X_i and its variance being k times the variance of the X_i. The question is, how large must k be for the normal distribution to become a viable approximation of $X_1 + \dots + X_n$.

Fig. 3 illustrates the approximation of the distribution for the sum of k `l_quantity`s for different $k \in [2, 7]$. Again, the x-axis shows the different possibilities for `sum(l_quantity)`. The y-axis shows the measured frequencies divided by the total frequency and the approximation via a normal distribution. The mean and standard deviation needed as parameters for the normal distribution were calculated using the result of query Q_d. We see that for $k = 2, 3$, the approximation is not too good. For $k > 3$, using the cumulative distribution function of the normal distribution for cardinality estimation (similar to Sec. 3.3) seems a viable way.

4.5.2 Extending the Simple Profile. We extend the simple profile by $\min_C$, $\max_C$, μ_B, σ_B, γ_B. The latter three numbers can either be determined from the data or, if the *discrete* uniform distribution is assumed, be calculated using Equations 10 to 12. If we do so, we get the estimates $\mu_B = 25.5$, $\sigma_B = 14.4309$, $\gamma_B = 0$ for $B =$ `l_quantity`, which are very close to the true values in Table 3.

We abbreviate the cumulative distribution function (cdf) of the normal distribution by $\Phi_{\mu,\sigma}$, where μ is the mean and σ is the standard deviation. For our purpose, the mean and the standard deviation depend on the length k of our sums (which equals the count values). According to the central limit theorem, we define

$$\Phi_k = \Phi_{k\mu_B, \sqrt{k\sigma_B^2}} \tag{25}$$

Then, we define the estimators for the *equality* case as follows:

$$\hat{E}_1[\text{sum}](b) = p_B \hat{F}_1 \tag{26}$$
$$\hat{E}_2[\text{sum}](b) = p_B^2 N(2, [\min_B, \max_B], b) \hat{F}_2 \tag{27}$$
$$\hat{E}_3[\text{sum}](b) = p_B^3 N(3, [\min_B, \max_B], b) \hat{F}_3 \tag{28}$$
$$\hat{E}_k[\text{sum}](b) = p_B^k(\Phi_k(b + 0.4) - \Phi_k(b - 0.4))\hat{F}_k \tag{29}$$

where $p_B = 1/d_B$. The equations for $\hat{E}_1$ and the general procedure for $\hat{E}_k$ do not contain any reference to $N(k, [a, b].n)$ and can thus serve as fallback if no implementation of it is available or if B contains floating point numbers.

The estimators for the *between* case are:

$$\hat{E}_1[\text{sum}](l, u) = \frac{u - l}{\max_B - \min_B} \hat{F}_1 \tag{30}$$
$$\hat{E}_k[\text{sum}](l, u) = (\Phi_k(u + 0.4) - \Phi_k(l - 0.4))\hat{F}_k \tag{31}$$

(Remember $\hat{F}_k$ from Eq. 8.) In both cases we used 0.4 instead of 0.5 since this gave slighly better estimates. Again, in case B is a floating point number, the above equations for $k = 1$ and general k can be used.

The following table contains the maximum q-error for `having sum(l_quantity) = b` occurring within different ranges for b. The rows for the simple profile contains the estimates produced by limiting the special cases: eSP(1) applies Eqns. 26 and 29, eSP(2) additionally uses Eqn. 27. The row IC denotes the estimates produced by Eq. 23.

`having sum(B) = 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

We make the following observations:

(1) The β-Distribution is the worst estimator.
(2) The Fent-Estimator is second but still not satisfactory.
(3) The other estimators are almost en par.
(4) IC is the most precise estimator.

4.6 Predicates in avg(B)

The estimation of predicates in `avg(B)` can be reduced to `sum(B)`. If B does not contain any NULL values, we have that

$$\text{avg}(B) = \frac{\text{sum}(B)}{\text{count(B)}}$$

and, thus,

$$\text{avg}(B) = \frac{\text{sum}(B)}{\text{count(B)}} = b$$

implies

$$\text{sum}(B) = \text{count(B)} * b.$$

In the presence of NULL values, we have to use $\text{count}^{\text{NN}}(B)$ instead of count(B), which only counts non NULL values of B.

The estimators are

$$\hat{E}_k[\text{avg}](b) = \hat{E}_k[\text{sum}](k * b) \tag{32}$$
$$\hat{E}_k[\text{avg}](l, u) = \hat{E}_k[\text{sum}](k * l, k * u) \tag{33}$$

5 PREDICATES IN MIN(B) AND MAX(B)

We propose a new estimation method for having predicates in $\min(B)$ and $\max(B)$ which are applicable for all types of B.

5.1 Estimation

Fent and Neumann suggest using the extreme value distribution, which they then approximate by the skew-normal distribution [21]. This implies that their approach is only applicable in case B is a number. However, B could also be of type `varchar`. Consequently, our estimation method does not depend on the type of B.

Let us first consider `having min(B) = b`. Two conditions must be fulfilled at the same time:

(1) One of the B values must be equal to b.
(2) All the other B values must be $\geq b$.

Accordingly, we define p to be the selectivity of $B = b$ and q to be the selectivity of $B \geq b$. For numbers and under the uniform distribution assumption, we can use $p = p_B = 1/d_B$ and $q = \frac{\max_B - b}{\max_B - \min_B}$. In either case, the estimate can then be produced by

$$\hat{E}_k[\min](b) = \begin{cases} pF_1 & k = 1 \\ kpq^{k-1}F_k & k > 1 \end{cases} \tag{34}$$

For `having max(B) = b` we can keep p but have to redefine q to be the selectivity of $B \leq b$. In case of numbers and under the uniform distribution assumption we can use $q = \frac{b - \min_B}{\max_B - \min_B}$. Then, the estimate can be produced by

$$\hat{E}_k[\max](b) = \begin{cases} pF_1 & k = 1 \\ kpq^{k-1}F_k & k > 1 \end{cases} \tag{35}$$

For `having min(B) between l and u`, we define p to be the selectivity of `B between l and u` and q to be the selectivity of `B between` $\min_B$ `and l`. For numbers and under the uniform distribution assumption, we can use

$$p = \frac{u - l + \delta}{\delta d_B}$$
$$q = \frac{u - \min_B + \delta}{\delta d_B}$$

where $\delta = (\max_B - \min_B)/(d_B - 1)$ is the average distance between two B values. In any case, the estimate is produced by

$$\hat{E}_k[\min](l, u) = \begin{cases} pF_k & k = 1 \\ kp(1 - q)^{k-1}F_k & k > 1 \end{cases} \tag{36}$$

For `having max(B) between l and u`, we just need to redefine q to the selectivity of `B between u and` $\max_B$, which for numbers under the uniform distribution assumption becomes

$$q = \frac{\max_B - u + \delta}{\delta d_B}$$

Then,

$$\hat{E}_k[\max](l, u) = \begin{cases} pF_k & k = 1 \\ kp(1 - q)^{k-1}F_k & k > 1 \end{cases} \tag{37}$$

5.2 Evaluation

The following table contains the q-errors for `min(l_quantity) = b` for different b:

b	eSP	Umbra
10	1.021	1.041
20	1.011	1.249
30	1.002	1.641
40	1.002	9.591
50	1.036	296.067

Since Fent and Neumann [21] do not provide sufficient details for a reimplementation, we give the Umbra estimates as we know that their approach has been integrated into it.

6 CONJUNCTIONS AND DISJUNCTIONS IN THE HAVING-CLAUSE

Conjunctions and disjunctions can be handled using the independence assumption. However, in case the `having`-clause contains an additional restriction on `count(*)`, special care is required.

6.1 Conjunctions

Obviously, any predicate in some aggregate function, no matter whether it is `sum`, `avg`, `min`, or `max` is correlated with any additional restriction on `count(*)`. Thus, applying the independence assumption is bound to fail. The following example illustrates this:

```
select count(*)
from (select l_orderkey
      from Lineitem
      group by l_orderkey
    having sum(l_quantity) < 20 and count(*) >= 4)
```

yields 134, whereas

```
select count(*)
from (select l_orderkey
      from Lineitem
      group by l_orderkey
    having sum(l_quantity) < 20 and count(*) <= 4)
```

yields 98021.

Thus, it is preferable not to use the independence assumption: Remember that all counts are in $[1, 7]$, `count(*)` ≥ 4 comprises the counts 1,2,3,4 and `count(*)` ≤ 4 comprises the counts 4,5,6,7. The selectivity of `sum(l_quantity)` < 20 is 0.06536. For both queries, the estimate would be $1500000 * 0.06535 * (4/7) \approx 56014$. Thus, treating the predicate on `count(*)` as an independent predicate with a selectivity 4/7 will result in very bad estimates.

Fortunately, our estimation procedure (see Eqs. 13 and 14) already contains a sum over the possible count values. Thus, restricting this sum to the range specified by the predicate on `count(*)` does the job. We only discuss the case `having agg(B) ... and count(*) between c and d`. Define $K = [c, d] \cap [\min_C, \max_C]$. Then, we adapt Eqs. 13 and 14 to

$$\hat{E}[\text{agg}](b, K) = \sum_{k \in K} \hat{E}_k[\text{agg}](b) \tag{38}$$

$$\hat{E}[\text{agg}](l, u, K) = \sum_{k \in K} \hat{E}_k[\text{agg}](l, u) \tag{39}$$

The produced estimates are

c	true	eSP	Umbra
[1, 4]	98'021	98'524	171'000
[4, 7]	134	466	174'000

where we added the estimates of Umbra [56], since Fent and Neumann did not discuss conjunctions in their paper.

If there are multiple conjunctively connected predicates in aggregates $\text{agg}(B_i)$ and there is a restriction on `count`, we assume conditional independence [18]

$$p(x_i, x_j|z) = p(x_i|z)p(x_j|z) \tag{40}$$

where the conditions on the $\text{agg}(B_i)$ are expressed as x_i and the restriction on `count` as z. If there is no restriction on `count`, independence is assumed.

6.2 Disjunctions

The query

```
select count(*)
from (select l_orderkey
      from Lineitem
      group by l_orderkey
    having sum(l_quantity) < 20 or count(*) >= 4)
```

yields 954907 and

```
select count(*)
from (select l_orderkey
      from Lineitem
      group by l_orderkey
    having sum(l_quantity) < 20 or count(*) <= 4)
```

yields 856718.

To produce an estimate for

```
agg(B) ... or count(*) between c and d
```

we first estimate the number of result tuples for the count case. Then, we add to this number the number of result tuples where we restrict the sum over the possible counts to the complement of the count restriction. Thus, after calculating the set of remaining counts

$$K = [\min_C, \max_C] \setminus [c, d]$$

we apply Eqns. 38 and 39.

The estimates for the above two queries are

c	true	eSP	Umbra
[1, 4]	856'718	857'229	173'000
[4, 7]	954'907	949'247	814'000

where we added the estimates of Umbra [56], since Fent and Neumann did not discuss disjunctions in their paper.

7 ADDING A WHERE-CLAUSE

We split our discussion in one part treating restrictions on `count(*)` and another for the remaining aggregate functions.

7.1 `count(*)`

Let us consider

```
select ..
from R
where p
group by A
```

```
having count(*) = c
```

Let s be the selectivity of p. Then

$$\hat{E}[cnt](c,s) = \sum_{k=c}^{\max_C} \binom{k}{c} (1-s)^{k-c} s^c \, \hat{F}_c \tag{41}$$

where we can replace the sum by a closed form expression (see Appendix B). The start index of the sum follows from the fact that any group with less than c elements *without* the where-clause can not qualify *with* the where-clause. Thus, consider a group of size k with $k \geq c$ and a fixed subset with c elements thereof. The probability that exactly these c elements survive and the other $k-c$ elements are eliminated is $(1-s)^{k-c}s^c$. Since there are exactly $\binom{k}{c}$ such subsets the claim follows.

For having count(*) between l and u, we simply have

$$\hat{E}[cnt](l,u,s) = \sum_{c=l}^{u} \sum_{k=c}^{\max_C} \binom{k}{c} (1-s)^{k-c} s^c \, \hat{F}_c \tag{42}$$

Again, the inner sum can be replaced by a closed form expression.

The q-errors produced for the equality case and the predicate l_suppkey modulo X = 0 for two different group sizes c are contained in the following table:

where l_suppkey % X = 0 having count(*) = c			
c	X	eSP	Umbra
1	2	1.000	1.402
1	4	1.001	1.016
1	10	1.002	1.777
4	2	1.000	1.201
4	4	1.009	1.288
4	10	1.009	6.860

Since Fent and Neumann [21] do not discuss where-clauses, we added the q-errors of Umbra.

7.2 Other Aggregate Functions

In this section we collectively treat the aggregate functions sum, avg, min, and max. However, before we do so let us have a look at some example queries, once without and once with a where-clause. Our example query without a where-clause is

```
select count(*)
from (select l_orderkey
      from Lineitem
      group by l_orderkey
      having sum(l_quantity) < 50)
```

and it yields 351'209, whereas the same query with an added where-clause

```
select count(*)
from (select l_orderkey
      from Lineitem
      where l_suppkey % 2 = 0
      group by l_orderkey
      having sum(l_quantity) < 50)
```

yields 623'204. We see that additional selection predicates can *increase* the result size. Clearly, using the predicate's selectivity (0.5) and multiplying the 351'209 of the first query without the *where*-clause with the selectivity 0.5 will go in the *wrong* direction.

Let us also consider another example query with and without a where-clause, but this time a more lucky one. The query without a where-clause

```
select count(*)
from (select l_orderkey
      from Lineitem
      group by l_orderkey
      having sum(l_quantity) > 50)
```

yields 1'137'386 and the same query with an added where-clause

```
select count(*)
from (select l_orderkey
      from Lineitem
      where l_suppkey % 2 = 0
      group by l_orderkey
      having sum(l_quantity) > 50)
```

yields 645'251. Accidentally, a multiplication of 1'137'386 by 0.5 does give a pretty good estimate.

We now come to our proposal for an estimation procedure for queries exhibiting a *where*-clause. For any aggregate function agg $\in$ {sum, avg, min, max} and selectivity s of the predicate p in the where clause, we have that

$$\hat{E}_k[\text{agg}](b,s) \quad = \quad \sum_{j=1}^{k} \binom{k}{j} (1-s)^{k-j} s^j \hat{E}_j[\text{agg}](b) \tag{43}$$

$$\hat{E}_k[\text{agg}](l,u,s) \quad = \quad \sum_{j=1}^{k} \binom{k}{j} (1-s)^{k-j} s^j \hat{E}_j[\text{agg}](l,u) \tag{44}$$

To see this, consider a group with k elements. The probability that exactly a fixed subset of j elements of this group survives is $(1-s)^{k-j}s^j$. Further, there are exactly $\binom{k}{j}$ such subsets. The equations follow. Since Eqns. 13 and 14 add another sum, this results in an unfortunate double sum. However, this can be eliminated (see Appendix B).

The following table provides the q-errors for the proposed estimation formula using the where-clause l_suppkey % 10 = 0:

having	eSP	Umbra
sum(l_quantity) < 50	1.010	1.333
sum(l_quantity) > 50	1.004	2.899

8 RELATED WORK

There exists a huge body of work on cardinality estimation. Standard techniques (including histograms [3, 7, 30–36, 38, 40, 41, 51, 54, 55, 60], wavelets [12, 23, 48, 71], sampling [15, 16, 26, 27, 45, 52, 54, 57, 70, 77], and sketches [1, 2, 8, 20, 22]) are reviewed by Cormode, Garofalakis, Haas, and Jermaine [17]). Other interesting approaches use the discrete cosine transform [42, 43], Baysian statistics [24, 69, 75] (from these, we borrowed the idea of using conditional independence), or learning techniques [19, 28, 39, 46, 66, 73, 76, 78]. None of these techniques have yet been applied to estimate the cardinality of GBH-queries.

Another interesting line of research is approximate query processing (e.g. [14, 47, 58] for an overview see [44]). So far, these techniques have not yet been applied to estimate the cardinality of GBH-queries.

If more complex expressions than simple arithmetic expressions in attributes are used in aggregate functions, statistical views may prove helpful [4, 9, 10, 72].

9 CONCLUSION

We conclude that implementing eSP as the default for cardinality estimation for GBH-queries into a DBMS is the right choice for several reasons:

(1) eSp uses only a minimal amount of storage.
(2) eSp is universally applicable.
(3) eSp fits into the context of the simple profile.

The latter is important since the simple profile has been implemented as the default into many systems. Further, the simple profile assumes a uniform distribution of attribute values. If the estimation procedures for GBH-queries would assume a different distribution, there would be a chasm in the default estimation procedure.

Finally, assuming a uniform distribution is not always correct and can lead to high estimation errors (see Sec. 3.7), which means that other methods have to be developed and applied for cases where the uniform distribution assumption results in unacceptably large q-errors.

A NUMBER OF POSITIVE INTEGER COMPOSITIONS

For integers $0 \leq l \leq m$, $k > 0$ and $n > 0$, let $N(k, [l, m], n)$ be the number of k-compositions $(a_1, \ldots, a_k)$ of integers $a_i \in [l, m]$ of n:

$$N(k, [l, m], n) = |\{(a_1, \ldots, a_k) \mid a_i \in [l, m], \sum_{i=1}^{k} a_i = n\}|. \quad (45)$$

Note that $N(k, [l, m], n) = 0$ if $n < kl$ or $n > km$.

If the lower bound for a_i is larger than 1, i.e., $l > 1$, then

$$N(k, [l, m], n) = N(k, [1, m - (l - 1)], n - k(l - 1)) \quad (46)$$

Thus, it suffices to consider the cases $l = 1$ and $l = 0$.

For $k \leq n \leq m$,

$$N(k, [1, m], n) = \binom{n-1}{k-1}. \quad (47)$$

If the lower bound $l = 0$, the problem is called *weak* k-composition. For $k \leq n \leq m$,

$$N(k, [0, m], n) = \binom{n+k-1}{k-1} \quad (48)$$

(see [5], [65, p14], for $a_i \in [0, m]$ see [11]).

$N(k, [l, m], n)$ can also be calculated using the following recurrence

$$N(k, [l, m], n] = \sum_{i=l}^{\min(m,n)} N(k - 1, [l, m], n - i), \quad (49)$$

but this is not a viable approach as it is highly compute intense. It becomes feasible, if the implementation considers the different cases for which a closed form expression exists.

We now consider the problematic case $n > m$, where we stick to the case $l = 1$. We want to derive simple formulas for the cases $k = 2$ and $k = 3$. For $k = 2$, we have that $N(2, [1, m], n) = 0$ if $n < 2$ or $2m < n$. In case $n > m$, $N(2, [1, m], n) = (m - (n - m) + 1)$ holds. This follows from the fact that $a_1 + a_2 = n$ and $n - m \leq a_1 \leq m$, since a_2 can be at most m, a_1 must be at least $n - m$. Further, it cannot be larger than m. Summarizing, for $k = 2$, we have in general that

$$N(2, [1, m], n) = \begin{cases} 0 & 2m < n \ \vee \ n < 2 \\ n - 1 & n \leq m, n \geq 2 \\ 2m - n + 1 & m < n \leq 2m, n \geq 2 \end{cases} \quad (50)$$

where the case $n \leq m$ follows from Eqn. 47.

For $k = 3$, we have the following;

$$N(3, [1, m], n) = \sum_{i=1}^{m} N(2, [1, m], n - i) \quad (51)$$

Clearly, $N(3, [1, m], n) = 0$ if either $n < 3$ or $3m < n$. Thus, in the following we assume $3 \leq n \leq 3m$.

Since some of the summands in Eqn. 51 are zero, we can refine the range for i:

$$\begin{aligned} a &= \begin{cases} n - 2m & n > 2m \\ 1 & else \end{cases} \\ b &= \min(m, n - 2) \\ N(3, [1, m], n) &= \sum_{i=a}^{b} N(2, [1, m], n - i) \end{aligned}$$

To see this, abbreviate $n' = n - i$. For the lower bound a, consider the case $n > 2m$. Since we must have that $n' = n - i \leq 2m$ for $N(2, [1, m], n') \neq 0$, thus

$$\begin{aligned} n - i &\leq 2m \\ n - 2m &\leq i \\ i &\geq n - 2m \end{aligned}$$

For the upper bound b, consider Eqn. 50. We see that $N(2, [1, m], n')$ is 0 if $2 > n'$ and thus

$$\begin{aligned} & 2 \leq n' \\ \iff & 2 \leq n - i \\ \iff & i \leq n - 2 \end{aligned}$$

Remember that we must have that $n \geq 2$ for $N(2, [1, m], n) \neq 0$. Now n' is in

$$\begin{aligned} & [n - b, n - a] \\ = & [n - \min(m, n - 2), n - \max(1, n - 2m)] \\ = & [\max(n - m, n - (n - 2)), \min(n - 1, n - (n - 2m))] \\ = & [\max(n - m, n - n + 2), \min(n - 1, n - n + 2m)] \\ = & [\max(n - m, 2), \min(n - 1, 2m)] \\ =: & [d, e] \end{aligned}$$

Next, we check the conditions under which the range is non-empty. We always have that $n - m \leq n - 1$ and $2 \leq 2m$ since $m \geq 1$. Further $2 \leq n - 1 \iff 3 \leq n$ and $n - m \leq 2m \iff n \leq 3m$. Thus, the conditions for $N(3, [1, m], n) \neq 0$ mentioned at the beginning of the case $k = 3$ guarantee a non-empty range. Since $N(k, [1, 1], n) = 1$ if $k = n$ and $N(k, [1, 1], n) = 0$ else, we assume $m > 1$.

Next, we see that according to Eqn. 50, we have to distinguish two non-zero cases. One for $n' \leq m$ (Case 1) and one for $n' > m$ (Case 2).

If $m \geq e$, we are always in Case 1. Thus

$$N(3, [1, m], n) = f_1(m, [d, e], n)$$

with

$$\begin{aligned} f_1(m, [d, e], n) &= \sum_{n'=d}^{e} n' - 1 \\ &= g(e) - g(d-1) - (e - d + 1) \end{aligned}$$

where $g(x) = x(x+1)/2$.

If $m < d$, we are always in Case 2. Thus

$$N(3, [1, m], n) = f_2(m, [d, e], n)$$

with

$$\begin{aligned} f_2(m, [d, e], n) &= \sum_{n'=d}^{e} 2m - n' + 1 \\ &= (e - d + 1)(2m + 1) - (g(e) - g(d-1)) \\ &= (e - d + 1)(2m + 1) - g(e) + g(d-1) \end{aligned}$$

In case $m \in [d, e]$, we only have to split the range $[d, e]$ into $[d, m]$ and $[m + 1, e]$:

$$N(3, [1, m], n) = f_2(m, [d, m], n) + f_1(m, [m + 1, e], n) \tag{52}$$

B USING THE HYPERGEOMETRIC FUNCTION

Denote by 2F1 the Gauss hypergeometric function [37, p17]. We will use the identity

$$\begin{aligned} &\sum_{k=1}^{n} (1-s)^{k-1} \binom{k+m-1}{m} \\ = \quad & s^{-m-1} - (1-s)^n \binom{m+n}{m} \text{2F1}(1, m+n+1, n+1, 1-s) \end{aligned}$$

For some given s, we define $H(m, N)$

$$\begin{aligned} = \quad & \sum_{k=m}^{N} \binom{k}{m} (1-s)^{k-m} \\ = \quad & \sum_{k=1}^{N-m+1} \binom{k+(m-1)}{m} (1-s)^{(k+(m-1)-m)} \\ = \quad & \sum_{k=1}^{N-m+1} \binom{k+m-1}{m} (1-s)^{k-1} \\ = \quad & s^{-m-1} - (1-s)^n \binom{m+n}{m} \text{2F1}(1, m+n+1, n+1, 1-s) \end{aligned}$$

where $n = N - m + 1$.

Concerning Eqns. 41 and 42 note that

$$\sum_{k=c}^{\max_C} \binom{k}{c} (1-s)^{k-c} s^c \hat{F}_c = s^c \hat{F}_c H(c, \max_C) \tag{53}$$

For Eqn. 43, we get using $a = \min_C$ and $b = \max_C$:

$$\begin{aligned} \hat{E}[agg](b, s) &= \sum_{k=a}^{b} \hat{E}_k[\text{agg}](b, s) \\ &= \sum_{k=a}^{b} \sum_{j=1}^{k} \binom{k}{j} (1-s)^{k-j} s^j \hat{E}_j[\text{agg}](b) \\ &= \sum_{j=1}^{b} \sum_{k=j}^{b} \binom{k}{j} (1-s)^{k-j} s^j \hat{E}_j[\text{agg}](b) \\ &\quad - \sum_{j=1}^{a-1} \sum_{k=j}^{a-1} \binom{k}{j} (1-s)^{k-j} s^j \hat{E}_j[\text{agg}](b) \\ &= \sum_{j=1}^{b} s^j \hat{E}_j[\text{agg}](b) \sum_{k=j}^{b} \binom{k}{j} (1-s)^{k-j} \\ &\quad - \sum_{j=1}^{a-1} s^j \hat{E}_j[\text{agg}](b) \sum_{k=j}^{a-1} \binom{k}{j} (1-s)^{k-j} \\ &= \sum_{j=1}^{b} s^j \hat{E}_j[\text{agg}](b) H(j, b) \\ &\quad - \sum_{j=1}^{a-1} s^j \hat{E}_j[\text{agg}](b) H(j, a-1) \end{aligned}$$

Eqn. 44 is handled the same way.

C SKEW-NORMAL DISTRIBUTION

Azzalini [6] introduced the skew-normal distribution $\text{SN}(\xi, \eta, \lambda)$ where

- ξ is the location parameter
- $\eta > 0$ is the scale parameter
- λ is the skewness parameter

For given mean μ, standard deviation σ and skewness γ, we can derive these parameters [59] by first defining

$$a = \left(\frac{2|\gamma|}{4-\pi}\right)^{\frac{1}{3}} \tag{54}$$

$$b = \frac{a}{\sqrt{1+a^2}} \tag{55}$$

$$\delta = \sqrt{\frac{\pi}{2}} b \tag{56}$$

Then, we must check whether $\delta \in [-0.99527, +0.99527]$. If it is outside, we set δ to the closest boundary of the interval. After that, we finally get

$$\eta = \sqrt{\frac{\sigma^2}{1-b^2}} \tag{57}$$

$$\xi = \mu - \eta b \tag{58}$$

$$\lambda = \frac{\delta}{\sqrt{1-\delta^2}} \tag{59}$$

ACKNOWLEDGMENTS

We thank Simone Kehrberg, Daniel Flachs, Nazanin Rashedi, and the reviewers for their comments. All of them helped to greatly improve the paper.

REFERENCES

[1] N. Alon, P. Gibbons, Y. Matias, and M. Szegedy. 2002. Tracking Join and Self-Join Sizes in Limited Storage. *J. Comput System Sciences* 35, 4 (2002), 391–432.

[2] N. Alon, Y. Matias, and M. Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. of Computer and System Sciences* 58, 1 (1999), 137–147.

[3] K. Alway and A. Nica. 2016. Constructing Join Histograms from Histograms with q-error Guarantees. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 2245–2246.

[4] C. Zuzarte an X. Yu. 2006. Fast Approximate Computation of Statistics on Views. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 724.

[5] G. Andrews. 1984. *The Theory of Partitions*. Cambridge University Press.

[6] A. Azzalini. 1985. A Class of Distributions which Includes the Normal Ones. *Scand. J. Statist* 12 (1985), 171–178.

[7] L. Baltrunas, A. Mazeika, and M Böhlen. 2006. Multidimensional Histograms with Tight Bounds for the Error. In *IDEAS*. 105–112.

[8] K. Beyer, P. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. 2007. On Synopses for Distinct-Value Estimation Under Multiset Operations. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 199–210.

[9] N. Bruno and S. Chaudhuri. 2002. Exploiting Statistics on Intermediate Tables for Query Optimization. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 263–274.

[10] N. Bruno and S. Chaudhuri. 2004. Condidional Selectivity Estimation for Statistics on Query Expressions. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 311–322.

[11] C. Caiado and P. Rathie. 2007. Polynomial Coefficients and Distributions of the Sum of Discrete Uniform Variables. In *Eighth Annual Conference of the Society of Special Functions and Their Applications*.

[12] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. 2001. Approximate Query Processing Using Wavelets. *The VLDB Journal* 10, 2-3 (2001), 199–223.

[13] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. 2000. Towards Estimation Error Guarantees for Distinct Values. In *Proc. ACM SIGMOD/SIGACT Conf. on Princ. of Database Syst. (PODS)*. 268–279.

[14] S. Chaudhuri, B. Ding, and S. Kandula. 2017. Approximate Query Processing: No Silver Bullet. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 511–519.

[15] Y. Chen and K. Yi. 2017. Two-Level Sampling for Join Size Estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 759–774.

[16] S. Christodoulakis. 1983. Estimating Block Transfers and Join Sizes. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 40–54.

[17] G. Cormode, M. Garofalakis, P. Haas, and C. Jermaine. 2012. *Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches*. NOW Press.

[18] A. Dawid. 1979. Conditional Independence in Statistical Theorey. *J. of the Royal Statistical Society: Series B (Methodological)* 41, 1 (1979), 1–15.

[19] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. Narasayya, and S. Chaudhuri. 2019. Selectivity Estimation for Range Predicates using Lightweight Models. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 1044–1057.

[20] O. Ertl. 2017. New Cardinality Estimation Algorithms for HyperLogLog Sketches. *arXiv* 1702.01284v2 (2017).

[21] P. Fent and T. Neumann. 2021. A Practical Approach to Groupjoin and Nested Aggregates. *Proc. of the VLDB Endowment (PVLDB)* 14, 1 (2021).

[22] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. 2007. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In *Conf. on Analysis of Algorithms (AofA). Discrete Mathematics and Theoretical Computer Science*. 127–146.

[23] M. Garofalakis and P. Gibbons. 2002. Wavelet synopses with error guarantees. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 476–487.

[24] L. Getoor, B. Taskar, and D. Koller. 2001. Selectivity Estimation Using Probabilistic Models. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 461–472.

[25] A. Gut. 2005. *Probability: A Graduate Course*. Springer.

[26] P. Haas, J. Naughton, S. Seshadri, and A. Swami. 1996. Selectivity and Cost Estimation for Joins Based on Random Sampling. *J. Comput. System Sci.* 52 (1996), 550–569.

[27] P. Haas, J. Naughton, and A. Swami. 1994. On the Relative Cost of Sampling for Join Selectivity Estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 14–24.

[28] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig. 2019. DeepDB: Learn from Data, not from Queries! *CoRR* (2019). http://arxiv.org/abs/1909.00607

[29] R. Hogg, E. Tanis, and L. Zimmerman. 2015. *Probability and Statistical Inference*. Pearson.

[30] Y. Ioannidis. 1993. Universality of Serial Histograms. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 256–267.

[31] Y. Ioannidis. 2003. The History of Histograms (abridged). In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 19–30.

[32] Y. Ioannidis and S. Christodoulakis. 1993. Optimal Histograms for Limiting Worst-Case Error Propagation in the Size of Join Results. *ACM Trans. on Database Systems* 18, 4 (1993), 709–748.

[33] Y. Ioannidis and V. Poosala. 1999. Histogram-based Approximation of Set-Valued Query Answers. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 174–185.

[34] Y. Ioannidis and V. Poosola. 1995. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 233–244.

[35] Y. Ioannidis and V. Poosola. 1995. Histogram-Based Solutions to Diverse Database Estimation Problems. *IEEE Data Engineering Bulletin* 18, 3 (Sept 1995), 10–18.

[36] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. 1998. Optimal Histograms with Quality Guarantees. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 275–286.

[37] N. Johnson, S. Kotz, and A. Kemp. 1992. *Univariate Discrete Distributions*. Wiley.

[38] C.-C. Kanne and G. Moerkotte. 2010. Histograms reloaded: the merits of bucket diversity. In *SIGMOD*. 663–674.

[39] A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper. 2018. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *arXiv preprint arXiv:1809.00677* (2018).

[40] A. König and G. Weikum. 1999. Combining Histograms and Parametric Curve Fitting for Feedback-Driven Query Result-Size Estimation. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 423–434.

[41] N. Koudas, S. Muthukrishnan, and D. Srivastava. 2000. Optimal Histograms for Hierarchical Range Queries. In *Proc. ACM SIGMOD/SIGACT Conf. on Princ. of Database Syst. (PODS)*. 196–204.

[42] J.-H. Lee, D.-H. Kim, and C.-W Chung. 1998. *Multi-dimensional Selectivity Estimation Using Compressed Histogram Information*. Technical Report CS/TR-98-131. KAIST.

[43] J.-H. Lee, D.-H. Kim, and C.-W Chung. 1999. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 205–214.

[44] Kaiju Li and Guoliang Li. 2018. Approximate Query Processing: What is New and Where to Go? *Data Science and Engineering* 3 (2018), 379–397.

[45] R. Lipton, J. Naughton, and D. Schneider. 1990. Practical Selectivity Estimation through Adaptive Sampling. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 1–11.

[46] H. Liu, M. Xu, Z. Yu, V. Corvinelli, and C. Zuzarte. 2015. Cardinality estimation using neural networks. In *CASCON*. 53–59.

[47] Q. Ma and P. Triantafillou. 2019. DBEst: Revisiting Approximate Query Processing Engines with Machine Learning Models. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 1553–1570.

[48] Y. Matias, J. Vitter, and M. Wang. 1998. Wavelet-Based Histograms for Selectivity Estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 37–48.

[49] G. Moerkotte. 1998. Small Materialized Aggregates: A Light Weight Index Structure for Data Warehousing. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 476–487.

[50] G. Moerkotte. 2024. Building Query Compilers. (2024). pi3.informatik.uni-mannheim.de/ moer/querycompiler.pdf.

[51] G. Moerkotte, D. DeHaan, N. May, A. Nica, and A. Böhm. 2014. Exploiting ordered dictionaries to efficiently construct histograms with q-error guarantees in SAP HANA. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 361–372.

[52] G. Moerkotte and A. Hertzschuch. 2020. α to ω: The G(r)eek Alphabet of Sampling. In *CIDR*.

[53] G. Moerkotte, T. Neumann, and G. Steidl. 2009. Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors. *Proc. of the VLDB Endowment (PVLDB)* 2, 1 (2009), 982–993.

[54] M. Müller, G. Moerkotte, and O. Kolb. 2016. Improved Selectivity Estimation by Combining Knowledge from Sampling and Synopses. *Proc. of the VLDB Endowment (PVLDB)* 11 (2016), 1016–1028.

[55] M. Muralikrishna and D.J. DeWitt. 1987. *Equi-Depth Multi-Dimensional Histograms*. Technical Report 733. Univ. Wisconsin.

[56] T. Neumann and M. Freitag. 2020. Umbra: a Disk-Based System with In-Memory Performance. In *Proc. Biennial Conference on Innovative Data Systems Research (CIDR)*. 29.

[57] F. Olken and D. Rotem. 1986. Simple Random Sampling from Relational Databases. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*.

[58] Y. Park, B. Mozafari, J. Sorenson, and J. Wang. 2018. VerdictDB: Universalizing Approximate Query Processing. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 1461–1476.

[59] A. Pewsey. 2000. Problems of Inference for Azzalini's skew-normal distribution. *J. of Applied Statistics* 27, 7 (2000), 859–870.

[60] V. Poosala and Y. Ioannidis. 1997. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 486–495.

[61] M. Raasveldt and H. Mühleisen. 2019. DuckDB: an Embeddable Analytical Database. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 1981–1984.

[62] V. Rohatgi. 1976. *An Introduction to Probability Theory and Mathematical Statistics*. Wiley.

[63] P. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proc. of the ACM*

SIGMOD Conf. on Management of Data. 23–34.

[64] S. Setzer, G. Steidl, T. Teuber, and G. Moerkotte. 2010. Approximation related to quotient functionals. *Journal of Approximation Theory* 162, 3 (2010), 545–558.

[65] R. Stanley. 1997. *Enumerative Combinatorics, Volume I*. Cambridge Studies in Advanced Mathematics, Vol. 49. Cambridge University Press.

[66] W. Tan, M. Alhamid, M. Kalil, R. Yang, V. Corvinelli, C. Zuzarte, and L. Finnie. 2021. Query Predicate Selectivity Using Machine Learning in DB2. In *CASCON*.

[67] Transaction Processing Council (TPC). 1993-2022. TPC Benchmark H (Standard Specification, Revision 3.0.1). (1993-2022). http://www.tpc.org/tpch.

[68] Transaction Processing Council (TPC). 2021. TPC Benchmark DS (Standard Specification, Revision 3.2.0). (2021). http://www.tpc.org/tpcds.

[69] K. Tzoumas, A. Deshpande, and C. Jensen. 2013. Efficiently Adapting Graphical Models for Selectivity Estimation. *Proc. Int. Conf. on Very Large Data Bases (VLDB)* 22 (2013), 3–27.

[70] D. Vengerov, A. Menck, M. Zait, and S. Chakkappen. 2015. Join Size Estimation Subject to Filter Conditions. *Proc. of the VLDB Endowment (PVLDB)* 8, 12 (2015), 1530–1541.

[71] J. Vitter and M. Wang. 1999. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 193–204.

[72] F. Waas, C. Galindo-Legaria, M.-C. Wu, and M. Joshi. 2003. Statistics on Views. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*. 952–962.

[73] J. Wang, C. Chai, J. Liu, and G. Li. 2024. Cardinality Estimation Using Normalizing Flow. *VLDB Journal* 33 (2024), 323–348.

[74] P. White. 2017. Cardinality Estimation for a Predicate on a COUNT Expression. (2017). https://sqlperformance.com/2017/04/sql-optimizer/cardinality-count [01.09.24].

[75] Ziniu Wu and Amir Shaikhha. 2020. BayesCard: A Unified Bayesian Framework for Cardinality Estimation. *CoRR* (2020). https://arxiv.org/abs/2012.14743

[76] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J. Hellerstein, S. Krishnan, and I. Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proc. of the VLDB Endowment (PVLDB)* 13, 3 (2019), 279–292.

[77] F. Yu, W.-C. Hou, C. Luo, D. Che, and M. Zhu. 2013. CS2: A New Database Synopsis for Query Estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*. 469–480.

[78] R. Zhu, Z. Wu, Y. Han, K. Zeng, A. Pfadler, Z. Qiang, J. Zhou, and B. Cui. 2020. FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation. *arXiv preprint arXiv:2011.09022* (2020).

[79] B. Zolfaghari, M. Fallah, and M. Sedighi. 2017. S-Restricted Compositions Revisited. *Discrete Mathematics and Theoretical Computer Science* 19 (2017).