α to ω: The G(r)eek Alphabet of Sampling

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

Axel Hertzschuch
University of Dresden
Axel.Hertzschuch@tu-dresden.de

ABSTRACT

Sampling is the most versatile and easiest to implement cardinality estimation method. Therefore, it is implemented in almost every database management system, commercial or not. Consequently, the main purpose of this paper is to provide the reader with an intuition about sampling precision. In the context of query optimization, the basic procedure can be described as follows. From a relation R containing n tuples, a sample of $m < n$ tuples is drawn. Then, a query predicate p is evaluated on the m sample tuples, and the number k of qualifying sample tuples is recorded. Assume the evaluation of the same predicate p on the relation R results in l qualifying tuples. The task now is to produce an estimate $\hat{l}$ for l where n, m, k are given. The standard answer to this task is $\hat{l} = k\frac{n}{m}$. However, there are some (yet unanswered) fundamental questions:

1. Is the standard estimator the best way to derive an estimate?
2. What are the upper and lower bounds for l?
3. How can we derive an estimate that minimizes the q-error?
4. How large is the q-error we can expect for this estimate?
5. For a given maximal allowed q-error, which sample size m should we choose?

Since sampling is a probabilistic process, we will give probabilistic answers to these questions. Further, we show how result cardinality estimates for selections and joins can significantly be improved.

1. INTRODUCTION

The purpose of this paper is twofold. Firstly, we want to sharpen the reader's intuition about q-errors produced by sampling. Secondly, we provide means to minimize the q-error by introducing new estimators for selections and joins.

This article is published under a Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), which permits distribution and reproduction in any medium as well allowing derivative works, provided that you attribute the original work to the author(s) and CIDR 2020.
CIDR 2020 January 12–15 , 2020, Amsterdam, NL

R	relation (or view content)
S	sample of R
p	query predicate
R_p	$\sigma_p(R)$
S_p	$\sigma_p(S)$
$n := \|R\|$	number of tuples in relation
$l := \|\sigma_p(R)\|$	number of qualifying tuples
$m := \|S\|$	number of samples
$k := \|\sigma_p(S)\|$	number of qualifying samples

Table 1: Abbreviations

After some preliminaries, we start out with a simple experiment establishing the q-error of estimates produced by sampling using the standard estimator (Sec. 3). Then, we improve the precision of these estimates by introducing two new estimators (Sec. 4). Afterwards, we look at some curves which solely serve the purpose of gaining more insights into sampling and errors produced (Sec. 5). Here, we gain some insights into the connection between sample size and expected q-errors. If the query optimizer is given a query with a conjunction of predicates, it typically needs estimates for partial conjuncts. Sec. 6 treats this case. Finally, we show how to significantly reduce the q-error of join size estimations produced by correlated sampling [21] and two-level sampling [7] in case of zero qualifying sample tuples (Sec. 7).

2. PRELIMINARIES

We assume that we are given a set of tuples R as a base relation or the contents of a view. From these tuples, we draw without replacement a sample (random subset) $S \subseteq R$. By $n := |R|$ we denote the number of tuples in R and by $m := |S|$ the sample size. Further, we are given a query predicate p. The result of evaluating p on R is denoted by $R_p := \sigma_p(R)$ and its size by $l = |R_p|$. Analogously, we define $S_p := \sigma_p(S)$ and $k := |S_p|$. For convenience, these abbreviations are summarized in Table 1.

The q-error of some estimate e for some true value t is defined as $\max(e/t, t/e)$. We will use the q-error throughout this paper since it is the only known error metric with a tight connection to plan quality [15, 20]. For convenience, Appendix A summarizes the most important facts.

A weaker error is introduced in [19]. It is motivated by the fact that if the estimate is 21 tuples, but the true value is 3 tuples, then the q-error is 7 and thus pretty high. An estimate e for some true value t is called θ, q-acceptable if either both $e \leq \theta$ and $t \leq \theta$ or the q-error of e is at most

q. It has been shown that a careful choice of θ prevents the query optimizer from making wrong decisions [19].

Given a set of estimates (e.g., for many different queries), the q-error is the maximum of all q-errors of each estimate, and the θ, q-error is the minimum q such that all estimates produced are θ, q-acceptable. Thereby, θ will be given.

3. THE START

The experiments reported are based on the forest data set[1], which is a relation with 54 attributes and $n = 581.012$ tuples. Since sampling is a random process, experiments with other datasets lead to similar results if sufficiently many and diverse queries are used. More on this can be found in Appendix B.

3.1 The Standard Estimator

For the *standard estimator* EST_S, the estimated selectivity is $|S_p|/|S| = k/m$ if $|S_p| \neq 0$. and $1/|S|$ if $|S_p| = 0$. This approach is often found in current systems.

3.2 Experiment 1

For 10 different numbers of range predicates ($z \in [2, 11]$), 50.000 conjunctive query predicates containing z range predicates were randomly generated. For a given z, a random conjunctive query predicate is generated by picking z attributes at random. For each randomly picked attribute, a random range predicate is then generated by randomly picking two numbers from the domain of the attribute. Then, the smaller number serves as the lower bound of the range predicate, and the higher number serves as the upper bound. A query is rejected if it yields the empty result. For each of these queries, samples of different sizes (1000, 2000, 4000, 8000) are used to produce estimates using EST_S. Thus, a total of 10*4*50000 = 2.000.000 estimates is produced. The following table gives the number of queries for which EST_S exhibits a q-error larger than 2.0[2], broken down by sample sizes:

m	#query
1000	202438
2000	152860
4000	118390
8000	91503

These numbers are out of a total of 500.000 queries for each sample size. We observe that the number of estimates with a high q-error decreases only very slowly with increasing sample sizes. Remember that 1000 is a typical sample size used in current systems.

Fig. 1 shows the maximum observed q-error for different numbers k of qualifying sample tuples. We added a horizontal line at a q-error of 2.0 to make it easier to distinguish between good and bad estimates. We observe that the largest q-errors are found for those queries having a small k. Especially the case $k = 0$, i.e., no sample tuple qualifies, leads to the largest q-error: the estimates can be orders of magnitudes away from the truth even for large sample sizes.

Beyond $|S_p| > 30$ or $|S_p| > 40$, sampling yields pretty precise estimates. Consequently, it is a good idea to have a

[1]http://kdd.ics.uci.edu/databases/covertype/covertype.html

[2]For an argument why 2.0 is a reasonable boundary for the q-error in order to distinguish good from bad estimates see [20] or Appendix A.

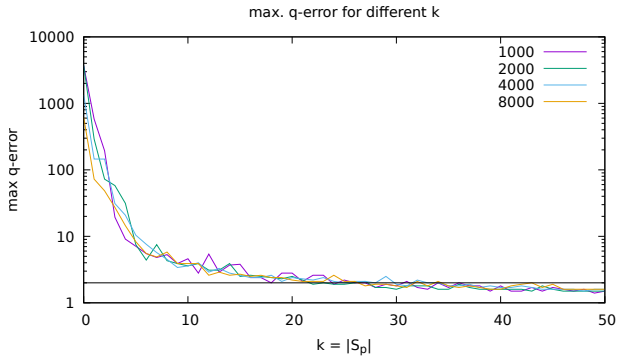

Figure 1: Forest: q-error vs. number of qualifying sample tuples ($k = |S_p|$)

large sample of say 10000 tuples and stop if 30 or 40 sample tuples qualify among the first 1000 sample tuples. (For a formula calculating the exact value see next section and Sec. 5.4.) Thus, depending on the query, it may be possible that only a fraction of the samples must be read. The large sample extracted just serves as a fallback and must not always be processed in total. It simply consumes memory and only consumes as much cpu time as is necessary for the query. Let us see how often k exceeds 30 or 40 (again, out of 500.000 queries per sample size):

m	$\#(\lvert S_p\rvert > 30)$	$\#(\lvert S_p\rvert > 40)$
1000	123735	109433
2000	156292	145671
4000	194909	177970
8000	233771	214777

Thus, for all these queries the predicate is evaluated on more sample tuples than necessary. Breaking early is not a new idea (e.g., [13, 12, 16]). However, the stopping critera used in these papers have no connection to the q-error. Our stopping criterion will exhibit a clear connection to the q-error and, thus, to plan quality.

However, even if the sample is large, still many queries will yield empty sample results:

m	$\#(\lvert S_p\rvert = 0)$
1000	167888
2000	127585
4000	99350
8000	79982

4. A SIMPLE FORMULA

Recall that

n	:=	$\lvert R\rvert$	number of tuples in relation
l	:=	$\lvert\sigma_p(R)\rvert$	number of qualifying tuples
m	:=	$\lvert S\rvert$	number of samples
k	:=	$\lvert\sigma_p(S)\rvert$	number of qualifying samples

Then, $k \leq l$, $k \leq m$, $l \leq n$, $m \leq n$, and $m - k \leq n - l$ must hold. Keeping this in mind, we can see that the total number of samples is $\binom{n}{m}$ and the number of samples for a given combination of l and k is $\binom{n-l}{m-k}\binom{l}{k}$. If each sample is equally likely, we can conclude the following:

	Sample A				Sample B				Formula			
k	α^*	ω^*	μ^*	ρ^*	α^*	ω^*	μ^*	ρ^*	α	ω	μ	ρ
0	1	3687	60.7	60.7	1	4158	64.5	64.5	1	6646	81.5	81.5
1	1	5520	74.3	74.3	1	5416	73.6	73.6	1	8170	90.4	90.4
2	6	7109	206.5	34.4	13	7832	319.1	24.5	4	9464	194.6	48.6
3	90	8114	854.6	9.5	52	8655	670.9	12.9	25	10642	515.8	20.6
4	256	8302	1457.8	5.7	124	8805	1044.9	8.4	77	11748	951.1	12.4
5	409	9643	1985.9	4.9	313	9436	1718.6	5.5	163	12801	1444.5	8.9
6	638	9792	2499.5	3.9	619	9998	2487.7	4	280	13815	1966.8	7
7	843	10310	2948.1	3.5	763	10585	2841.9	3.7	425	14798	2507.8	5.9
8	874	12441	3297.5	3.8	910	11000	3163.9	3.5	594	15754	3059.1	5.1
9	1358	12936	4191.3	3.1	1133	12279	3729.9	3.3	784	16690	3617.3	4.6
10	1255	14530	4270.3	3.4	1303	14671	4372.2	3.4	994	17607	4183.5	4.2
11	2318	13557	5605.8	2.4	1923	13111	5021.2	2.6	1219	18508	4749.9	3.9
12	1296*	16555	4632	3.6	2214	14512	5668.3	2.6	1460	19395	5321.3	3.6
13	2590	16775	6591.5	2.5	2199	17411	6187.6	2.8	1714	20270	5894.3	3.4
14	2183	15366	5791.7	2.7	2731	17993	7009.9	2.6	1980	21133	6468.6	3.3
15	2269	18237	6432.7	2.8	3137	16250	7139.8	2.3	2256	21987	7042.9	3.1
16	3806	16287	7873.3	2.1	3796	19474	8597.9	2.3	2543	22831	7619.7	3
17	4096	18619	8732.9	2.1	3817	18735	8456.4	2.2	2839	23667	8197	2.9
18	5225	19157	10004.8	1.9	3362	19321	8059.6	2.4	3143	24495	8774.3	2.8
19	3938	18835	8612.3	2.2	4610	20078	9620.8	2.1	3455	25316	9352.4	2.7
20	4111	19570	8969.5	2.2	5138	20134	10171	2	3774	26130	9930.5	2.6
21	5842	23280	11662	2	5380	21486	10751.5	2	4100	26938	10509.3	2.6
22	4941	22589	10564.7	2.1	5465	21916	10944	2	4433	27741	11089.4	2.5
23	5204	23015	10944	2.1	6472	22242	11997.9	1.9	4771	28538	11668.5	2.4
24	7164	24877	13349.9	1.9	6257	22920	11975.4	1.9	5114	29330	12247.2	2.4
25	6479	24075	12489.3	1.9	6891	24434	12975.9	1.9	5463	30117	12826.9	2.3
30	9474	27520	16147	1.7	9928	29302	17056.1	1.7	7277	33991	15727.4	2.2
35	12238	33012	20099.8	1.6	10944	31131	18458	1.7	9186	37778	18628.7	2
50	18931	41667	28085.5	1.5	16814	38659	25495.3	1.5	15324	48761	27335.2	1.8
99	47233	74548	59339.1	1.3	44733	71544	56571.9	1.3	37710	82480	55770.3	1.5

Table 2: Comparison of experimental and theoretical values for two different samples of size 1000

THEOREM 1. *Assume we are given a relation R and a query predicate p. Define $n := |R|$ and $l := |\sigma_p(R)_p|$. Then, for any sample $S \subseteq R$ of size $m := |S|$, the probability that for some k $k = |\sigma_p(S)|$ holds is*

$$\mathfrak{P}(n, m, k, l) := \frac{\binom{n-l}{m-k}\binom{l}{k}}{\binom{n}{m}}. \quad (1)$$

This is known as the hypergeometric distribution.

Let us now come to the major definitions of this paper. They will serve as a basis for all observations made further down and allow us to derive better estimators. For a given probability ϵ, define

$$\alpha(n, m, k) := \min\{l | \mathfrak{P}(n, m, l, k) \geq \epsilon\} \quad (2)$$

$$\omega(n, m, k) := \max\{l | \mathfrak{P}(n, m, l, k) \geq \epsilon\} \quad (3)$$

$$\mu(n, m, k) := \sqrt{\alpha(n, m, k) * \omega(n, m, k)} \quad (4)$$

$$\rho(n, m, k) := \sqrt{\alpha(n, m, k)^{-1} * \omega(n, m, k)} \quad (5)$$

$$\zeta_q(n, m) := \min\{k | \rho(n, m, k) \leq q\} \quad (6)$$

Assume n and m and ϵ are fixed. Then, for a given number k of qualifying sample tuples, α (ω) is the *smallest* (*largest*) number of tuples from R that has a probability larger than ϵ. Their geometric mean is denoted by μ. The q-error of μ with respect to α and ω is denoted by ρ. ζ_q denotes the minimum number of qualifying tuples needed to assure a maximal q-error less than q. Thus, ζ_q gives us the number of qualifying samples we should consider before we stop evaluating further samples (see Sec. 5.4 for details).

Their experimentally determined counterparts are denoted with an additional asterisk. Thus, α^* contains for some k the minimal l observed whereas ω^* contains for some k the maximal l observed.

Table 2 illustrates these definitions. For the forest relation with $n = 581012$ tuples and two different samples A and B, each of size $m = 1000$, it contains the observed and calculated (with $\epsilon = 10^{-5}$) values for α, ω, μ, ρ. Additionally, we can determine $\zeta_2 = 35$, whereas ζ_2^* seems to be 24 for Sample A and 22 for Sample B. Table 3 contains under otherwise equal conditions the results for a sample of size $m = 8000$. We observe that α^* and α and ω^* and ω are relatively close together. However, since sampling is a random process, it might happen that $\alpha^* < \alpha$ or $\omega^* > \omega$. One such case occurred for $k = 12$ and is marked by a '*' in Table 2. Further, note that the calculated and the observed values for ρ- and μ are close together. More on α^* and ω^*, also for a different dataset, can be found in Appendix B.

4.1 The Estimators Est_M and Est_T

Since α, ω, and μ seem to be close to reality, let us exploit them to derive two new estimators. The μ-estimator Est_M returns $\mu(n, m, k)$. If some θ is given, the estimates can be improved by returning an estimate of at least θ. The estimator Est_T returns $\max(\theta, \mu(n, m, k))$.

Since μ can (easily) be derived from α and ω, there is a need to efficiently calculate α and ω. Appendix C shows how this can be done.

4.2 Experiment 2

Table 4 shows experimental results using the estimators Est_S, Est_M, and Est_T. For the experiments, we have chosen a moderate $\theta = 1000$. As expected, there is an improvement

	Sample C				Formula			
k	α^*	ω^*	μ^*	ρ^*	α	ω	μ	ρ
0	1	602	24	24.5	0	830	28	28.8
1	1	787	28	28.1	1	1022	32	32
2	3	795	48	16.3	2	1185	48	24.3
3	8	946	87	10.9	4	1333	73	18.3
4	25	1091	165	6.6	11	1472	127	11.6
5	41	1267	227	5.6	22	1605	187	8.5
6	84	1623	369	4.4	37	1733	253	6.8
7	79	1605	356	4.5	56	1857	322	5.8
8	131	1531	447	3.4	77	1978	390	5.1
9	148	1675	497	3.4	101	2096	460	4.6
10	171	1765	549	3.2	127	2212	530	4.2
11	195	1860	602	3.1	156	2326	602	3.9
12	322	2040	810	2.5	186	2438	673	3.6
13	349	2083	852	2.4	218	2549	745	3.4
14	397	2045	901	2.3	251	2658	816	3.3
15	449	2288	1013	2.3	286	2766	889	3.1
16	475	2417	1071	2.3	321	2873	960	3.0
17	456	3134	1195	2.6	358	2979	1032	2.9
18	687	2782	1382	2.0	396	3084	1105	2.8
19	696	3367	1530	2.2	435	3188	1177	2.7
20	610	2904	1331	2.2	475	3292	1250	2.6
50	2506	6132	3920	1.6	1907	6173	3431	1.8
99	5434	9791	7294	1.3	4660	10498	6994	1.5

Table 3: Sample size 8000, comparison of experimental and theoretical values

of the q-error for the Est_S and Est_T. Especially for the very difficult case of no qualifying sample tuple ($k = 0$), we see that we can reduce the maximum θ,q-error from 3687 (Est_S) to 3.7 (Est_T).

5. SHARPENING INTUITION

5.1 Sample Size and Selectivity

Remember that ζ_q gives us the minimal number of qualifying sample tuples needed to guarantee a q-error of at most q. An obvious question is to ask for several relation sizes n and sample sizes m, how many qualifying sample tuples k we need to observe in order to be quite sure that the produced estimate $\mu(n, m, k)$ is precise up to a given q-error of, say, 2. Further, it might be interesting to look at the selectivity at this point. The latter can be determined by

$$\lambda_q(n, m) := \mu(n, m, \zeta_q(n, m))/n. \quad (7)$$

Table 5 contains ζ, μ, and λ values for different sample sizes and a fixed maximal q-error of 2.0. We observe that only for relatively large selectivities, sampling yields a maximal q-error of 2. Take the quite common sample size $m = 1000$ for example. There, we observe that we need a selectivity of at least 2% to guarantee a q-error of at most 2. If we double the sample size, selectivities higher than 1% can be estimated precisely. Note that these selectivities are almost independent of the relation sizes. Only for very small relations and very large sample fractions, λ_2 deviates significantly. Even for $m = 8000$ sample tuples only selectivities above 0.25 percent can be estimated precisely. Further, 0.25 percent of a large relation containing 10^9 or more tuples is a significant number.

These findings might be unsatisfactory and we might ask whether for a less strict error definition, where we essentially ignore q-errors for cases where both the true cardinality *and* the estimated cardinality are below a certain threshold, there is more hope to achieve good results with small samples. In order to answer this question, we use θ, q-acceptability and ask for the minimum sample size such that all estimates produced from sampling via μ are θ, q-acceptable. This minimum sample size can be determined by

$$\psi_{\theta,q}(n) := \min\{m | \forall k \rho(n, m, k) \leq q \vee \omega(n, m, k) \leq \theta\}. \quad (8)$$

The following table contains the size of the sample needed (in percent of $|R|$) to assure different q-errors for $\theta = 1000$:

	$\theta = 1000$						
$\|R\|$	2.0	3.0	4.0	5.0	6.0	7.0	8.0
20000	6.31	3.73	2.99	2.51	2.34	2.17	2.17
40000	6.35	3.76	3.01	2.53	2.36	2.19	2.19
80000	6.36	3.78	3.02	2.54	2.37	2.2	2.20
160000	6.37	3.78	3.03	2.55	2.48	2.48	2.48
320000	6.40	3.79	3.03	2.48	2.48	2.48	2.48
640000	6.40	3.79	3.03	2.48	2.48	2.48	2.48
1280000	6.40	3.79	3.03	2.48	2.48	2.48	2.48

First, we observe that the sample fraction for sufficiently large relations is almost a constant for a given q. Further, we see that if we want to guarantee, e.g., $1000, 2$-acceptability, we need a sample fraction n/m of about 6.5% of the relation.

Increasing θ to 10000 gives the following results:

	Est S		Est M		Est T	
	theta		theta		theta	
k	-	1000	-	1000	-	1000
0	3687.0	3687.0	81.5	45.2	1000.0	3.7
1	581.0	9.5	90.4	61.1	1000.0	5.5
2	193.7	193.7	36.5	36.5	166.7	7.1
3	19.4	19.4	15.7	15.7	11.1	8.1
4	9.1	9.1	8.7	8.7	8.3	8.3
5	7.1	7.1	6.7	6.7	6.7	6.7
6	5.5	5.5	5.0	5.0	5.0	5.0
7	4.8	4.8	4.1	4.1	4.1	4.1
8	5.3	5.3	4.1	4.1	4.1	4.1
9	3.9	3.9	3.6	3.6	3.6	3.6
10	4.6	4.6	3.5	3.5	3.5	3.5
11	2.8	2.8	2.9	2.9	2.9	2.9
12	5.4	5.4	4.1	4.1	4.1	4.1
13	2.9	2.9	2.8	2.8	2.8	2.8
14	3.7	3.7	3.0	3.0	3.0	3.0
15	3.8	3.8	3.1	3.1	3.1	3.1
16	2.4	2.4	2.1	2.1	2.1	2.1
17	2.4	2.4	2.3	2.3	2.3	2.3
18	2.0	2.0	2.2	2.2	2.2	2.2
19	2.8	2.8	2.4	2.4	2.4	2.4
20	2.8	2.8	2.4	2.4	2.4	2.4
21	2.1	2.1	2.2	2.2	2.2	2.2
22	2.6	2.6	2.2	2.2	2.2	2.2
23	2.6	2.6	2.2	2.2	2.2	2.2
24	1.9	1.9	2.0	2.0	2.0	2.0
25	2.2	2.2	2.0	2.0	2.0	2.0
26	2.0	2.0	1.8	1.8	1.8	1.8
27	2.0	2.0	2.0	2.0	2.0	2.0
28	1.7	1.7	1.8	1.8	1.8	1.8
29	1.9	1.9	1.8	1.8	1.8	1.8
30	1.8	1.8	1.7	1.7	1.7	1.7

Table 4: Q-error and θ, q-error of different Estimators

	1000			2000			4000			8000		
n	ζ_2	μ	λ_2	ζ_2	μ	λ_2	ζ_2	μ	λ_2	ζ_2	μ	λ_2
16000	35	318	0.0199	33	163	0.0102	28	87	0.0054	17	22	0.0014
32000	36	625	0.0195	35	316	0.0099	33	163	0.0051	28	87	0.0027
64000	36	1240	0.0194	36	622	0.0097	35	315	0.0049	33	163	0.0025
128000	37	2469	0.0193	37	1234	0.0096	36	620	0.0048	35	315	0.0025
256000	37	4928	0.0193	37	2457	0.0096	37	1231	0.0048	37	620	0.0024
512000	37	9846	0.0192	37	4904	0.0096	37	2451	0.0048	37	1229	0.0024
1024000	37	19683	0.0192	37	9799	0.0096	38	4893	0.0048	38	2448	0.0024

Table 5: ζ, μ, and λ

	$\theta = 10000$						
$\|R\|$	2.0	3.0	4.0	5.0	6.0	7.0	8.0
20000	0.48	0.3	0.24	0.21	0.19	0.18	0.16
40000	0.58	0.34	0.26	0.24	0.22	0.20	0.20
80000	0.64	0.37	0.29	0.26	0.23	0.21	0.21
160000	0.66	0.38	0.30	0.27	0.23	0.22	0.22
320000	0.67	0.38	0.30	0.27	0.24	0.24	0.22
640000	0.67	0.40	0.30	0.27	0.24	0.24	0.22
1280000	0.68	0.40	0.31	0.27	0.24	0.24	0.16

We observe that for $q = 2$, the sample fraction decreased from 6.5% to 0.65%. This might suggest that we should increase θ with the relation size. Let us look at the following table, which contains for a fixed maximal q-error of 2.0 and different θ the minimum sample size in percent of $|R|$ in order to guarantee θ, q-acceptability:

	θ			
$\|R\|$	1000	2000	4000	8000
20000	6.31	3.14	1.505	0.65
40000	6.3475	3.2425	1.605	0.7525
80000	6.3625	3.26	1.6263	0.805
160000	6.3725	3.3331	1.6688	0.8313
320000	6.3988	3.3394	1.6731	0.8356
640000	6.4014	3.3414	1.6755	0.8381
1280000	6.4024	3.3425	1.6766	0.8555

We observe again that the sample fraction for sufficiently large relations is almost a constant and decreases almost linearly with θ. However, remember that we cannot increase θ arbitrarily, since there are clear bounds on it which prevent the query optimizer from making wrong decisions.

5.2 k-Curves

Over a wide range of relation sizes and sample fractions, the maximum q-error mainly depends on k. Let us consider a fixed $k > 0$ and ask how the q-error decreases with increasing sample sizes m. Optimists might assume that it decreases linearly with the sample size. However, this is far from true. Consider Fig. 2. For three different $k = 4, 8, 28$ it shows on the x-axis the sample fraction m/n scaled by 1000. On the y-axis, it shows the maximal q-error ρ of the estimator μ. All curves start at a minimum sample size of 1000. We observe that below a sample fraction of 0.1% and relation sizes between 10^6 and 10^9, the maximum q-error is almost a constant. Only for larger sample fractions and only if k is small ($k = 4$ in this case), the q-error decreases significantly.

5.3 ω-0-Curve

In the previous subsection, we left out the case $k = 0$, which we consider now. Since $\alpha(n, m, 0) = 0$ for all n and m, we concentrate on ω. Fig. 3 contains what we call the ω-0-curve. It depicts for zero qualifying sample tuples $\omega(n, m, 0)$ on the y-axis for different relation sizes n and sample fractions n/m on the x-axis. Surprisingly, we observe that ω is independent of the relation size and slowly decreases with the sampling fraction. Thus, in order to guarantee an upper bound for ω at about 10.000 tuples, we need a sampling fraction of 0.1% of the relation.

5.4 Breaking Early

Assume we have a sample of size m and want to break early after having evaluated the query predicate on m' sample tuples with $m' < m$. Then, we can stop examining more sample tuples if $k \geq \zeta_2(n, m')$, where k denotes the number of qualifying sample tuples seen so far. In a real implementation, it might be useful to prematerialize a *break table*. It contains the values of $\zeta_2(n', m')$ for different relation sizes n' and early breaks m':

m'	n'				
	10^5	10^6	10^7	10^8	10^9
30	18	18	18	18	18
50	23	23	23	23	23
100	29	29	29	29	29
300	34	34	34	34	34
1000	36	37	37	37	37
3000	36	37	38	38	38
10000	34	37	38	38	38
30000	25	37	38	38	38

We observe that for a fixed m', $\zeta(n', m')$ is almost independent of the relation size. Further, for large m' it does not increase anymore. In fact, for small relation sizes it starts decreasing again. Note that although the $\zeta(n', m')$ look similar for $m' \geq 300$, they correspond to different selectivities for different m' (cf. Table 5). The break table can be used as follows. At the prespecified break points m', we compare the observed number k of qualifying sample tuples with the stored $\zeta_2(n', m')$ values for n' being the smallest value at least as large as the cardinality $n = |R|$ of the relation R for which the estimate is to be produced. If $k \geq \zeta_2(n', m')$, we can stop and produce the E_M estimate using $\mu(n, m', k)$.

We might ask the same question for the standard estimator: What is the minimal k such that the standard estimator leads to a q-error of less than q? The answer is given by η:

$$\eta_q(n, m) := \min\{k | k\frac{n}{m} \leq q * \alpha(n, m, k), \omega(n, m) \leq qk\frac{n}{m}\}.$$

The following table gives some values for η_2:

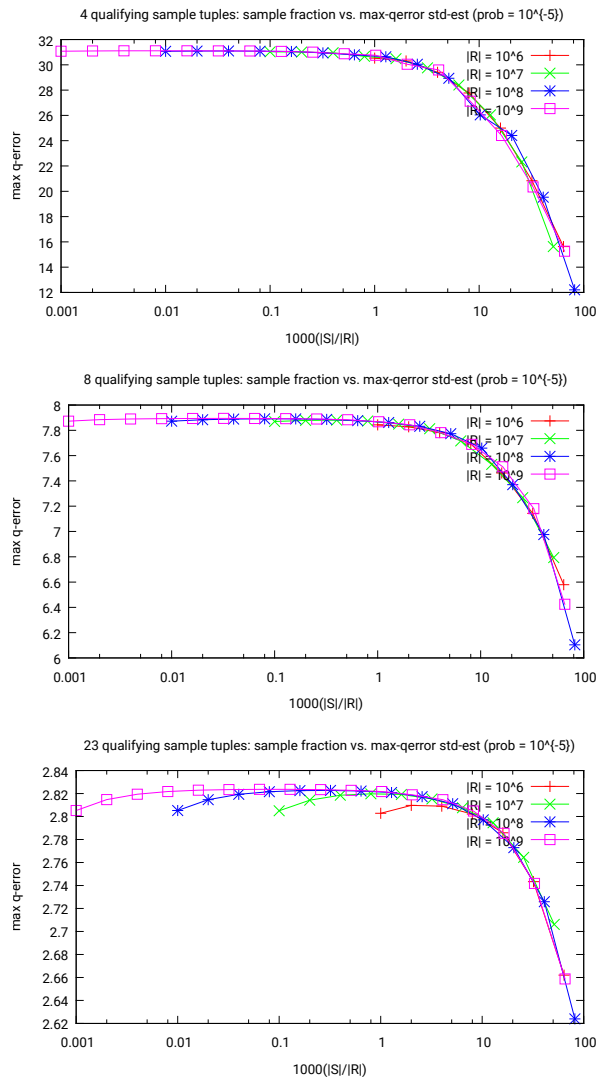

Figure 2: k-Curves

m'	$\lvert R\rvert$				
	10^5	10^6	10^7	10^8	10^9
30	24	24	24	24	24
50	30	30	30	30	30
100	36	36	36	36	36
300	42	42	42	42	42
1000	44	44	44	44	44
3000	44	45	45	45	45
10000	40	45	45	45	45
30000	29	44	45	45	45

As expected, η_q is larger than ζ_q. This means that if we use the standard estimator Est_S instead of Est_M, we need to observe more qualifying sample tuples and, consequently, a larger sample has to be inspected.

6. CONJUNCTIONS OF PREDICATES

Let $P = \{p_0, \ldots, p_{z-1}\}$ denote a set of z predicates. Assume we are given a query with a conjunction $p_0 \wedge \ldots \wedge p_{z-1}$

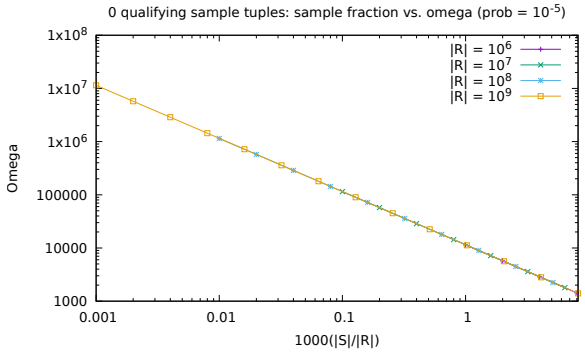

Figure 3: ω-0-Curve

of z predicates p_i. Then, the query optimizer during plan generation requires estimates not only for the full conjunct but also for conjuncts of subsets of the predicates in P [3, 15]. In this section, we consider the generation of estimates for all subsets of the predicates in P using sampling.

6.1 Preliminaries

For a subset of predicates $P' \subseteq P$, we denote by $F_\beta(P')$ the formula

$$F_\beta(P') = \bigwedge_{p_i \in P'} p_i.$$

Thus, $\beta(P')$ is simply a conjunction of all predicates contained in P'. By $F_\gamma(P')$ we denote the formula

$$F_\gamma(P') = \bigwedge_{p_i \in P'} p_i \wedge \bigwedge_{p_i \in P \setminus P'} \neg p_i.$$

This formula is a conjuction in which every predicate of P occurs exactly once, either positively (without $\neg$) or negatively (with $\neg$ preceding it). These formulas are called *minterms*.

Every subset $P' \subseteq P$ can be expressed as a bitvector $\text{bv}(P')$ of length $|P|$. Also, $\text{bv}(P')$ can be interpreted as a positive integer in the range $[0, 2^z - 1]$, which it represents. Subsequently, we will identify these different notations. Thus, we will use P' as an integer index into some vector. Define $n := 2^z$. For a given relation R (or sample S), we introduce two vectors $\beta \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}^n$, each containing exactly 2^z values by defining

$$\beta_{P'} := |\{r | r \in R, F_\beta(P')(r)\}|, \quad (9)$$
$$\gamma_{P'} := |\{r | r \in R, F_\gamma(P')(r)\}|. \quad (10)$$

Thus, $\beta_{P'}$ contains the number of qualifying tuples for a conjunction of the predicates in P', and $\gamma_{P'}$ contains the number of qualifying tuples for the minterm defined by P'. Note that the query optimizer needs estimates for $\beta_{P'}$.

Define the *complete design matrix* C as

$$C[i,j] = \begin{cases} 1 & \text{if } j \supseteq i \\ 0 & \text{else,} \end{cases}$$

where $j \supseteq i$ denotes the fact that every bit set to one in i is also set in j, i.e., $i = i \& j$ and i, j range from 0 to $2^z - 1$. Note that C is binary, non-singular, and persymmetric.

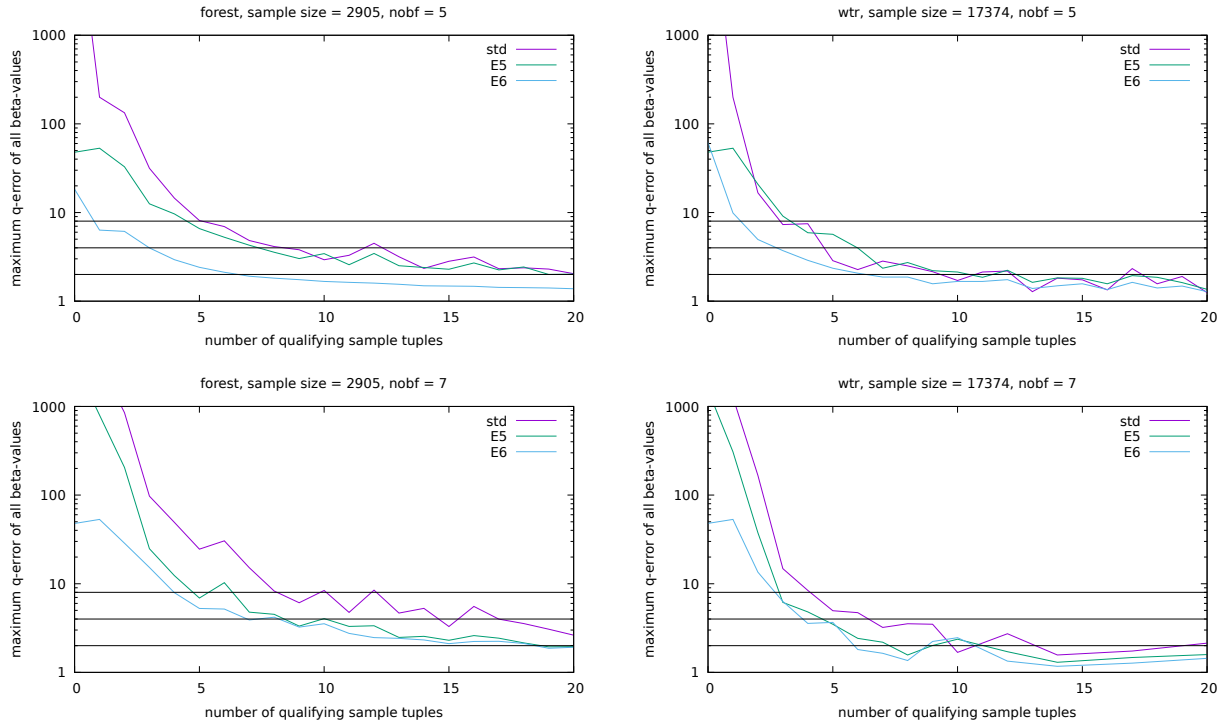

Figure 4: Max q-errors for different estimators for γ-sampling

The complete design matrix $C \in \mathbb{R}^{n,n}$ allows us to go from $\gamma \in \mathbb{R}^n$ to $\beta \in \mathbb{R}^n$ by

$$C\gamma = \beta. \tag{11}$$

Note that due to the recursive structure of C, a multiplication Cx can be calculated efficiently.

6.2 γ-Sampling

The following procedure `gamma-sampling` (proposed in [15]) takes as input a sample and a conjunction $P = p_1 \wedge \ldots \wedge p_z$ of simple predicates. Given an input sample S, for every of the 2^z possibilities

$$(\neg)p_1 \ldots (\neg)p_z$$

of either negating or not negating a simple predicate, it counts the number of occurrences within the sample S. These counts correspond the γ-values introduced in the previous subsection (Eqn. 10). Hence, the name of the sampling procedure.

```
gamma-sampling(P, z, S)
  // P is vector of predicates,
  // z its length,
  // S is the sample
  int n = (1 << z);
  // array of counters, all initialized to zero
  int c_gamma[n] = {0};
  for(s : S) // for all sample tuples in S
    int k = 0;
    for(int i = 0; i < z; ++i)
      // p[i](s): evaluate pi on sample tuple s
      k |= (p[i](s) << i);
    ++c_gamma[k];
  return c_gamma;
```

For every sample tuple $s \in S$, `gamma-sampling` evaluates each predicate p_i and keeps all these z results in a bitvector denoted by k. Interpreted as an integer, this z-bit bitvector k is the correct index for γ. The evaluation of some predicate p_i on some sample tuple $s \in S$ is denoted by $p[i](s)$. The result is either 0 or 1. Shifting this result by i and bitwise or-ing it with k stores this result in the i-th bit of k. Thus, after the inner loop, k contains a bitpattern representing the outcome of all predicates. Then, k is used as an index into an array of counters `c_gamma`, and the according field is increased. In formulas, we will denote the result vector `c_gamma` of `gamma-sampling` by γ^s.

6.3 Estimators

6.3.1 *Estimator* $\mathrm{Est}_{\mathrm{std}}$

The only estimator proposed so far in the literature [15] is the following:

$$\widehat{\beta_{\mathrm{std}}} := C(\max(1, \gamma^s)/|S|)|R|, \tag{12}$$

where γ^s equals `c_gamma` produced by `gamma-sampling`. Further, $|S|$ denotes the sample size, '/' and max are applied componentwise, and C denotes the complete design matrix. Let us call this estimator $\mathrm{Est}_{\mathrm{std}}$. It corresponds to an extension of Est_S to a vector of estimates.

Let us now extend α and ω to vectors by defining

$$\begin{aligned} \vec{\alpha}(x)[i] &:= \alpha(n, m, x[i]) \\ \vec{\omega}(x)[i] &:= \omega(n, m, x[i]) \end{aligned}$$

for $i = 0, \ldots, 2^z - 1$, where n is the relation size and m is the sample size. Additionally, we will need the componentwise geometric mean of these two bounds. Thus, we first define

for two vectors $x, y \in \mathbb{R}^n$ with $x \leq y$ and $0 < y$, the vector q-mid$(x, y) \in \mathbb{R}^n$ such that

$$\text{q-mid}(x, y)[i] := \sqrt{\max(1, x[i]) * y[i]}$$

for all $1 \leq i \leq n$. Then, for a given vector $x \in \mathbb{R}^n$ with $x \geq 0$, we define

$$\vec{\mu}(x) := \text{q-mid}(\vec{\alpha}(x), \vec{\omega}(x)).$$

6.3.2 Estimator E5

The idea of the first new estimator proposed here is to use the probabilistic boundaries described in Section 4 to produce probabilistic bounds for all components of γ. In order to minimize the q-error, we take the geometric mean of these bounds as a first approximation of our γ-estimate. Then, as it is known that the sum of all components of γ must equal the cardinality of the original relation/view, we scale the γ-estimates linearly and equally. The estimates $\widehat{\beta_{E5}}$ for β are then calculated as

$$\widehat{\beta_{E5}} := C * (s * \vec{\mu}(\gamma^s)), \tag{13}$$

where C is the complete design matrix and γ^s equals `c_gamma` produced by `gamma-sampling`.

The *scaling constant* s assures that the sum of the adjusted γ-estimates still equals the total cardinality of the relation or view R. It is thus defined as

$$s := \frac{|R|}{\sum_{i=1}^{n} \vec{\mu}(\gamma^s)[i]}.$$

6.3.3 Estimator E6

The idea of this new estimator is to use the bounds and geometric mean on the β-side. The estimates $\widehat{\beta_{E6}}$ for β are then calculated as

$$\widehat{\beta_{E6}} := \vec{\mu}(C\gamma^s), \tag{14}$$

where C is the complete design matrix and γ^s equals `c_gamma` produced by `gamma-sampling`.

6.4 Evaluation

Since many estimates are produced, we examine the maximum of all q-errors of all components in the estimate for β. An excerpt of our experiments can be found in Fig. 4. The figure shows experiments for the forest dataset and the wtr dataset (see Appendix B for details). Further, we consider $z = 5$ and $z = 7$ predicates (also called boolean factors). On the x-axis we find the number of qualifying sample tuples k and on the y-axis the maximum q-error of estimates $\beta[i]$ for $0 \leq i < 2^z$ over all queries. We see that both E5 and E6 are superior to Est_{std}. Further, E6 clearly outperforms E5. Still, for very small numbers of qualifying sample tuples ($k < 5$), the maximal q-errors are unacceptably high.

7. JOIN SELECTIVITY ESTIMATES

Consider two relations R and S. Assume both have an attribute a. Further, we assume that a is the key in R and a foreign key in S. The problem we consider is to produce an estimate for

$$|\sigma_{p_R}(R) \bowtie \sigma_{p_S}(S)|, \tag{15}$$

where p_R and p_S are selection predicates. We solve this problem by two well-known sampling techniques: correlated sampling (CS2) [21] and two-level sampling (TLS) [7]). We also take a look at *simple random sampling* (SRS) [1], which simply takes a random sample from the join. This can be done quite efficiently in the presence of indices [12, 18], even in case of many joins [6, 22].

Correlated sampling roughly works as follows. First, a random sample

$$V := \text{Sample}(\Pi_a^D(R) \cap \Pi_a^D(S)) \tag{16}$$

of the distinct values occurring in the join attributes is taken. Then, for both relations R and S those tuples with an a-value in D are selected to yield the samples for R and S:

$$\begin{aligned} \text{Sample}(R) &:= \sigma_{a \in V}(R), \\ \text{Sample}(S) &:= \sigma_{a \in V}(S). \end{aligned}$$

Note that there is no randomness applied here to select the tuples from R and S: simply all of them with an a-value in V are selected.

Two-level sampling deviates in the second step. Instead of selecting all tuples from R and S whose a-value occurs in V, one tuple of R and S is randomly drawn for each value in V, this tuple is called the *sentry*. Then, more tuples are drawn with a certain probability. For every distinct value $v \in D$, we denote by r_v the sentries of R and by s_v the sentries from S. Then, two-level sampling can be described as

$$\begin{aligned} r_v &\in \sigma_{a=v}(R) \text{ drawn randomly for all } v \in V, \\ s_v &\in \sigma_{a=v}(S) \text{ drawn randomly for all } v \in V, \\ \text{Sample}(R) &:= \text{Sample}(\sigma_{a \in V}(R) \setminus \{r_v | v \in V\}) \cup \{r_v | v \in V\}, \\ \text{Sample}(S) &:= \text{Sample}(\sigma_{a \in V}(S) \setminus \{s_v | v \in V\}) \cup \{s_v | v \in V\}. \end{aligned}$$

In both cases (CS2 and TLS), we are left with two samples Sample(R) and Sample(S) of R and S. The size of the join of these two samples is our sample size m, the size $|R \bowtie S|$ is our total size n, and the number of tuples in the join of the samples satisfying the predicates p_R and p_S gives our k:

$$\begin{aligned} n &= |R \bowtie S| && (17) \\ m &= |\text{Sample}(R) \bowtie \text{Sample}(S)| && (18) \\ k &= |\sigma_{p_R \wedge p_S}(\text{Sample}(R) \bowtie \text{Sample}(S))| && (19) \end{aligned}$$

Now, we can calculate $\alpha(n, m, k)$, $\omega(n, m, k)$, the estimate $\mu(n, m, k)$ and its maximum q-error $\rho(n, m, k)$.

To evaluate the precision of the estimator EST_M, which returns $\mu(n, m, k)$, for the join size problem (15), we conducted the following experiment. We generated two relations $R(a, b, c)$ and $S(a, d, e)$ with three attributes each. We fixed the size of R to 10^5 and the size of S to 10^6. The values are generated as follows:

- $R.a$ is the key of R and contains the numbers in $[0, |R|-1]$ exactly once.
- $S.a$ is a foreign key referencing $R.a$. These numbers are exponentially distributed with parameter λ and a possible shift s.
- $R.b$ and $S.d$ are exponentially distributed with parameter λ and a possible shift s in the range $[0 : 99999]$.
- $R.c$ and $S.e$ are copies of $R.a$ and $S.a$ resp.

For the exponential distribution, we use $\lambda \in \{1, 5, 10\}$. $\lambda = 1$ is mildly non-uniform, $\lambda = 5$ is pretty skewed and $\lambda = 10$ is highly skewed (see below). Since zero is always the most frequent number under exponential distribution, we added a *shift* s. A number for $R.b$, $S.a$, and $S.d$ is generated according to

$$\lfloor \texttt{rnd}_{exp,\lambda} * n + s \rfloor \mod n,$$

where $\texttt{rnd}_{exp,\lambda}$ generates exponentially distributed floating point numbers in $[0, 1]$ and $n = 100000$. We examined 19 shifts s between 0 and 900 in steps of 50. For any of these s, we derive the actual shifts for the attributes $R.b$, $S.a$, and $S.d$ by $s/2$, s, and $2 * s$, resp.

For our experiments, we use randomly generated range queries on b, c, d, and e. Thus, we produce estimates for

$$|\sigma_{b/c \in [b_{\text{lo}}, b_{\text{hi}}]}(R) \bowtie \sigma_{d/e \in [c_{\text{lo}}, c_{\text{hi}}]}|, \tag{20}$$

where b_{lo}, b_{hi}, c_{lo}, and c_{hi} are randomly chosen constants. For every relation instance, we generated 10000 random queries for each of the four attribute combinations (b, d), (b, e), (c, d), (c, e).

Both sampling methods were instructed to generate a sample such that $|\text{Sample}(R)| + |\text{Sample}(S)| \approx 0.01(|R| + |S|)$. The sizes of the join of the samples were on average 15585 for TLS, 19982 for CS2. For SRS we chose 16500 as the sample size. This corresponds to a sample fraction of 1.6% of $R \bowtie S$.

Before we get to the evaluation of the different sampling methods and estimators, let us take a brief look at the precision of the very simple estimator IDP:

$$(|\sigma_{p_R}(R)|/|R|) * (|\sigma_{p_S}(S)/|S|) * |R \bowtie S| \tag{21}$$

Thus, IDP builds on the independence of the selection predicates p_R and p_S. Its estimates have the following q-errors:

λ	max	avg
1	1.7	1.1
5	65.2	2.4
10	10902.9	36.6

The following table contains the theoretical q-errors (ρ-values) for SRS for $k = 0, \ldots, 7$:

k	0	1	2	3	4	5	6	7
ρ	26.3	29.2	22.2	16.7	11.1	8.4	6.7	5.7

The results of our experiments are shown in Tables 6, 7 and 8. Table 6 contains the maximum and average q-error for every estimator. Tables 7 and 8 show the maximum and average θ,q-errors for $\theta = 100$. The first column contains the parameter λ of the exponential distribution. The second column k contains the number of qualifying sample tuples. Under the *original* column, we find the aggregated q-errors of the original estimators, as proposed in the according papers. Under EST_M, we find the aggregated q-errors of the estimate $\mu(n, m, k)$. In Tables 7 and 8, we also find the aggregated q-errors of the estimator EST_T.

We would like to highlight the following findings from Table 6:

1. Comparing the maximal q-errors produced by SRS using EST_M, we find that they are very close to the theoretical q-error ρ given in the table above.
2. IDP is superior to all sampling methods for $\lambda = 1$. Even for $\lambda = 5$, it provides better or comparable estimates for $k = 0$ or $k = 1$.
3. Comparing the original estimators of CS2 and TLS, we find TLS being mostly superior to CS2 for $\lambda = 1$. For higher λ, there is no clear picture.
4. For SRS, the estimator EST_M is always superior to EST_S.
5. Compared to the original estimators for CS2 and TLS, EST_M can reduce the maximum q-error for $k = 0$ significantly. For larger k, there is no clear picture.
6. EST_M performs better on TLS than on CS2.

The latter can be explained by the fact that CS2 is not really a random sample of the join. This is also true for TLS, but to a lesser extent.

For the evaluation of the θ,q-errror, we used a small θ of 100. The maximum and average $100, q$-errors are shown in Tables 7 and 8. Observe that EST_T is in almost all cases the best estimator. Further, for $\lambda = 1$ and zero qualifying sample tuples ($k = 0$), it allows us the reduce the maximum $100, q$-error from 2538, 689, and 432 to 25.4, 6.9 and 4.3 in case of CS2, TLS, and SRS, resp. The reductions are as impressive for $\lambda = 5, 10$ and $k = 0$. The effect vanishes for larger k in case of CS2 and TLS. Further, for $k = 0$ and $\lambda = 1$, the *average* $100, q$-error can be reduced from 242, 181, 160 to 2.4, 1.8, 1.6 for CS2, TLS, SRS, resp. For $\lambda = 5, 10$, the results are similar.

8. CONCLUSION

We presented two new estimators Est_M and Est_T to more precisely estimate the result cardinality of selections and joins. Further, we hope that we sharpened the reader's intuition about the precision of estimates produced by sampling using either the standard or the new estimators.

An interesting question for future research is the following. Whereas TLS was successfully designed to minimize the variance of the estimates, it would be interesting to know whether there exists a sampling method that minimizes the q-error.

APPENDIX

A. WHY Q?

This section reviews some findings which motivate the usage of the q-error instead of other error metrics. We start by reviewing a theorem from [20]. We introduce the following abbreviation:

$$\|y\|_Q := \max\{y, 1/y\}.$$

Let $x > 0$ be a value and $\hat{x} > 0$ be an estimate for x. Then, the *q-error* of the estimate $\hat{x}$ is defined as

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

Note that the q-error is quite old (see, e.g., [5, 11]) and already found its way into textbooks (see, e.g., [8]).

Let $\mathcal{C}(e)$ denote the result of some cost function applied to some algebraic expression e, and let $\mathcal{M}(e)$ denote the

		max q-error						avg q-error					
		CS2		TLS		SRS		CS2		TLS		SRS	
λ	k	orig	EST_M	orig	EST_M	EST_S	EST_M	orig	EST_M	orig	EST_M	EST_S	EST_M
1	0	2538.0	105.8	689.0	27.3	432.0	26.3	79.3	4.8	70.4	4.1	56.5	3.9
1	1	25.0	47.0	46.7	32.2	61.0	29.2	2.7	4.3	4.3	4.9	2.4	4.1
1	2	20.0	26.8	26.6	23.6	20.2	19.9	2.1	4.1	2.7	4.7	1.9	4.0
1	3	50.0	21.9	17.2	19.7	6.7	11.9	1.9	3.5	2.1	4.3	1.7	3.5
1	4	12.5	18.9	9.3	12.7	8.1	7.3	1.7	2.9	1.8	3.3	1.6	2.7
1	5	14.7	13.9	12.8	8.4	5.0	6.8	1.6	2.4	1.7	2.6	1.5	2.2
1	6	10.0	10.8	7.8	7.7	3.8	5.6	1.5	2.1	1.6	2.3	1.4	2.0
1	7	10.3	8.3	6.4	6.6	3.5	4.7	1.5	1.9	1.5	2.1	1.4	1.8
5	0	2586.0	108.0	757.0	28.0	519.0	26.3	80.2	5.2	53.0	4.5	45.4	4.0
5	1	69.1	130.9	168.0	34.8	61.0	29.2	3.1	4.8	5.4	4.2	2.4	3.7
5	2	50.0	80.4	68.0	27.0	40.3	17.3	2.3	4.4	3.1	4.1	1.9	3.7
5	3	25.3	62.5	29.6	18.5	18.2	13.0	2.0	3.8	2.5	3.7	1.7	3.3
5	4	14.5	32.1	22.8	14.4	12.1	9.2	1.8	3.1	2.2	2.9	1.5	2.6
5	5	31.3	23.4	14.6	24.1	8.2	7.0	1.7	2.5	2.0	2.6	1.5	2.2
5	6	18.8	16.2	11.2	16.1	6.9	5.5	1.6	2.1	1.9	2.3	1.4	1.9
5	7	29.2	18.6	10.9	14.3	5.1	4.8	1.6	2.0	1.8	2.3	1.4	1.8
10	0	3795.0	159.8	1003.0	37.0	814.0	30.9	71.6	7.6	31.6	7.8	25.1	6.9
10	1	104.5	198.1	266.7	31.7	61.0	33.6	4.2	5.9	5.4	5.5	3.0	3.4
10	2	52.2	130.0	116.4	26.6	60.5	22.2	3.0	5.2	3.3	4.7	2.0	3.3
10	3	50.0	96.2	45.9	22.9	30.3	16.6	2.3	4.0	2.4	4.3	1.7	3.1
10	4	50.0	76.6	28.3	22.8	12.1	9.5	2.1	3.2	2.1	3.8	1.6	2.4
10	5	50.0	34.0	24.2	33.8	9.8	7.0	2.1	2.7	1.9	3.5	1.5	2.1
10	6	42.9	25.0	23.4	32.1	9.3	5.8	1.8	2.3	1.8	3.4	1.5	1.9
10	7	50.0	31.9	36.0	26.2	4.8	4.8	1.9	2.2	1.7	3.4	1.4	1.7

Table 6: Q-errors of different sampling approaches and estimators

		CS2			TLS			SRS		
λ	k	orig	EST_M	EST_T	orig	EST_M	EST_T	EST_S	EST_M	EST_T
1	0	2538.0	105.8	25.4	689.0	25.3	6.9	432.0	16.4	4.3
1	1	24.8	47.0	12.4	46.7	32.2	9.7	9.2	19.1	5.6
1	2	10.8	26.8	10.8	26.6	23.6	10.9	20.2	19.9	8.9
1	3	50.0	21.9	13.2	17.2	19.7	13.6	6.7	11.9	7.9
1	4	12.5	18.9	16.9	9.3	12.7	12.7	8.1	7.3	7.3
1	5	14.7	13.9	13.9	12.8	8.4	8.4	5.0	6.8	6.7
1	6	10.0	10.8	10.8	7.8	7.7	7.7	3.8	5.6	5.6
1	7	10.3	8.3	8.3	6.4	6.6	6.6	3.5	4.7	4.7
5	0	2586.0	108.0	25.9	757.0	28.0	7.6	519.0	19.7	5.2
5	1	69.1	130.9	34.5	168.0	34.8	10.4	10.6	22.2	6.5
5	2	32.4	80.4	32.4	68.0	27.0	12.3	40.3	17.3	7.7
5	3	25.3	62.5	38.0	29.6	18.5	12.7	18.2	13.0	8.7
5	4	14.5	32.1	29.0	22.8	14.4	14.4	12.1	9.2	9.2
5	5	31.3	23.4	23.4	14.6	24.1	24.1	8.2	7.0	7.0
5	6	18.8	16.2	16.2	11.2	16.1	16.1	6.9	5.5	5.5
5	7	29.2	18.6	18.6	10.9	14.3	14.3	5.1	4.8	4.8
10	0	3795.0	159.8	38.0	1003.0	37.0	10.0	814.0	30.9	8.1
10	1	104.5	198.1	52.3	266.7	31.7	9.5	16.1	33.6	9.8
10	2	52.2	130.0	52.2	116.4	26.6	12.2	60.5	19.7	8.8
10	3	50.0	96.2	58.0	45.9	20.9	14.4	30.3	16.6	11.1
10	4	50.0	76.6	68.7	28.3	22.8	22.8	12.1	9.5	9.5
10	5	50.0	34.0	34.0	24.2	33.8	33.8	9.8	7.0	7.0
10	6	42.9	25.0	25.0	23.4	32.1	32.1	9.3	5.8	5.8
10	7	50.0	31.9	31.9	36.0	26.2	26.2	4.8	4.8	4.8

Table 7: Maximum 100,q-errors of different sampling approaches and estimators

		CS2			TLS			SRS		
λ	k	orig	EST_M	EST_T	orig	EST_M	EST_T	EST_S	EST_M	EST_T
1	0	242.7	10.2	2.4	181.5	6.7	1.8	160.6	6.1	1.6
1	1	3.8	7.1	1.9	4.9	7.1	2.1	3.0	6.4	1.9
1	2	2.1	5.2	2.1	2.7	5.6	2.6	1.9	4.7	2.1
1	3	1.9	3.9	2.4	2.1	4.5	3.1	1.7	3.7	2.5
1	4	1.7	3.1	2.7	1.8	3.3	3.3	1.6	2.7	2.7
1	5	1.6	2.4	2.4	1.7	2.6	2.6	1.5	2.2	2.2
1	6	1.5	2.1	2.1	1.6	2.3	2.3	1.4	2.0	2.0
1	7	1.5	1.9	1.9	1.5	2.1	2.1	1.4	1.8	1.8
5	0	279.4	11.7	2.8	182.1	6.7	1.8	153.7	5.8	1.5
5	1	4.8	9.0	2.4	7.4	6.8	2.0	2.9	6.0	1.8
5	2	2.4	5.9	2.4	3.4	5.3	2.4	1.9	4.6	2.0
5	3	2.0	4.4	2.6	2.6	4.3	2.9	1.7	3.6	2.4
5	4	1.8	3.3	2.9	2.2	2.9	2.9	1.5	2.6	2.6
5	5	1.7	2.5	2.5	2.0	2.6	2.6	1.5	2.2	2.2
5	6	1.6	2.1	2.1	1.9	2.3	2.3	1.4	1.9	1.9
5	7	1.6	2.0	2.0	1.8	2.3	2.3	1.4	1.8	1.8
10	0	351.3	14.7	3.5	174.9	6.5	1.7	149.1	5.7	1.5
10	1	6.9	13.1	3.5	7.2	6.9	2.1	2.7	5.7	1.7
10	2	3.2	8.0	3.2	3.8	5.6	2.6	2.0	4.3	1.9
10	3	2.3	4.8	2.9	2.6	4.5	3.1	1.7	3.4	2.3
10	4	2.1	3.4	3.1	2.2	3.8	3.8	1.6	2.4	2.4
10	5	2.1	2.7	2.7	2.0	3.5	3.5	1.5	2.1	2.1
10	6	1.8	2.3	2.3	1.9	3.4	3.4	1.5	1.9	1.9
10	7	1.9	2.2	2.2	1.8	3.4	3.4	1.4	1.7	1.7

Table 8: Average 100,q-errors of different sampling approaches and estimators

true measured costs (e.g., runtime). Then, according to our definition, the q-error of the cost function $\mathcal{C}(e)$ is

$$\text{q-error}(\mathcal{C}(e)) = \|\mathcal{C}(e)/\mathcal{M}(e)\|_Q .$$

Let $\mathcal{E} = \{e_1, \ldots, e_k\}$ denote a set of plans. This set could be, for example, a set of plans equivalent to a given query and generated/explored by the plan generator. Further, let e_{opt} be the optimal plan for a query Q, minimizing $\mathcal{M}(e)$, and e_{best} the best plan, minimizing $\mathcal{C}(e)$. We are now interested in the factor by which the true costs of e_{best} are larger than the true costs of the optimal plan e_{opt}. An upper bound for this factor is given in the following theorem.

THEOREM A.1. *If for all* $e_i \in \mathcal{E}$

$$\|\mathcal{C}(e_i)/\mathcal{M}(e_i)\|_Q \leq q$$

for some q*, then*

$$\|\mathcal{M}(e_{best})/\mathcal{M}(e_{opt})\|_Q \leq q^2.$$

An important corollary to the theorem is:

COROLLARY A.2. *If for all* $e_i \in \mathcal{E}$

$$\|\mathcal{C}(e_i)/\mathcal{M}(e_i)\|_Q \leq q$$

for some q *and for all* $e_i \neq e_{opt}$

$$q < \sqrt{\|\mathcal{M}(e_i)/\mathcal{M}(e_{opt})\|_Q},$$

then

$$\mathcal{M}(e_{best}) = \mathcal{M}(e_{opt}).$$

That is, if we are able to make the q-error of our cost function small enough, then the corollary guarantees that we will find an optimal plan.

As input, cost functions take some cardinalities. In order to prove a result similar to Theorem A.1, the propagation of q-errors through cost functions has to be analyzed. This has been done in [20] for some cost functions, e.g., for hash joins. The results given there show that if the q-error of cardinality estimates is below q, then

$$\|\mathcal{M}(e_{best})/\mathcal{M}(e_{opt})\|_Q \leq q^4.$$

Here, we have to assume that the cost function is precise and errors only occur for the cardinality estimates. For details, an analogue to Corollary A.2 and more arguments, we refer the reader to [20]. For example, Ioannidis and Christodoulakis showed that errors propagate exponentially through joins [14]. While they use the relative error, a similar result can be shown for the q-error.

If both the cost function and the cardinality estimates contain errors, their q-errors multiply. Further note that the bounds given are tight. Comparing a quality loss of a factor of q^2 for cost function errors to the loss of a factor of q^4 for cardinality estimation errors leaves us to conclude that cardinality estimation errors are worse than cost function errors. The authors of [17] draw the same conclusion from their experiments.

B. COMPARING α^*, ω^* TO α, ω

Let R be a relation in $n = |R|$ tuples, $\mathcal{S} = \{S_1, \ldots, S_z\}$ a set of z samples of size $m = |S_j|$ and $P = \{p_1, \ldots, p_y\}$ a set of y predicates on R. Define $l_i := |\sigma_{p_i}(R)|$ as the number of tuples in R satisfying p_i and $k_{i,j} := |\sigma_{p_i}(S_j)|$ to be the

number of qualifying sample tuples from S_j for p_i. Then, the $k_{i,j}$ for any given i will be hypergeometrically distributed, independent of the dataset.

To show some experimental results illustrating this, consider Tables 9 and 10. The first column contains k and the last column the number of times $k = k_{i,j}$, i.e., the total number of observations of a number k of qualifying sample tuples. We define

$$\begin{aligned}\alpha^*(k) &:= \min_{i,j}\{l_i | k_{i,j} = k\},\\ \omega^*(k) &:= \max_{i,j}\{l_i | k_{i,j} = k\}.\end{aligned}$$

Further, α and ω are determined as in Eqns. 2 and 3 for $\epsilon = 10^{-5}$. Finally, $\#(l_i < \alpha)$ and $\#(l_i > \omega)$ denote the number of cases where l_i is below α or above ω. For the `forest` dataset, $n = 581012$, $m = 1000$, $z = 100$ different samples were considered for $y = 400000$ random queries. The `wtr` data set consists of the 7 attributes latitude, longitude and altitude of the weather station where the measurements took place, the day of the year when the measurement was taken, the minimum and maximum temperature as well as the precipitation measured. It was extracted from the weather data provided by `ftp.ncdc.noaa.gov` for the year 1962. The dataset consists of $n = 3474701$ tuples, we chose $m = 10000$ and generated $z = 100$ samples and $y = 50000$ random queries.

For both datasets, we observe that $\alpha^* \approx \alpha$ and $\omega^* \approx \omega$. Further, the number of cases where $l_i < \alpha$ or $l_i > \omega$ are rare.

C. FAST CALCULATION OF α AND ω

Since α and ω are used to produce better estimates, their fast calculation is vital. This is especially true if we need many estimates, as for E6. We start this part of the appendix by giving an exact method to calculate $\mathfrak{P}$ in Sec. C.1. This calculation allows us to calculate $\mathfrak{P}$ not only for positive integers but also for floating point numbers. In order to calculate $\alpha(n, m, k)$ and $\omega(n, m, k)$, we work on the natural logarithm of this continuous definition of $\mathfrak{P}$. Define

$$f(x) := \log(\mathfrak{P}(n, m, k, x)) - \log(\epsilon) \tag{22}$$

and x_0 and x_1 such that $f(x_i) = 0$, $x_0 < nk/m$, $x_1 > nk/m$. Then,

$$\begin{aligned}\alpha(n, m, k) &= \lceil x_0 \rceil,\\ \omega(n, m, k) &= \lfloor x_1 \rfloor.\end{aligned}$$

Thus, the problem can be reduced to finding the roots of f.

There exist many possibilities to find the root of a function. The simplest is bisection [9, p73]. Others include fixpoint calculation [9, p79], Newton's method [9, p78], Brent's algorithm [4], and TOMS Algorithm 748 by Alefeld, Potra, and Shi[2]. Here, we present the fixpoint calculation and Newton's method.

C.1 Calculation of $\mathfrak{P}$

$\mathfrak{P}$ can be evaluated using the `lgamma` function, which calculates the logarithm of the gamma function Γ. Define $\mathrm{F}(x) := \log(\Gamma(x + 1))$ (since $n! = \Gamma(n + 1)$). This has the advantage that $\mathfrak{P}$ can then also be evaluated for non-integer arguments. Using the definition of binomials, we have

$$\begin{aligned}\mathfrak{P}(n, m, l, k) &= \frac{\binom{n-l}{m-k}\binom{l}{k}}{\binom{n}{m}}\\ &= \frac{(n-m)!\,m!}{n!\,(m-k)!\,k!}\\ &\quad * \frac{(n-l)!\,l!}{((n-m)+(k-l))!\,(l-k)!}\end{aligned} \tag{23}$$

and, thus, we can calculate $\log(\mathfrak{P}(n, m, k, l))$ by

```
log_prob(n,m,k,l)
  if(0 == k)
    return −F(n − m − l) + F(n − l)
           +F(n − m) − F(n);
  if(l == k)
    return −F(m − k) + F(m) + F(n − k) − F(n);
  return −F(k) − F(m − k) + F(m)
         −F(l − k) + F(l) − F(n − m + k − l)
         +F(n − l) + F(n − m) − F(n);
```

The evaluation of `lgamma` needs approximately 18 ns, that of $\mathfrak{P}$ implemented using `lgamma` on average approximately 153 ns.

C.2 Approximation of $\mathfrak{P}$

Another way to evaluate F is to use Stirling's approximation of factorials:

$$n! \approx \sqrt{2\pi n}(\frac{n}{e})^n = \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}.$$

Replacing all factorials in Eqn. 23 by Stirling's approximation and applying the logarithm, we get after some simplifications:

$$\log(\mathfrak{P}(n, m, k, l)) \approx -m - \ln(d) + R + l * N, \tag{24}$$

where

$$\begin{aligned}d &:= \frac{n!\,(m-k)!\,k!}{(n-m)!\,m!} &(25)\\ &\approx \sqrt{2\pi} e^{-m} \frac{n^{n+\frac{1}{2}}(m-k)^{(m-k)+\frac{1}{2}} k^{k+\frac{1}{2}}}{(n-m)^{(n-m)+\frac{1}{2}} m^{m+\frac{1}{2}}},\\ R &:= +(n+\frac{1}{2})\ln(n-l) + \frac{1}{2}\ln(l)\\ &\quad -(n-m+k+\frac{1}{2})\ln(n-m+k-l)\\ &\quad +(k-\frac{1}{2})\ln(l-k), &(26)\\ N &:= \ln(n-m+k-l) - \ln(n-l) + \ln(l)\\ &\quad -\ln(l-k). &(27)\end{aligned}$$

Note that d is independent of l, and N and R contain only terms in which l occurs logarithmically. The evaluation of `log` requires approximately 4.8 ns. The evalution of Eqn. 24 requires about 147 ns. This is not much of a gain, but Eqn. 24 will be the basis for an efficient calculation of α and ω. For small k, $k!$ in d should be calculated exactly. Special cases are required for $k = 0$, $k = 1$, and $k = m$. Since we do not need α and ω for $k \geq \eta_2(n, m)$, we ignore the latter case here.

To treat the other two special cases, we define the selectivity $s := l/n$ and the sample fraction $t := m/n$. The rest of the appendix assumes that $1 - s - t \geq 0.9$, which assures that subsequent approximations are relatively precise. Since

k	α^*	ω^*	α	ω	$\#(l_i < \alpha)$	$\#(l_i > \omega)$	#obs
0	1	7251	0	6645	0	2	25337242
1	1	7706	1	8169	0	0	3913153
2	2	9540	4	9463	11	1	1742642
3	10	11088	25	10641	8	2	1067610
4	37	12577	77	11747	13	1	754422
5	124	12089	163	12800	7	0	574669
6	211	12782	280	13814	4	0	459945
7	309	14681	425	14797	9	0	379896
8	380	16217	594	15753	8	3	322437
9	613	16223	784	16689	4	0	277882
10	838	16052	994	17606	5	0	242548
11	990	18361	1219	18507	3	0	214886
12	1022	19536	1460	19394	2	1	191903
13	1467	20531	1714	20269	7	1	173017
14	1469	21831	1980	21132	1	2	158124
15	2313	20740	2256	21986	0	0	144492
16	2303	21894	2543	22830	3	0	132444
17	2402	22665	2839	23666	4	0	123376
18	3145	22864	3143	24494	0	0	114910
19	3216	24725	3455	25315	3	0	107034
20	3656	25051	3774	26129	1	0	99641

Table 9: Comparing α^*, ω^* to α, ω: forest

k	α^*	ω^*	α	ω	$\#(l_i < \alpha)$	$\#(l_i > \omega)$	#obs
0	1	3199	0	3992	0	0	4240748
1	1	4952	1	4911	0	1	178077
2	2	4544	3	5692	1	0	55821
3	29	5379	15	6405	0	0	31620
4	63	6519	47	7073	0	0	22428
5	113	7691	98	7710	0	0	16618
6	244	7296	168	8324	0	0	13832
7	346	7215	255	8918	0	0	11810
8	608	9359	356	9498	0	0	10386
9	494	8927	469	10065	0	0	9431
10	627	9017	594	10621	0	0	8654
11	730	9765	728	11167	0	0	7901
12	1005	10291	872	11705	0	0	7278
13	1245	10951	1023	12236	0	0	6699
14	1311	11109	1181	12760	0	0	6227
15	1490	11101	1346	13279	0	0	5665
16	1615	11714	1516	13791	0	0	5279
17	1990	13391	1692	14299	0	0	4913
18	1811	12794	1873	14803	1	0	4659
19	2705	13304	2058	15302	0	0	4240
20	2546	13431	2248	15797	0	0	3877

Table 10: Comparing α^*, ω^* to α, ω: wtr

the sample fraction is at most a low single digit percentage, and α and ω are only needed for low selectivities (cf. Table 5, λ-column), this is no restriction in practice.

Using s and t, we have the following two approximations for $k = 0$:

$$\log(\mathfrak{P}(n, m, 0, l)) \approx -stn, \quad (28)$$
$$\log(\mathfrak{P}(n, m, 0, l)) \approx -stn(1 + 0.5s), \quad (29)$$

where the latter approximation is a little bit more precise. Deriving these approximations is quite tedious, but the main idea is to use the approximations $\log(x) \approx (x-1)$ and $x\log(x) \approx x - 1$, where additionally $\log(x) \leq (x-1)$ and $x\log(x) \geq x - 1$ for $x \in [0.9, 1]$. We apply both approximations in turn to Eqn. 24 and then take the arithmetic mean of the two resulting approximations.

Using

$$\mathfrak{P}(n, m, 1, l) = \frac{lm}{(n - l - m + 1)}\mathfrak{P}(n, m, 0, l) \quad (30)$$

and Eqn 28, it is easy to show that

$$\log(\mathfrak{P}(n, m, 1, l)) \approx \log(n) + \log(t) + \log(s) - t + (1 - tn)s. \quad (31)$$

C.3 α, ω for $k = 0$

We have

$$\begin{aligned}
\alpha(n, m, 0) &= 0, \\
\omega(n, m, 0) &\approx -\log(\epsilon)\frac{n}{m}, \\
\omega(n, m, 0) &\approx (\sqrt{1 - 2\log(\epsilon)/(tn)} - 1)n,
\end{aligned}$$

where the first equation is obvious. The latter two can easily be derived using Eqn. 28 and Eqn. 29, resp.

C.4 α, ω for $k = 1$

To calculate α and ω for $k = 1$, we start by using Eqn. 31 and defining the following abbreviations:

$$\begin{aligned}
a &= 1 - tn = (1 - m), \\
b &= \log(\epsilon) - \log(n) - \log(t) + t.
\end{aligned}$$

Then, we consider the following series of equivalent equalities (in fact approximations only):

$$\begin{aligned}
\log(\mathfrak{P}(n, m, 1, l)) &= \epsilon, \\
\log(s) + as &= b, \\
s\exp(as) &= \exp(b), \\
as\exp(as) &= a\exp(b), \\
s &= \frac{W(a\exp(b))}{a},
\end{aligned}$$

where W is the Lambert W function. Finally,

$$\begin{aligned}
\alpha(n, m, 1) &\approx \lceil n\frac{W_0(a\exp(b))}{a}\rceil, \\
\omega(n, m, 1) &\approx \lfloor n\frac{W_{-1}(a\exp(b))}{a}\rfloor,
\end{aligned}$$

where W_0 and W_{-1} are the two branches of the Lambert W function. The Lambert W function can be implemented efficiently using Fukushima's excellent method [10], which executes in roughly 37 ns.

C.5 α, ω for $k > 1$: Fixpoint

We start by solving Eqn. 24 for l, which yields:

$$l = \frac{\ln(\epsilon) + m + \ln(d) - R}{N}. \quad (32)$$

Using this, it is easy to derive the following fixpoint procedure:

```
fixpoint(n, m, k, l_start)
  calculate d according Eqn. 25
  e = log(ε) + m + log(d)
  l = l_start
  do {
    l_old = l
    calculate R according to Eqn. 26
    calculate N according to Eqn. 27
    l = (e − R)/N
  } while(0.1 < |l_old − l|)
  return l;
```

C.6 α, ω for $k > 1$: Newton

Newton's method requires the derivative of f. We use an approximation of the derivative of f derived from plugging Eqn. 24 into the definition of f (Eqn. 22) for $\log(\mathfrak{P}(\cdot))$ and then calculating the derivative of this approximation of f. For convenience, we give this derivative here:

$$\begin{aligned}
f'(x) \approx\ & (k - m + n + 0.5)/(k - m + n - x) \\
& -x/(k - m + (n - x)) \\
& +\log(k) - m + (n - x)) \\
& +(k - 0.5)/(x - k) \\
& -x/(x - k) \\
& -\log(x - k) \\
& -(n + 0.5)/(n - x) \\
& +x/(n - x) \\
& -\log(n - x) \\
& +0.5/x + \log(x) + 1
\end{aligned}$$

Although Newton's method is well known, we give its pseudocode:

```
newton(n, m, k, l_start)
  l = l_start
  do {
    l = l − f(l)/f'(l)  // note:  f, f' need n, m, k, l
  } while(0.0001 < |l|)
  return l;
```

C.7 α, ω for $k > 1$: Initial Value

Both the fixpoint procedure and Newton's method need a start value. We use for α

$$l_{\text{start}} := \begin{cases}
\frac{8n}{11m}k & \text{if } m < 3 * k \\
0.4\frac{n}{m}k & \text{if } 50 < k \\
0.3\frac{n}{m}k & \text{if } 30 < k \\
0.2\frac{n}{m}k & \text{if } 20 < k \\
0.1\frac{n}{m}k & \text{if } 12 < k \\
\frac{n}{x_k m}k & \text{else,}
\end{cases}$$

where $x_k = 500, 80, 40, 20, 15, 10, 8, 7, 6, 6, 5$ for $k = 2, 3, \ldots, 12$.

For ω, we use

$$l_{\text{start}} := \begin{cases} \min(1.01 * \frac{n}{m}k, n-10) & \text{if } k > 0.95 * m \\ 1.04\frac{n}{m}k & \text{if } k > 0.9 * m \\ 1.06\frac{n}{m}k & \text{if } k > 0.8 * m \\ 1.09\frac{n}{m}k & \text{if } k > 0.7 * m \\ 1.13\frac{n}{m}k & \text{if } k > 0.6 * m \\ 1.17\frac{n}{m}k & \text{if } k > 0.4 * m \\ 1.20\frac{n}{m}k & \text{if } k > 0.3 * m \\ n/(m * (0.75/(10 + 1.7 * k))) & \text{else.} \end{cases}$$

C.8 Runtime

The following table gives the time to calculate α and ω via the three approaches discussed above.

	runtime [ns]		
	bisection	fixpoint	Newton
α	2149	216	229
ω	3731	524	372

We consider Newton's method to be the best. Since our implementation of Newton's method relies on approximations, one might wonder about the error: the q-error is always less than $1 + 7 * 10^{-5}$.

D. REFERENCES

[1] S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 275–286, 1999.

[2] G. Alefeld, F. Potra, and Y. Shi. Algorithm 748: Enclosing zeros of continuous functions. *ACM Trans. on Mathematical Software*, 21(3):327–344, 1995.

[3] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom. Adaptive ordering of pipelined stream filters. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 407–418, 2004.

[4] R. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14(4):422–425, 1971.

[5] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *Proc. ACM SIGMOD/SIGACT Conf. on Princ. of Database Syst. (PODS)*, pages 268–279, 2000.

[6] S. Chaudhuri, R. Motwani, and V. Narasayya. On random sampling over joins. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 263–274, 1999.

[7] Y. Chen and K. Yi. Two-level sampling for join size estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 759–774, 2017.

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

[9] L. Elden and L. Wittmayer-Koch. *Numerical Analysis*. Academic Press, 1990.

[10] T. Fukushima. Precise and fast computation of lambert w-functions without transcendental function evaluations. *J. Comp. Appl. Math.*, 244:77–89, 2013.

[11] P. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, pages 541–550, 2001.

[12] P. Haas, J. Naughton, S. Seshadri, and A. Swami. Fixed-precision estimation of join selectivity. In *Proc. ACM SIGMOD/SIGACT Conf. on Princ. of Database Syst. (PODS)*, pages 190–201, 1993.

[13] P. Haas and A. N. Swami. Sequential sampling procedures for query size estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 341–350, 1992.

[14] Y. E. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 268–277, 1991.

[15] F. Kastrati and G. Moerkotte. Optimization of conjunctive predicates for main memory column stores. *Proc. of the VLDB Endowment (PVLDB)*, 9(12):1125–1136, 2016.

[16] P.-Å. Larson, W. Lehner, J. Zhou, and P. Zabback. Cardinality estimation using sample views with quality assurance. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 175–186, 2007.

[17] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? *Proc. of the VLDB Endowment (PVLDB)*, 9(3), 2015.

[18] V. Leis, B. Radke, A. Gubichev, A. Kemper, and T. Neumann. Cardinality estimation done right: Index-based join sampling. In *CIDR*, 2017.

[19] G. Moerkotte, D. DeHaan, N. May, A. Nica, and A. Böhm. 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*, pages 361–372, 2014.

[20] G. Moerkotte, T. Neumann, and G. Steidl. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. of the VLDB Endowment (PVLDB)*, 2(1):982–993, 2009.

[21] F. Yu, W.-C. Hou, C. Luo, D. Che, and M. Zhu. CS2: A new database synopsis for query estimation. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 469–480, 2013.

[22] Z. Zhao, R. Christensen, F. Li, X. Hu, and K. Yi. Random sampling over joins revisited. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 1525–2539, 2019.