# Two-Level Sampling for Join Size Estimation

Yu Chen
Hong Kong University of Science and Technology
ychenbh@cse.ust.hk

Ke Yi*
Hong Kong University of Science and Technology
yike@cse.ust.hk

## ABSTRACT

Join size estimation is a critical step in query optimization, and has been extensively studied in the literature. Among the many techniques, sampling based approaches are particularly appealing, due to their ability to handle arbitrary selection predicates. In this paper, we propose a new sampling algorithm for join size estimation, called *two-level sampling*, which combines the advantages of three previous sampling methods while making further improvements. Both analytical and empirical comparisons show that the new algorithm outperforms all the previous algorithms on a variety of joins, including primary key-foreign key joins, many-to-many joins, and multi-table joins. The new sampling algorithm is also very easy to implement, requiring just one pass over the data. It only relies on some basic statistical information about the data, such as the $\ell_k$-norms and the heavy hitters.

## Keywords

Joins; sampling

## 1. INTRODUCTION

Join size estimation is a critical step in query optimization, as the intermediate result size is a dominating factor for the cost of a query plan, including CPU time, I/O cost, as well as communication cost in a distributed query processing engine. To be able to estimate join sizes quickly at query time, database systems reply on various statistics and small synopses collected from the base tables *a priori*. A major challenge here is that join queries often come with ad hoc selection predicates that are only known at query time, thus there is no way to take the predicates into account when the statistics are collected. To make things concrete, consider the (natural) join between two tables `Customers(CustomerID, Name, State)` and `Orders(OrderID, CustomerID, Quantity, Price, Comment)`. While it is easy to infer that the two tables will join on the attribute `CustomerID` beforehand (e.g., by looking for the foreign key constraint), which subset of customers and/or orders the query wants to perform the join on can only be known at runtime. The selection predicates can take a variety of forms, such as equality constraints, range constraints, complex conditions such as "`Comment LIKE '%complain%'`", or even user-defined functions.

*This work is supported by HKRGC under grants GRF-16211614 and GRF-16200415.



*SIGMOD '17, May 14–19, 2017, Chicago, IL, USA.*



DOI: http://dx.doi.org/10.1145/3035918.3035921

### 1.1 Related Work

There are three main approaches to the join size estimation problem in the literature.

Sketching based solutions [19, 3, 6, 8, 20] build a sketch for each table on the join attribute, while ignoring all the other attributes. Sketches give very accurate estimates on the join size without selection predicates, but the quality deteriorates rapidly when predicates are present. The way to handle a predicate, e.g., on the `State` attribute, is to turn the problem into a 3-table join, where the third table has a single column `State` containing all the values satisfying the predicate. Note that the third table is only imaginary, and its sketch is built at query time, as it depends on the predicate. However, as shown in [21], this makes the estimation drastically worse. Adding a second predicate, thus turning the problem into a 4-table join, makes the estimation completely useless (50 times off the true join size as reported in [21]). Moreover, building a sketch on the imaginary table only works when the attribute has a small domain so that we can enumerate all possible values satisfying the predicate to build the imaginary table. If the attribute has a large domain, then the predicate must be a range constraint so as to allow the use of range-summable hash functions [18]. It cannot handle predicates that cannot be expressed as a range condition, such as "`Comment LIKE '%complain%'`" or user-defined functions.

The second approach is to build a synopsis on the data in the multidimensional array defined by all the attributes, by adopting signal-processing techniques, such as wavelets, to do a lossy compression of the data [4]. However, this approach has the following drawbacks: (1) Similar to sketching based solutions, the predicate can only be equality or a range constraint as the compression is based on a hierarchical decomposition of the multi-dimensional space. (2) The effectiveness of compression can be very low on sparse data, which is particularly the case when attributes have large domains or the dimensionality is high (the experiments in [4] only used 4 dimensions with each dimension having a domain size at most 64). It is not clear how this approach

**Table 1: Notations**

| Notation | Definition |
|---|---|
| $A$, $B$ | set of all tuples in two tables |
| $A(v)$, $B(v)$ | set of tuples with join value $v$ |
| $a_v$, $b_v$ | number of tuples with join value $v$ |
| $a$, $b$ | the frequency vector $a = (a_1, a_2, \ldots, a_u)$, $b = (b_1, b_2, \ldots, b_u)$ |
| $S_A$, $S_B$ | set of all sampled tuples (excluding the sentry) |
| $S_A(v)$, $S_B(v)$ | set of all sampled tuples with join value $v$ (excluding the sentry) |
| $s_A(v)$, $s_B(v)$ | sentry of $v$ |
| $A^{c_A}$, $B^{c_B}$ | set of tuples in $A$ that satisfy $c_A$ or $c_B$, respectively |
| $J$ | join size of $A$ and $B$ |
| $J_v$ | join size $A(v)$ and $B(v)$ |
| $A^{c_A}(v)$, $B^{c_B}(v)$ | set of tuples with join value $v$ that satisfy $c_A$ or $c_B$, respectively |
| $a_v^{c_A}$, $b_v^{c_B}$ | number of tuples in $A(v)$ that satisfy $c_A$ or $c_B$, respectively |
| $S_A^{c_A}(v)$, $S_B^{c_B}(v)$ | set of sampled tuples with join value $v$ that satisfy $c_A$ or $c_B$, respectively (excluding the sentry) |
| $p_v$ | first level sampling probability for join value $v$ |
| $q$ | second level sampling probability |

would work at all if the attribute is real numbers (stored using finite precision) or strings, which have a very large domain.

The third approach, random sampling, is "insensitive" to the domain size, the form of the predicates, and how many of them there are. We can simply apply the predicates on the sample, and then perform join size estimation using only those tuples in the sample that satisfy the predicates. This work also studies the sampling approach to join size estimation. For a sampling based approach, one can consider collecting samples in an offline stage versus drawing sample online after a query is given. This paper focuses on the former, while Quickr [13] is a good example of the latter. The two approaches can be complementary to each other. The offline approach has very small overhead at query time, as estimation is done only on the sample. But it may suffer from large errors on highly selective queries, as few sampled tuples may satisfy the predicates. On the other hand, the online approach is more effective with selective predicates, as it can focus the sampling only on those tuples that pass the predicates. However, it incurs a higher cost at query time to retrieve tuples from the base tables, which may reside on disks on even on remote machines.

Before reviewing the existing sampling algorithms, we first formally define the problem.

## 1.2 Problem Definition

The most part of the paper considers the join between two relations $A$ and $B$; extensions to multiway joins are discussed in Section 4. Let $c_A$ and $c_B$ be the selection predicates on $A$ and $B$, respectively, and let $A^{c_A}$ and $B^{c_B}$ be the subsets of tuples in each relation satisfying the predicates. The goal is to estimate the join size $|A^{c_A} \bowtie B^{c_B}|$ at query time by using samples $S_A$ and $S_B$ gathered from $A$ and $B$ *a priori* without knowing the predicates $c_A, c_B$ beforehand. The notations used in this paper are listed in Table 1.

Assume that the join attribute[1] between the two tables take values from the domain $[u] = \{1, 2, \ldots, u\}$. For each $v \in [u]$, define $A(v)$ (resp. $B(v)$) as the set of tuples with join value $v$ in $A$ (resp. $B$), and let $a_v = |A(v)|$ and $b_v = |B(v)|$. We call $a = (a_1, a_2, \ldots, a_u)$ and $b = (b_1, b_2, \ldots, b_u)$ the *frequency vector* of $A$ and $B$. As will be clear later, our sampling algorithm does not rely on the domain being integers from 1 to $u$ at all. It will work as long as a hash function can be defined. This essentially covers all data types including strings and user-defined types. The vector representation offers some notational convenience. For instance, the join size is simply the inner product: $J = |A \bowtie B| = a \cdot b$. We also make heavy use of the $\ell_k$-norms, which is defined as $\|a\|_k = (\sum_{v \in [u]} a_v^k)^{1/k}$. In particular, $\|a\|_0$ is the number of distinct values (on the join attribute) in $A$, $\|a\|_1$ is simply the size of $A$, and $\|a\|_2^2$ is the self-join size of $A$.

Two types of joins are of particular importance. The first is where the join attribute is a primary key (PK) in one table and a foreign key (FK) in the other, such as "`CustomerID`" in the earlier example. For PK-FK joins, one of $a$ and $b$ is an all-1 vector. The second type is the more general many-many joins where both $a_v$ and $b_v$ can be large. Consider for example the "follows" relation in Twitter: `Follows(Follower, Followee)`. A join between this table and its (logical) copy can find us all the 2-hop "follows" relationships. As more complex graph pattern queries make use of these intermediate results, their accurate estimation is critical in choosing the optimal execution plan. It is worth mentioning that some recent theoretical developments have suggested that in some cases, none of the pairwise join plans is optimal due to unavoidable large intermediate result size, and new join algorithms have been proposed [5]. However, these new algorithms are not always better (they are only optimal in the worst case), thus it is important to be able to determine at run time if this is the case.

## 1.3 Previous Sampling Algorithms

### *Independent Bernoulli sampling.*

The simplest sampling algorithm is independent Bernoulli sampling, that is, each tuple is sampled independently with probability $p$. To estimate the join size $J = |A \bowtie B|$, we simply compute the join size of the sample and scale it up by $1/p^2$, i.e., $\hat{J}_{Ber} = \frac{|S_A \bowtie S_B|}{p^2}$. This is unbiased because each

[1]We assume that the join attribute name is given beforehand, which can often be inferred from the foreign key constraints.

pair of joined tuples appear in the join of the sample iff both tuples are sampled.

The variance is [21]:

$$\mathrm{Var}[\hat{J}_{Ber}] = \sum_v a_v b_v \left[ \left( \frac{1}{p^2} - 1 \right) + (a_v - 1) \left( \frac{1}{p} - 1 \right) + (b_v - 1) \left( \frac{1}{p} - 1 \right) \right]$$

Selection predicates can be easily handled by applying the predicates on the sample before making the estimation.

The Aqua system [1] does independent Bernoulli sampling only on the largest fact table, then take all the joining tuples from other tables. It thus only works for PK-FK joins and all such PK-FK relationships must form a directed acyclic graph. In addition, since the tuples sampled from a PK table depend on those sampled in a FK table, the tables have to be sampled sequentially. On the other hand, all the other sampling algorithms covered in this paper, including ours, sample all tables in parallel.

### *Correlated sampling.*

Independent sampling essentially ignores the join relationship between the two tables. To address this drawback, *correlated sampling* [21] uses hash function to generate correlated samples. Let $h : [u] \to [0, 1]$ be a random hash function. It takes every tuple with join attribute value $v$ into the sample if $h(v) < p$. Since every pair of joined tuples share the same join attribute value, with probability $p$, both tuples appear in the sample, and with probability $1 - p$, neither is sampled. Thus, the unbiased estimator for join size is $\hat{J}_{Cor} = \frac{|S_A \bowtie S_B|}{p}$.

The variance for correlated sampling is [21]:

$$\mathrm{Var}[\hat{J}_{Cor}] = \left( \frac{1}{p} - 1 \right) \sum_v a_v^2 b_v^2.$$

Note that both sampling algorithms have the same (expected) sample size $p(\|a\|_1 + \|b\|_1)$. However, there is no definitive relationship between $\mathrm{Var}[\hat{J}_{Ber}]$ and $\mathrm{Var}[\hat{J}_{Cor}]$. In one extreme case (Figure 1(a)), when the join is a one-to-one mapping between the two tables, independent Bernoulli sampling has a variance of (roughly) $\|a\|_1/p^2$, while the variance of correlated sampling is $\|a\|_1/p$. This is because independent sampling has trouble matching the tuples, while correlated sampling makes sure that any pair of matched tuples are always sampled together. In the other extreme case (Figure 1(b)), when all tuples share the same join value, i.e., the join becomes a Cartesian product of the two tables, correlated sampling either samples all data (with probability $p$) or samples nothing, resulting in a variance of $\|a\|_1^2\|b\|_1^2/p$. While in this case, independent Bernoulli sampling has variance $\|a\|_1\|b\|_1/p^2$, which is always smaller than the former, since we must have $p \geq 1/(\|a\|_1 + \|b\|_1)$ for the sampling to make sense. In the earlier studies on correlated sampling [21], experiments were only performed on PK-FK key joins, which is more towards the first extreme case, thereby drawing the conclusion that correlated sampling is better than independent Bernoulli sampling. However, in our experiments on many-to-many joins on Twitter data, independent Bernoulli sampling actually performs much better than correlated sampling, due to many high-degree tuples, which push the problem more towards the second extreme case.

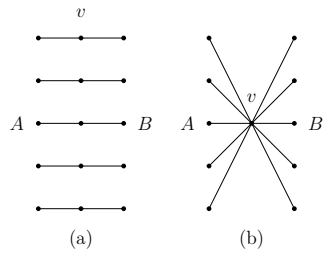


**Figure 1: Two extreme cases: (a) correlated sampling is better; (b) independent Bernoulli sampling is better.**

### *End-biased sampling.*

End-biased sampling [10] is similar to correlated sampling, also using a hash function $h$ to perform the sampling. It can be seen as the "frequency-aware" version of correlated sampling. It uses the frequency information, i.e., the frequency vectors $a$ and $b$, to favor join values that are heavy. It adopts the natural choice that the sampling probability of a join value $v$ should be linearly proportional to its frequency. More precisely, let $K_A$ and $K_B$ be two parameters set by the user. Each tuple in $A$ with join value $v$ is sampled if $h(v) < \frac{a_v}{K_A}$, and each tuple in $B$ with join attribute $v$ is sampled if $h(v) < \frac{b_v}{K_B}$. Note that a pair of joined tuples are sampled with probability $p_{\min}(v) = \min(\frac{a_v}{K_A}, \frac{b_v}{K_B}, 1)$, so the join size estimator is the same as that for correlated sampling by just replacing $p$ with $p_{\min}(v)$, i.e., $\hat{J} = \sum_v \frac{|S_A(v) \bowtie S_B(v)|}{p_{\min}(v)}$. And the variance is

$$\mathrm{Var}[\hat{J}_{end}] = \sum_v \left( \frac{1}{p_{\min}(v)} - 1 \right) a_v^2 b_v^2.$$

### *Index-assisted sampling.*

There are also many sampling algorithms that rely on indexes, including adaptive sampling [16], bifocal sampling [11], and wander join [15]. The use of indexes allows the sampling to be much focused, retrieving only tuples that are relevant to the query. However, indexes are not always available; even if they are, doing repeated index lookups can be expensive when the index does not fit into memory.

All sampling algorithms examined in this paper, including ours, perform the sampling by only making one pass over the data, while only relying on some basic statistics of the data. In fact, these sampling methods are complementary to index-assisted sampling for dealing with very large data set: We can first use these one-pass algorithms to draw a sample that fits into memory, build indexes on the sample, and then run index-assisted sampling algorithms on the sample. In essence, these one-pass sampling algorithms aim at reducing the space complexity (original data is not needed when making estimation), while index-assisted sampling aims at reducing the running time of the estimation by making a guided search in a potentially large space.

## 1.4 Our Contributions

Our new sampling algorithm, called *two-level sampling*, actually builds upon the three sampling algorithms described above, by combining their advantages in the "right" way. The observation starts exactly from the two extreme cases in Figure 1. We notice that correlated sampling is good

at identifying tuples from the two tables sharing the same join value, but when dealing with tuples sharing the same join attribute from the same table, it is just "all or nothing". On the other hand, independent Bernoulli sampling handles tuples from the same table well, but fails to capture the correlation across the tables. In our two-level sampling algorithm, we sample the join values in a correlated way, and use independent Bernoulli to sample tuples sharing the same join values. When the frequencies $a_v, b_v$'s are available or approximately available, we adopt the idea of end-biased sampling to adjust sampling probabilities to favor more frequent join values. But interestingly enough, our analysis shows that the natural choice of setting the sampling probability linear in the frequency is actually not optimal. There is a more delicate relationship between the optimal sampling probability and the frequency.

We describe the algorithm more formally in Section 2, and show how to implement the algorithm in one pass over the data. We show that, for both PK-FK joins and many-to-many joins, two-level sampling achieves a smaller variance than all three sampling algorithms above. The comparison is done both analytically and empirically, using TPC-H benchmark data and Twitter data.

We also extend the basic two-level sampling algorithm in the following aspects.

1. An important requirement of estimation techniques is that, in addition to the estimate itself, the estimation procedure should also return a confidence interval, so as to give users an idea how accurate the estimate is. In Section 3, we give methods to compute the confidence interval, and validate their effectiveness by experiments.

2. Note that the join size is the same as applying a `COUNT` aggregation on the join results. In Appendix B, we extend the two-level sampling algorithm to estimate other aggregates such as `SUM` and `AVG`. Thus, the algorithm can be used in a general-purpose database system for approximate query processing by sampling, such as BlinkDB [2], which currently does not support joins.

3. In Section 4, we extend the algorithm to two most common multi-table joins: chain joins and star joins.

Finally, experimental results are presented in Section 5 before concluding the paper.

# 2. TWO-LEVEL SAMPLING

## 2.1 Sampling Algorithm

Let $h : [u] \to [0, 1]$ be a random hash function. Let $p_v$ be the level-one sampling probability for join value $v$. If the join value frequencies are available (or approximately available), $p_v$ could be a function of $a_v$ and $b_v$; otherwise, the $p_v$'s can only be equal, which is simply written as $p$. The level-two sampling probability, $q$, will always be the same for all tuples. Later we will discuss how to set $p_v$ and $q$ appropriately.

Let $A(v)$ be all the tuples in $A$ with join value $v$. If $h(v) < p_v$, one tuple in $A(v)$ is chosen uniformly at random. This tuple is called the *sentry* of $v$ in $A$, denoted as $s_A(v)$. The rationale here is that, if $v$ is sampled, at least one tuple with this join value should be sampled so as to join with the tuples from the other table. Then, each of the other tuples in $A(v)$ are sampled independently with probability $q$. We use $S_A(v)$ to denote the set of tuples sampled, excluding the sentry. Set $S_A = \bigcup_v S_A(v)$. We exclude the sentries from $S_A$ just for notional convenience; in practice, the sentries are stored together with $S_A$, but with a special bit indicating its sentry status.

Exactly the same is done on table $B$. Correspondingly, $s_B(v)$ is the sentry for join value $v$ in $B(v)$, and $S_B = \bigcup_v S_B(v)$ consists all the tuples sampled from $B$, excluding the sentries.

The sampling process can be easily performed in one pass over the data, as described in the algorithm below.

Two-level sampling($A$)

```
1  for every tuple t in A do
2      if h(t.v) < p_v then
3          if v has never been sampled then
4              s_A(v) ← t
5              c_v ← 1
6          else
7              c_v ← c_v + 1
8              set s_A(v) ← t with probability 1/c_v
9          Sample t with probability q into S_A(t.v)
10 Exclude s_A(v) from S_A(v) for each v
```

Note that line 3–8 essentially uses reservoir sampling [22] to pick the sentry for each value $v$. The counter $c_v$ records the number of tuples with join value $v$ that have been seen, so as to sample the sentry with the correct probability.

We also note that this sampling algorithm is "embarrassingly parallel". The table can be arbitrarily partitioned and we can run the algorithm on each partition. The tuples sampled in level two can be trivially combined. For each join value $v$, each partition will return a sentry. To combine them into one, we just need choose one randomly, using probabilities proportional to $a_v^p$, where $a_v^p$ denotes the frequency of join value $v$ in partition $p$. Note that $a_v^p$ can be easily computed as the sentry is sampled from each partition $p$.

## 2.2 The Estimator

Let $J_v = a_v b_v$. As long as we have an unbiased estimator $\hat{J}_v$ for $J_v$, the join size can be estimated as $\hat{J}_{2lvl} = \sum_v \hat{J}_v$.

An unbiased estimator for $J_v$ is

$$\hat{J}_v = \begin{cases} \frac{1}{p_v}\left(\frac{|S_A(v)|}{q} + 1\right)\left(\frac{|S_B(v)|}{q} + 1\right), \\ \qquad\qquad \text{if } v \text{ is sampled for } A \text{ and } B; \\ 0, \qquad \text{otherwise.} \end{cases}$$

We give the proof of unbiasedness in Appendix A.1. Intuitively, the "1" stands for the sentry, and $\frac{|S_A(v)|}{q}$ estimates $a_v - 1$, as every tuple (excluding the sentry) is sampled with probability $q$. But this is all conditioned upon $v$ being sampled in the first level, which happens with probability $p_v$, so we scale up the whole thing up by $1/p_v$.

The derivation of the variance is a bit more involved than that for independent Bernoulli and correlated sampling, due to the two-step sampling process. The basic idea is to express the sampling process by two random variables, one

indicating whether $v$ is sampled at all while the other capturing $|S_A(v)|$ and $|S_B(v)|$. Note that conditioned upon $v$ being sampled in the first level, $|S_A(v)|$ and $|S_B(v)|$ follow a binomial distribution. Then we can use the law of total variance to combine the variances of the two levels. The detailed derivation is given in Appendix A.2, which yields

$$\mathrm{Var}[\hat{J}_{2lvl}] = \sum_{v:a_v b_v \neq 0} \mathrm{Var}[\hat{J}_v] = \sum_{v:a_v b_v \neq 0} \left( \frac{1}{p_v}\sigma_1^2(v) + \sigma_2^2(v) \right), \tag{1}$$

where

$$\begin{aligned}\sigma_1^2(v) = &\left(\frac{1}{q^2} - 1\right)(a_v - 1)(b_v - 1) \\ &+ \left(\frac{1}{q} - 1\right)(b_v - 1)(a_v^2 - a_v + 1) \\ &+ \left(\frac{1}{q} - 1\right)(a_v - 1)(b_v^2 - b_v + 1),\end{aligned}$$

and

$$\sigma_2^2(v) = \left(\frac{1}{p_v} - 1\right) a_v^2 b_v^2.$$

When selection predicates $c_A$ and $c_B$ are present, we simply apply the predicate on the sample before making the estimation. More precisely, let $S_A^{c_A}(v)$ and $S_B^{c_B}(v)$ be the set of sampled tuples in $S_A(v)$ and $S_B(v)$ satisfying the respective predicates. Let $I_A^{c_A}(v)$ (resp. $I_B^{c_B}(v)$) be 1 if the sentry $s_A(v)$ (resp. $s_B(v)$) satisfies $c_A$ (resp. $c_B$) and 0 otherwise. The estimator is still $\hat{J}_{2lvl} = \sum_v \hat{J}_v$, but $\hat{J}_v$ now becomes:

$$\hat{J}_v = \begin{cases} \frac{1}{p_v}\left(\frac{|S_A^{c_A}(v)|}{q} + I_A^{c_A}(v)\right)\left(\frac{|S_B^{c_B}(v)|}{q} + I_B^{c_B}(v)\right), \\ \qquad\qquad \text{if } v \text{ is sampled for } A \text{ and } B; \\ 0, \qquad \text{otherwise.} \end{cases}$$

We prove that this estimator is unbiased Appendix A.3.

The variance retains the same form as in (1), but $\sigma_1^2(v)$ and $\sigma_2^2(v)$ now become

$$\begin{aligned}\sigma_1^2(v) = &\left(\frac{1}{q^2} - 1\right)(a_v^{c_A} - \frac{a_v^{c_A}}{a_v})(b_v^{c_B} - \frac{b_v^{c_B}}{b_v}) + \\ &\left(\frac{1}{q} - 1\right)(b_v^{c_B} - \frac{b_v^{c_B}}{b_v})\left((a_v^{c_A})^2 - a_v^{c_A} + \frac{a_v^{c_A}}{a_v}\right) + \\ &\left(\frac{1}{q} - 1\right)(a_v^{c_A} - \frac{a_v^{c_A}}{a_v})\left((b_v^{c_B})^2 - b_v^{c_B} + \frac{b_v^{c_B}}{b_v}\right),\end{aligned}$$

and

$$\sigma_2^2(v) = \left(\frac{1}{p_v} - 1\right)(a_v^{c_A})^2 (b_v^{c_B})^2,$$

where $a_v^{c_A} = |A^{c_A}(v)|$, $b_v^{c_B} = |B^{c_B}(v)|$. The detailed derivation can be found in Appendix A.4. Since $a_v^{c_A} \leq a_v, b_v^{c_B} \leq b_v$, it is obvious that the variance can only become smaller under selection predicates. As pointed out in [21], this is a major advantage of sampling based approaches.

In the rest of this section, we discuss how to set the parameters $p_v$ and $q$, depending on the join type and whether frequency information is available. We also make analytical comparisons with previous sampling algorithms, identifying the quantitative improvement offered by two-level sampling. We realize that this is essentially an optimization problem where the goal is to minimize $\mathrm{Var}[\hat{J}_{2lvl}]$ under a given sample size constraint. However, $\mathrm{Var}[\hat{J}_{2lvl}]$ is different for different predicates, and what is worse, the $a_v^{c_A}, b_v^{c_B}$'s are not known until query time, while we need to do the sampling beforehand. Therefore, we decide to set the sampling parameters to minimize $\mathrm{Var}[\hat{J}_{2lvl}]$ without predicates. Essentially, we optimize for the worst case; when predicates are present, the variance can only be smaller.

## 2.3 PK-FK Joins, Frequencies Unknown

Without loss of generality, assume that the join attribute is a primary key in $B$ and a foreign key in $A$, so $b_v = 1$ for all $v$. When the frequencies are unknown, we set all the $p_v$ to $p$. Note that the objective that we want to minimize, $\mathrm{Var}[\hat{J}_{2lvl}]$, is a function of the $a_v$'s and $b_v$'s, as well as $p$ and $q$. But it turns out to minimize $\mathrm{Var}[\hat{J}_{2lvl}]$, we do not need to know all the frequencies, but only some aggregate statistics of the frequency vector. Specifically, we assume that we know the distinct count of $a$, i.e., $\|a\|_0$, and an AMS sketch [9] of $a$ so that we know $\|a\|_2$ (approximately). If we are deprived of even these basic statistics, then the problem is hopeless since we know nothing about the data at all.

Let $n$ be the desired sample size. With $p_v = p$ and $b_v = 1$ for all $v$, $\mathrm{Var}[\hat{J}_{2lvl}]$ simplifies to $\sum_v \frac{1}{p}\left(\frac{1}{q}(a_v - 1) + a_v^2\right)$. Thus, the problem becomes the following optimization problem.

$$\begin{aligned}\text{minimize} \quad & \sum_v \frac{1}{p}\left(\frac{1}{q}(a_v - 1) + a_v^2\right) \\ \text{s.t.} \quad & p \cdot (q(\|a\|_1 - \|a\|_0) + \|a\|_0) + p\|b\|_1 = n \\ & p, q \in [0, 1]\end{aligned}$$

This problem can be solved optimally, with the following solution (details given in Appendix A.5).

$$q = \begin{cases} \frac{n - \|a\|_0 - \|b\|_1}{\|a\|_1 - \|a\|_0}, & \text{if } \sqrt{\frac{\|a\|_0 + \|b\|_1}{\|a\|_2^2 - \|a\|_1 + \|a\|_0}} < \frac{n - \|a\|_0 - \|b\|_1}{\|a\|_1 - \|a\|_0}; \\ 1, & \text{if } \sqrt{\frac{\|a\|_0 + \|b\|_1}{\|a\|_2^2 - \|a\|_1 + \|a\|_0}} > 1; \\ \sqrt{\frac{\|a\|_0 + \|b\|_1}{\|a\|_2^2 - \|a\|_1 + \|a\|_0}}, & \text{otherwise,} \end{cases}$$

and

$$p = \frac{n}{\|b\|_1 + \|a\|_0 + q(\|a\|_1 - \|a\|_0)}.$$

Observe that two-level sampling degenerates into correlated sampling when $q = 1$, so two-level sampling can never be worse than correlated sampling. Meanwhile, whenever the optimal value of $q$ is less than 1, the reason must be that it can achieve a smaller variance. This in turn means that as long as $\|a\|_2^2 > \|a\|_1 + \|b\|_1$, two-level sampling is strictly better than correlated sampling. This condition is easily met for PK-FK joins, e.g., when each primary key is joined with at least 2 foreign keys. Note that if each primary key is joined with only one foreign key, the problem simply becomes the extreme case in Figure 1(a), which is also the best case for correlated sampling.

Next we examine, quantitatively, how much two-level sampling can be better than correlated sampling. As the choice of $p$ and $q$ depends on the desired sample size $n$, we consider the following two cases.

Let $\tau = \|a\|_0 + \|b\|_1 + (\|a\|_1 - \|a\|_0) \cdot \sqrt{\frac{\|a\|_0+\|b\|_1}{\|a\|_2^2-\|a\|_1+\|a\|_0}}$. When $n < \tau$, the variance for two-level sampling is:

$$\begin{aligned}
&\mathrm{Var}(\hat{J}_{2lvl}) = \\
&\frac{1}{n}\left(\sqrt{\frac{\|a\|_0+\|b\|_1}{\|a\|_2^2-\|a\|_1+\|a\|_0}}(\|a\|_1-\|a\|_0)+\|a\|_0+\|b\|_1\right)\cdot \\
&\left(\sqrt{\frac{\|a\|_2^2-\|a\|_1+\|a\|_0}{\|a\|_0+\|b\|_1}}(\|a\|_1-\|a\|_0)+\|a\|_2^2\right)-\|a\|_2^2 \\
\approx&\frac{(\|a\|_1+\sqrt{\|a\|_2^2\|b\|_1})^2}{n}.
\end{aligned}$$

When $n > \tau$, the variance for two-level sampling is

$$\begin{aligned}
\mathrm{Var}(\hat{J}_{2lvl}) &= \frac{(\|a\|_1-\|a\|_0)^2}{n-\|a\|_0-\|b\|_1} \\
&\le (\|a\|_1-\|a\|_0)\sqrt{\frac{\|a\|_2^2-\|a\|_1+\|a\|_0}{\|a\|_0+\|b\|_1}}.
\end{aligned}$$

For correlated sampling, we must set $p = \frac{n}{\|a\|_1+\|b\|_1}$. Its variance is thus

$$\mathrm{Var}(\hat{J}_{Cor}) = \left(\frac{\|a\|_1+\|b\|_1}{n}-1\right)\|a\|_2^2 \approx \frac{\|a\|_1+\|b\|_1}{n}\|a\|_2^2.$$

Therefore, when $n < \tau$,

$$\frac{\mathrm{Var}(\hat{J}_{2lvl})}{\mathrm{Var}(\hat{J}_{Cor})} \approx \frac{\|a\|_1\cdot\left(\frac{\|a\|_1}{\|a\|_2^2}+2\sqrt{\frac{\|b\|_1}{\|a\|_2^2}}\right)+\|b\|_1}{\|a\|_1+\|b\|_1} \approx \frac{\|b\|_1}{\|a\|_1}+\sqrt{\frac{\|b\|_1}{\|a\|_2^2}}.$$

When $n > \tau$,

$$\frac{\mathrm{Var}(\hat{J}_{2lvl})}{\mathrm{Var}(\hat{J}_{Cor})} < \frac{n}{\|a\|_1+\|b\|_1}\frac{\|a\|_1}{\sqrt{\|a\|_2^2\|b\|_1}} < \frac{n}{\|a\|_1+\|b\|_1}.$$

For PK-FK joins, we usually have $\|a\|_2^2 \gg \|a\|_1 \gg \|b\|_1$, and of course $\|a\|_1 + \|b\|_1 \gg n$ for the sampling to make sense. In Section 5, we validate these findings empirically using TPC-H benchmark data.

For independent Bernoulli sampling, we must set $q = \frac{n}{\|a\|_1+\|b\|_1}$. Its variance is thus

$$\begin{aligned}
\mathrm{Var}(\hat{J}_{Ber}) =&\|a\|_1\left(\frac{\|a\|_1+\|b\|_1}{n}\right)^2 \\
&+(\|a\|_2^2-\|a\|_1)\frac{\|a\|_1+\|b\|_1}{n}-\|a\|_2^2 \\
\approx&\|a\|_1\left(\frac{\|a\|_1+\|b\|_1}{n}\right)^2+\|a\|_2^2\left(\frac{\|a\|_1+\|b\|_1}{n}\right),
\end{aligned}$$

which is always larger than that of correlated sampling, therefore must also be larger than that of two-level sampling.

Note that in [21], it was observed empirically that correlated sampling is always better than independent Bernoulli; our analysis thus serves as the theoretical justification of this fact.

## 2.4 Many-Many Joins, Frequencies Unknown

For many-many joins, we need to solve the following optimization problem, where $\mathrm{Var}[\hat{J}_{2lvl}]$ is defined as in (1) and cannot be simplified.

$$\begin{aligned}
\text{minimize}\quad & \mathrm{Var}[\hat{J}_{2lvl}] \\
\text{s.t.}\quad & p(\|a\|_0+\|b\|_0+q(\|a\|_1-\|a\|_0)+q(\|b\|_1-\|b\|_0)) \\
& = n \\
& p,q\in[0,1].
\end{aligned}$$

Taking derivative of the objective function and equating it to zero, we have

$$\begin{aligned}
&(\|a\|_1+\|b\|_1-\|a\|_0-\|b\|_0)\cdot c_1q^3 \\
&-((\|a\|_1+\|b\|_1-\|a\|_0-\|b\|_0)c_2+(\|a\|_0+\|b\|_0)c_3)\cdot q \\
&-(\|a\|_0+\|b\|_0)c_2=0,
\end{aligned} \tag{2}$$

where

$$\begin{aligned}
c_1 &= \sum_v a_v^2b_v^2-c_2-c_3, \\
c_2 &= \sum_v (a_v-1)(b_v-1), \\
c_3 &= \sum_v (b_v-1)(a_v^2-a_v+1)+(a_v-1)(b_v^2-b_v+1).
\end{aligned}$$

Note that these quantities can all be expressed in terms of $\sum_v a_v^i b_v^j$ for $i, j = 0, 1, 2$, and each of these terms can be approximately computed by the AMS sketch of the data [9]. Therefore, we can solve (2) for the optimal solution of $p$ and $q$. The exact formula, though, is very complicated (it is a cubic equation) and thus omitted. In addition, we need to take the constraint $q \in [\frac{n-\|a\|_0-\|b\|_0}{\|a\|_1+\|b\|_1-\|a\|_0-\|b\|_0}, 1]$ into consideration, comparing the roots of (2) with the boundaries of this constraint.

Next, we compare the variance of two-level sampling with that of correlated sampling. However, using the complicated cubic roots of (2) is almost impossible for a clean comparison, so we assume $a_v \gg 1, b_v \gg 1$, and simplify the optimization problem by dropping some lower terms. Specifically, we simplify the objective function to

$$q\cdot(\frac{c_2}{q^2}+\frac{c_3}{q}+c_1), \tag{3}$$

where $c_1 \approx \langle a,a,b,b\rangle, c_2 \approx \langle a,b\rangle, c_3 \approx \langle a,a,b\rangle + \langle a,b,b\rangle$. Here we generalize the inner product notation to multiple vectors as $\langle x,y,z\rangle = \sum_i x_iy_iz_i$.

(3) is minimized when $q = \sqrt{\frac{c_2}{c_1}} = \sqrt{\frac{\|a\circ b\|_1}{\|a\circ b\|_2^2}}$. Plugging into the variance of two-level sampling and correlated sampling, we obtain

$$\frac{\mathrm{Var}(\hat{J}_{2lvl})}{\mathrm{Var}(\hat{J}_{Cor})} \approx \frac{c_3+\sqrt{c_1c_2}}{c_1}.$$

For $a_v \gg 1, b_v \gg 1$, we have $c_1 \gg c_3 \gg c_2$, so we conclude that two-level sampling is always better than correlated sampling for many-many joins.

However, there is no definitive comparison between two-level sampling with independent Bernoulli for many-many joins. In fact, according to our experimental results on Twitter data, independent Bernoulli remains the best algorithm for many-many joins when the frequencies are unknown.

## 2.5 Frequencies Known

Next, we examine how to make use the frequency information to improve the accuracy of sampling. We will first assume that the frequency vectors $a$ and $b$ are known in their entirety, and then discuss how to adapt the algorithm if only the heavy hitters are known.

Intuitively, the level-one sampling probability $p_v$ should be a monotonically increasing function of $a_v$ and $b_v$. Indeed, end-biased sampling [10] adopts the natural choice of a linear function. The level-two sampling probability $q$, on the other hand, is applied to each individual tuple, so we see no reason why this should be different for different tuples.

Let $n$ be the desired sample size. Then the problem becomes minimizing $\mathrm{Var}[\hat{J}_{2lvl}]$ as defined in (1), subject to the following constraint:

$$\sum_{v:a_v b_v \neq 0} p_v(2 + q(a_v + b_v - 2)) = n. \tag{4}$$

Note that we do not sample $v$ if $a_v b_v = 0$.

We first rewrite (1) as

$$\mathrm{Var}[\hat{J}_{2lvl}] = \sum_v \frac{1}{p_v}\left(\sigma_1^2(v) + a_v^2 b_v^2\right) - \sum_v a_v^2 b_v^2.$$

The second term does not depend on $p_v$ or $q$, so we just need to minimize the first term. Plugging in (4), we rewrite the first term as

$$\frac{1}{n}\left(\sum_v p_v(2 + q(a_v + b_v - 2))\right)\left(\sum_v \frac{1}{p_v}\left(\sigma_1^2(v) + a_v^2 b_v^2\right)\right). \tag{5}$$

Applying the Cauchy-Schwarz inequality, we have

$$(5) \geq \frac{1}{n} \cdot \left(\sum_v \sqrt{(\sigma_1^2(v) + a_v^2 b_v^2) \cdot (2 + q(a_v + b_v - 2))}\right)^2. \tag{6}$$

The equality holds when

$$p_v = C \cdot \sqrt{\frac{\sigma_1^2(v) + a_v^2 b_v^2}{2 + q(a_v + b_v - 2)}}, \tag{7}$$

where $C$ is a scaling constant. This also minimizes $\mathrm{Var}[\hat{J}_{2lvl}]$.

Interestingly, this suggests that the optimal sampling probability $p_v$ is not necessarily a linear function of the frequency of $v$. Consider for example the following two cases: For PK-FK joins, $b_v = 1$, then $p_v$ is roughly proportional to $\sqrt{a_v}$. For many-to-many joins with $a_v = b_v$, $p_v$ is roughly proportional to $a_v^{1.5}$. Neither is a linear function of $a_v$. In the former case, the contribution of a join value $v$ to the join size is linear in $a_v$, while this is $a_v^2$ in the latter case. So, high-frequency join values are more important in the latter case than in the former case, which is intuitively the reason why the sampling probability $p_v$ should take functions of different growth rates in different cases.

There is still an unknown parameter $q$, which can be found by minimizing (6). This is a complex algebraic function of $q$, consisting of as many terms as the number of distinct join values. But later we show how to simplify this by using only the heavy hitters information about the join values.

Finally, we plug (7) and the optimal value of $q$ into (4) to solve for $C$, which gives us the $p_v$'s. Note that the optimal values of some $p_v$'s may be larger than 1. For such $v$, we reduce $p_v$ to 1, and increase the corresponding $q$ so that $p_v(2 + q(a_v + b_v - 2))$ remains the same, so as to keep the desired sample size.

### *When only heavy hitters are available.*

The above procedure assumes that all the $a_v$'s and $b_v$'s are available. We observe that the small $a_v$'s and $b_v$'s have very little impact when solving the optimization problem for the optimal parameters. Therefore, in practice, we only use the frequency information of the top-$k$ most frequent join values when solving the optimization problem. For every infrequent join value $v$, we simply set

$$a_v = \frac{\|a\|_1 - \sum_{i=1}^k a_i}{\|a\|_0 - k},$$

where $a_1, \ldots, a_k$ are the frequencies of the top-$k$ most frequent join values. Even for the top-$k$ frequent values, we do not need to know their exact frequencies. Instead, there are many efficient one-pass algorithms [17, 7] that we can use for finding the heavy hitters and their approximate frequencies. The same is done on table $B$.

Since we use the same frequency for all the infrequent join values, their corresponding terms in (6) are the same, thus can be combined into one, reducing the number of terms to at most $2k + 1$. Then we can efficiently minimize (6) using numerical methods. Similarly, the equation involving the scaling constant $C$ also has at most $2k + 1$ terms, so can be solved easily.

Although the actual frequencies of the infrequent join values do not affect the optimal values of $q$ and $C$ significantly, they may affect the individual $p_v$'s a lot. Therefore, we use $a_v = a_k$ and $b_v = b_k$ to first compute an upper bound on its actual $p_v$, denoted $\bar{p}_v$, and use $\bar{p}_v$ to do the sampling. Then, for each infrequent join value sampled, we keep a counter to compute its actual $a_v, b_v$, which can then be used to compute the actual $p_v$. Finally, we delete from the sample any join value $v$ with $h(v) > p_v$.

## 3. CONFIDENCE INTERVALS

For any sampling method to be useful in practice, there must be a way to tell how accurate the result is. This is commonly expressed as a confidence interval, i.e., for a given confidence level $1 - \delta$, the algorithm should return an estimator $\hat{J}$ as well as an $\varepsilon$, such that

$$\Pr[\hat{J} - \varepsilon \leq J \leq \hat{J} + \varepsilon] \geq 1 - \delta,$$

where $J$ is the true join size. Here, $\varepsilon$ is also called the half-width of the confidence interval. The challenge is that the algorithm needs to compute $\varepsilon$ also from the sample, without looking at the raw data.

### 3.1 Correlated Sampling

We start from the simpler case, correlated sampling, which a special case of two-level sampling with $p_v = p$ for all $v$ and $q = 1$. Observe that the estimator is the sum of independent Bernoulli random variables, one for each $v$, that equals $a_v b_v / p_v$ with probability $p_v$ and 0 otherwise. According to central limit theorem[2], we have

$$\frac{\hat{J}_{Cor} - J}{\mathrm{Var}[\hat{J}_{Cor}]} \xrightarrow{d} N(0, 1),$$

where $N(0, 1)$ denotes the normal distribution with mean 0 and variance 1, and $\xrightarrow{d}$ means "converge in distribution".

[2]The standard central limit theorem also requires the random variables to be identically distributed, but the theorem actually applies to non-identical distributions as well, provided that each of them has bounded variance.

Thus it is sufficient to have an estimate for $\mathrm{Var}[\hat{J}_{Cor}] = \left(\frac{1}{q}-1\right)\sum_v a_v^2 b_v^2$. Since correlated sampling samples all the $a_v$ tuples in $A$ and the $b_v$ tuples from $B$ with probability $p$, and nothing otherwise, $\mathrm{Var}[\hat{J}_{Cor}]$ can be easily estimated as

$$\widehat{\mathrm{Var}}[\hat{J}_{Cor}] = \left(\frac{1}{q}-1\right)\frac{1}{q}\sum_v |S_A(v)|^2 |S_B(v)|^2. \quad (8)$$

When there are selection predicates, we simply replace $S_A(v)$ and $S_B(v)$ with $S_A^{c_A}(v)$ and $S_B^{c_B}(v)$, respectively. Plugging the estimated $\mathrm{Var}[\hat{J}_{Cor}]$ into (8), the half width of the confidence interval (with confidence level $1-\delta$) can be computed as $\varepsilon = z_{\delta/2}\sqrt{\widehat{\mathrm{Var}}[\hat{J}_{Cor}]}$, where $z_{\delta/2} = \Phi^{-1}(\delta/2)$ and $\Phi(\cdot)$ is the cdf of the normal distribution.

## 3.2 Two-Level Sampling

The estimator of two-level sampling is also the sum of independent random variables (though not Bernoulli), so we can still use the central limit theorem, and the problem reduces to estimating $\mathrm{Var}[\hat{J}_{2lvl}]$. However, the form of $\mathrm{Var}[\hat{J}_{2lvl}]$, as show in (1), is more complicated.

Expanding (1), we see that each term of $\mathrm{Var}[\hat{J}_{2lvl}]$ is of the form

$$f(p_v, q) g_a(v) g_b(v), \quad (9)$$

where $f(p_v, q)$ is a a function of $p_v$ and $q$, which are known, and $g_a(v)$ is one of $a_v^{c_A}, (a_v^{c_A})^2$, or $\frac{a_v^{c_A}}{a_v}$, $g_b(v)$ is one of $b_v^{c_B}, (b_v^{c_B})^2$, or $\frac{b_v^{c_B}}{b_v}$. Below, conditioned upon $v$ being sampled, we derive independent unbiased estimators $\hat{g}_a(v)$, $\hat{g}_b(v)$ for $g_a(v)$, $g_b(v)$. Since $v$ is sampled with probability $p_v$, an unbiased estimator for (9) is thus

$$\begin{cases} \frac{1}{p_v} f(p_v, q)\hat{g}_a(v)\hat{g}_b(v), & \text{if } v \text{ is sampled;} \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

We just describe how to estimate $g_a(v)$; $g_b(v)$ can be estimated the same way. An unbiased estimator for $\frac{a_v^{c_A}}{a_v}$ is $I_A^{c_A}(v)$. Recall that $I_A^{c_A}(v)$ is 1 if the sentry $s_A(v)$ satisfies the predicate $c_A$, and 0 otherwise. Since we choose the sentry uniformly at random from the $a_v$ tuples in $A(v)$, we have $\mathbb{E}[I_A^{c_A}(v)] = \Pr[s_A(v) \text{ satisfies } c_A] = \frac{a_v^{c_A}}{a_v}$.

An unbiased estimator for $a_v^{c_A}$ is

$$\widehat{a_v^{c_A}} = \frac{|S_A^{c_A}(v)|}{q} + I_A^{c_A}(v).$$

An unbiased estimator for $(a_v^{c_A})^2$ is

$$\widehat{(a_v^{c_A})^2} = \left(\frac{S_A^{c_A}(v)}{q} + I_A^{c_A}(v)\right)^2 + \left(1-\frac{1}{q}\right)\frac{|S_A^{c_A}(v)|}{q}.$$

We prove the unbiasedness of these estimators in Appendix A.7.

By linearity of expectation, summing up (10) over all $v$ gives us an unbiased estimator of $\mathrm{Var}[\hat{J}_{2lvl}]$. Then the half-width of the confidence interval is $\varepsilon = z_{\delta/2}\sqrt{\widehat{\mathrm{Var}}[\hat{J}_{2lvl}]}$.

# 4. MULTI-TABLE JOINS

In this section, we extend two-level sampling to two most common multi-table joins, chain joins and star joins. Unfortunately, it is still not clear how it can be applied to an arbitrary multi-table join.

## 4.1 Chain Join

For concreteness, we describe how two-level sampling is applied on a three-table chain join

$$A \bowtie_{A.PK=B.FK} B \bowtie_{B.PK=C.FK} C,$$

where $A$'s primary key joins a foreign key in $B$, and $B$'s primary key joins a foreign key in $C$. This is very common in relational database schemas. For example, in the TPC-H benchmark, `customer` joins `orders` on `c_custkey = o_custkey`, while `orders` joins `lineitem` on `o_orderkey = l_orderkey`. Extension to 4 or more tables is straightforward.

The idea is to use only the first level of sampling on $A$ and $B$, while use two levels on $C$. More precisely, we make use of two independent random hash functions $h_1$ and $h_2$. Tuples in $A$ are sampled into $S_A$ if $h_1(u) < p$ where $u$ is the join value of $A$ (i.e., its primary key). Tuples in $B$ are sampled into $S_B$ if $h_1(u) < p$ and $h_2(v) < p$, where $u$ is its join value with $A$ (i.e., the foreign key in $B$) and $v$ is its join value with $C$ (i.e., primary key of $B$). For each join value $v$ in $C$, if $h_2(v) < p$, a sentry $s_C(v)$ is sampled uniformly at random from $C(v)$, the set of tuples in $C$ sharing the join value $v$, while each of the other tuples in $C(v)$ is sampled with probability $q$.

The unbiased estimator for join size $J$, when each table may have a selection predicate, is

$$\hat{J} = \sum_{u,v} \hat{J}_{u,v},$$

where $\hat{J}_{u,v}$ is defined as:

$$\hat{J}_{u,v} = \begin{cases} \frac{1}{p^2}\cdot|S_A^{c_A}(u)|\cdot|S_B^{c_B}(u,v)|\cdot\left(\frac{|S_C^{c_C}(v)|}{q} + I_C^{c_C}(v)\right), & \\ \qquad\qquad \text{if } h_1(u) < p \text{ and } h_2(v) < p; & \\ 0, \qquad \text{otherwise.} & \end{cases}$$

Here, $S_A^{c_A}(u)$ denotes the set of tuples in $A$ with join value $u$ that satisfy the predicate $c_A$, $S_B^{c_B}(u,v)$ denotes the set of tuples in $B$ with join value $u, v$ (i.e., its foreign key equals $u$ and primary key equals $v$) that satisfy the predicate $c_B$. $S_C^{c_C}(v)$ denotes the set of tuples in $C$ with join value $v$ that satisfy the predicate $c_C$; $I_C^{c_A}(v)$ is 1 if $s_C(v)$ satisfies $c_C$, and 0 otherwise.

Finding the optimal values of $p$ and $q$ for a multi-table join is a very complicated optimization problem. Observing that in practice, we usually have $|A| \ll |B| \ll |C|$ for chain joins, so the problem is largely dominated by $B$ and $C$. Thus, as a simple heuristic, we set the values of $p$ and $q$ by simply considering the join between $B$ and $C$, using the formula derived in Section 2.3.

## 4.2 Star Join

A star join is performed on a large fact table and several dimension tables. Each dimension table has a primary key that joins a foreign key in the fact table. Again, we describe the sampling algorithm on the following three-table star join; extension to more than 3 tables is straightforward.

$$A \bowtie_{A.PK=B.FK_A} B \bowtie_{B.FK_C=C.PK} C.$$

In the star join above, $B$ is the fact table, while $A$ and $C$ are dimension tables. Still using the TPC-H benchmark example, $B$ can be `lineitem`, while $A$ and $C$ are `part` and

`supplier`, respectively, joining on `p_partkey = l_partkey` and `s_suppkey = l_suppkey`.

The idea is to use one level of sampling on $A$ and $C$ while use two levels on $B$. Similar to chain joins, we make use of two independent random hash functions $h_1$ and $h_2$. Tuples in $A$ are sampled into $S_A$ if $h_1(u) < p$ where $u$ is the join value of $A$ (i.e., its primary key). In the same way, tuples in $C$ are sampled into $S_C$ if $h_1(v) < p$ where $v$ is the join value of $C$. Treatment of $B$ is sightly different, as it has two join attributes, both of which are foreign keys. The idea is to consider each combination of $u, v$ and perform the second level sampling for each combination. More precisely, let $B(u, v)$ be the set of tuples in $B$ with its $FK_A$ value equal to $u$ and $FK_C$ value equal to $v$, a sentry $s_B(u, v)$ is sampled uniformly from $B(u, v)$; then each of the other tuples in $B(u, v)$ is sampled with probability $q$.

The unbiased estimator for join size $J$ is

$$\hat{J} = \sum_{u,v} \hat{J}_{u,v},$$

where $\hat{J}_{u,v}$ is defined as:

$$\hat{J}_{u,v} = \begin{cases} \frac{1}{p^2} \cdot |S_A^{c_A}(u)| \cdot \left( \frac{|S_B^{c_B}(v)|}{q} + I_B^{c_B}(v) \right) \cdot |S_C^{c_C}(v)|, \\ \qquad \text{if } h_1(u) < p \text{ and } h_2(v) < p; \\ 0, \qquad \text{otherwise.} \end{cases}$$

To find the optimal values of $p$ and $q$, we imagine the star join as a two table PK-FK join, by combining all the dimension tables into one large dimension table, and then use the formula derived in Section 2.3.

# 5. EXPERIMENTS

We have experimentally compared the estimation accuracy of four sampling algorithms: independent Bernoulli, correlated sampling, end-biased sampling and two-level sampling on a variety of joins, under the same sample size constraint. For two-level sampling, the experiments included its two versions, when the frequency information is either unknown or known; for the latter case, we only use the top-10,000 most frequent join values as described in Section 2.5. Note that end-biased sampling also requires the frequency information, and we in fact give it the information in full.

For each experiment, we repeat the sampling procedure 500 times and measure the relative error from the true join size. In the figures below, we report both the median error and the 90%-quantile error.

## 5.1 PK-FK Joins

For PK-FK joins, we used the TPC-H data with a scale factor of 10. We used the join between `lineitem` and `supplier` on `l_suppkey = s_suppkey`. Here is the basic statistical information about the two tables: $\|a\|_1 = 5.9 \times 10^7, \|a\|_0 = \|b\|_1 = 10^5, \|a\|_2^2 = 3.6 \times 10^{10}$.

Figure 2 shows the relative error of the sampling algorithms for different sampling sizes, which vary from 0.01% to 3% of the raw data. Note that TPC-H data is quite uniform, so all the $a_v$'s are similar. This means that end-biased sampling essentially degenerates into correlated sampling, while frequency information is not useful for two-level sampling, either. So the results of these two frequency-aware sampling algorithms are not shown in the figure.

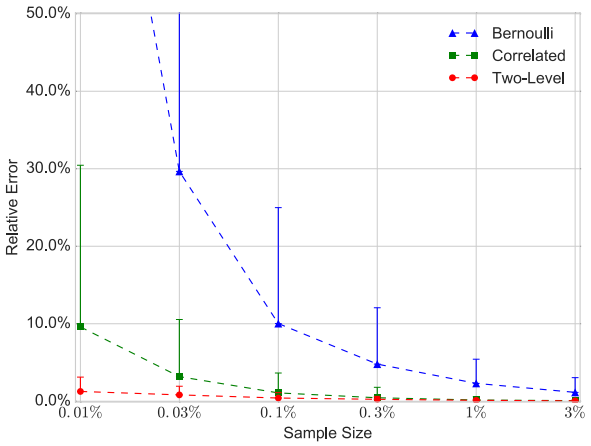


**Figure 2: PK-FK join on TPC-H data.**

These experimental results validate our analytical comparison in Section 2.3: The analysis shows that $\frac{\mathrm{Var}[\hat{J}_{2lvl}]}{\mathrm{Var}[\hat{J}_{Cor}]} \approx \frac{\|b\|_1}{\|a\|_1} + \sqrt{\frac{\|b\|_1}{\|a\|_2^2}} \approx 10^{-2}$. Note that the variance is the square of the error, so we expect a 10-fold reduction in terms of the error. Our experimental results largely confirm this prediction, with some small discrepancy due to some lower order terms being omitted in the analysis. We also see that correlated sampling performs better than independent Bernoulli, which agrees with the previous experimental results [21], and also validates our analysis in Section 2.3.

In addition, we see that the gap between the median error and the 90%-quantile error is quite small for two-level sampling, which means that it is more robust than the other sampling methods. This is not surprising, since by the central limit theorem, the errors follows a normal distribution, so this gap also depends on the variance of the estimators.

### *Selection predicates.*

Next, we add predicates to the join with varying selectivities. Specifically, we added a predicate `discount < x` and varied `x` to control the selectivity, which is defined as the percentage of tuples satisfying the predicate. We also tried using two predicates `discount < x` and `shipdate < y` together.

The experimental results are shown in Figures 3 and 4. We see that as the predicates become more selective, i.e., less tuples are selected, the errors of all sampling schemes increase . This is because the number of sampled tuples that pass the predicates decrease, reducing the effective sample size. The experimental results suggest that Bernoulli sampling suffers more from selective predicates than the other sampling schemes.

### *Data skew.*

The default TPC-H data is quite uniform. To see how the sampling algorithms deal with data skew, we regenerated the `lineitem` table following the Zipf distribution. This results in a few large suppliers with many lineitems, and a long tail of small suppliers. The skewness is controlled by the Zipf parameter $\alpha$, which we vary from 0 (no skew) to 2 (highly skewed). Note that a high skewness leads to a larger

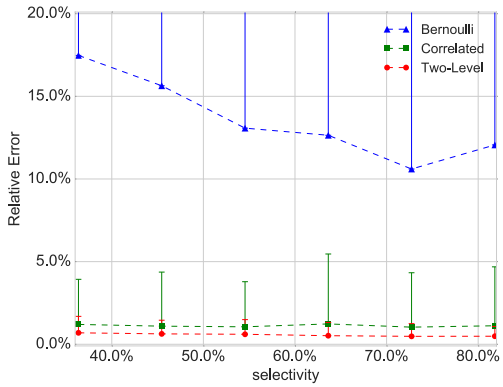


**Figure 3: PK-FK join on TPC-H data with one predicate**

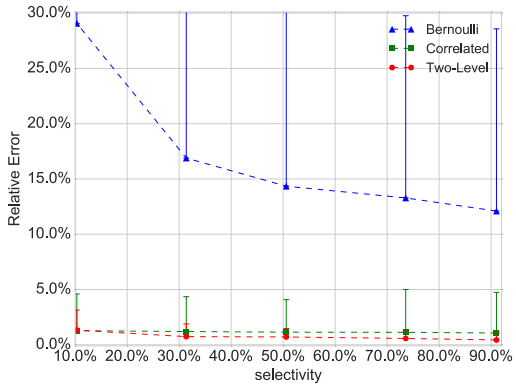


**Figure 4: PK-FK join on TPC-H data with two predicates**

$\|a\|_2^2$, and two-level sampling (even if frequency information is unknown) should perform better on more skewed data.

The results are shown in Figure 5 over a varying $\alpha$, while the sample size is fixed at 0.1% of raw data. From the results we see the following trends: (1) As predicted by our analysis, the gap between two-level sampling and the other sampling algorithms becomes larger as skewness increases. (2) All sampling algorithms, except the frequency-aware version of two-level sampling, suffer from high skewness. This suggests that frequency information (at least the heavy hitters) is very important in controlling the error of the estimation. (3) Somehow counter-intuitively, end-biased sampling also performs badly on skewed data. In fact, the same phenomenon was also observed in previous experimental studies [21]. The explanation is that, end-biased sampling uses a sampling probability proportional the frequency to sample the heavy hitters. This reduces their variances linearly, but the sample size allocated to the heavy hitters also grows linearly. As the error is the square root of the variance, this actually results in a worse net effect.

*Confidence intervals.*

Next, we validate our method for computing the confidence intervals. For this purpose, we run the two-level sampling algorithm 500 times, each time computing a confidence interval with various confidence levels, and then see how

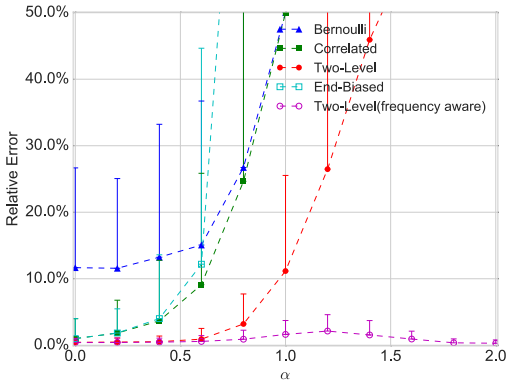


**Figure 5: PK-FK join on skewed TPC-H data**

| | Sample size | | |
|---|---|---|---|
| **Confidence level** | **0.1%** | **0.3%** | **1%** |
| **80%** | 93.2% | 90.8% | 80% |
| **90%** | 97.4% | 96.2% | 90% |
| **95%** | 99.4% | 99% | 96.4% |
| **98%** | 100% | 99.6% | 98.6% |
| **99%** | 100% | 99.6% | 99.4% |
| **99.8%** | 100% | 99.8% | 99.8% |

**Table 2: Percentages of runs in which true join size falls inside confidence interval.**

many times the true join size indeed falls inside the confidence interval. The results are shown in Table 2. Note that ideally (i.e., if the estimator's distribution follows exactly the normal distribution), we would expect to see the percentage of runs in which the true join size falls inside the confidence interval to match the given confidence level. Empirically, we see that for relatively large sample size, say 1%, the two percentages are almost equal, suggesting that the central limit theorem applies quite well. For smaller sample sizes, it seems that the actual distribution of the estimator is more concentrated than the normal distribution, so that the actual percentage of the confidence intervals containing the true join size is larger than the given confidence level. This means that when sample size is small, there is more leeway in the confidence intervals computed based on the central limit theorem.

## 5.2 Many-Many Joins

For many-many joins, we first used the TPC-H data to perform the join between `lineitem` and `partsupp` on `l_suppkey = ps_suppkey`. As this is a uniform data set, we did not include the two frequency-aware sampling algorithms. The results on the other sampling algorithms are shown in Figure 6. We note that the previous study [21] did not perform experiments on many-many joins, thus drawing the conclusion that correlated sampling is always better than independent Bernoulli. However, as we see from Figure 6, it is the other way around on many-many joins. Two-level sampling, which combines the advantages of both independent Bernoulli and correlated sampling, performs better than both.

To see how the sampling algorithms perform on skewed data, we used the Twitter data set, obtained from [14]. It describes the follower-followee relationship among Twitter

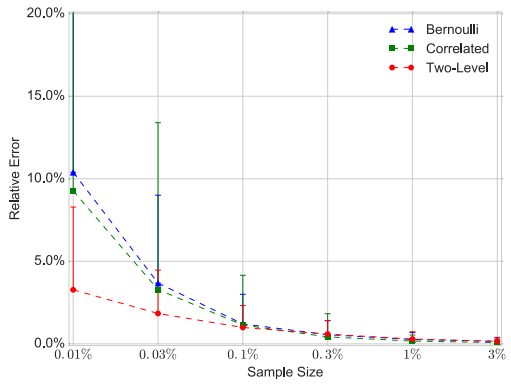


**Figure 6: Many-many join on TPC-H data**

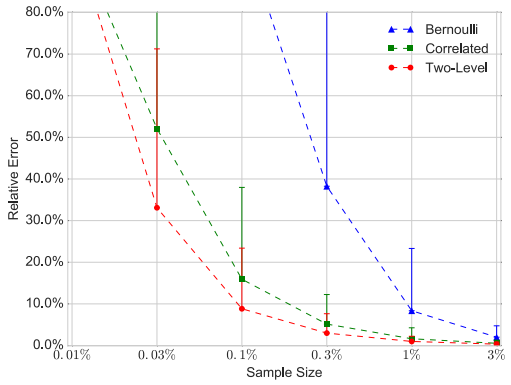


**Figure 8: 3-table chain join on TPC-H data**

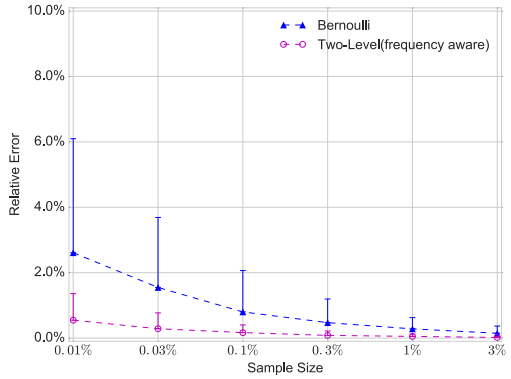


**Figure 7: Many-many join on Twitter Data**

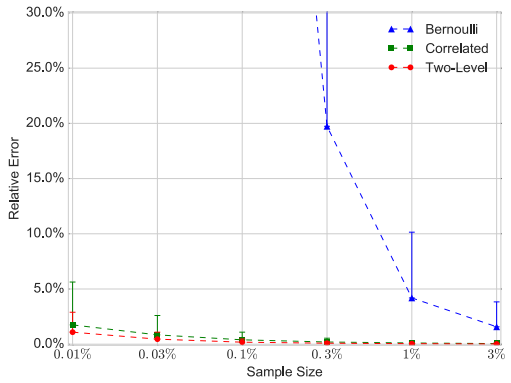


**Figure 9: 3-table star join on TPC-H data**

users. The table has $1.4 \times 10^9$ tuples involving $4.2 \times 10^7$ users. It features a highly skewed distribution, where a few users have many followers and followees, with a long tail of users with smaller degrees. We join the table with a logical copy of itself, to find all the 2-hop "follows" relationships, i.e., `person1` $\rightarrow$ `person2` $\rightarrow$ `person3`. Note that estimating the size of such joins is crucial in choosing the best query plan for more complex graph pattern joins, such as triangle joins. Figure 7 shows the results on this join. Correlated sampling and end-biased sampling perform very badly that their errors are beyond the scope of the plot. The relative error for correlated sampling is more than 1000 times that of two-level sampling, and the relative error for end-biased sampling is around 400 times that of two-level sampling. We already saw earlier that they perform badly on PK-FK joins when data is skewed in one table (there is no skew in the PK table by definition), while the join on the Twitter data is between two highly skewed tables, which explains why their errors are off the chart. Two-level sampling, when frequency information is unknown, also performs badly. So the only competitive algorithms in this case are independent Bernoulli and two-level sampling (frequency aware). Between the two, two-level sampling outperforms independent Bernoulli by roughly a factor of 3.

## 5.3 Multi-Table Joins

Finally, we conducted experiments on chain and star joins using TPC-H data. First, we used the 3-table chain join `customer` $\bowtie$ `orders` $\bowtie$ `lineitem`, where the first two relations join on `custkey` while the latter two join on `orderkey`. Figure 8 shows the results for chain join. The trends are similar to those observed earlier on PK-FK joins in Figure 2, because a chain join can be seen as a generalization of a PK-FK join. However, compared with Figure 2, both correlated sampling and two-level sampling now suffer from a very small sample size, since it is more difficult to sample joined tuples from three tables at the same time.

For star joins, where we used `lineitem` as the fact table, and used `part` and `supplier` as the dimension tables. The joining attributes are `partkey` and `suppkey`, respectively. The results on this join is shown in Figure 9. Again, qualitatively the trends are similar to those on PK-FK joins in Figure 2. However, the gap between correlated sampling and two-level sampling has narrowed, since the having a second dimension table effectively introduces more join values as each combination of the dimensions defines a "join value". This in turn reduces the $\ell_2$-norm of the fact table, which determines the relative advantage of two-level sampling over correlated sampling.

## 5.4 Performance

Eventually, the sampling will be used by the query optimizer for estimating intermediate join sizes at query time, before the query is actually executed. Thus it has to be very lightweight. To this end, we have run experiments measuring the running time of the join size estimation algorithm.

| Data set | Query | % runs leading to optimal plan | | |
|---|---|---|---|---|
| | | Bernoulli | Correlated | Two-Level |
| TPC-H data | `lineitem` ⋈ `orders` ⋈ `supplier` | 53% | 73% | 85% |
| Twitter | triangle query | 62% | 41% | 86% |

**Table 3: Join size estimation for query optimization**

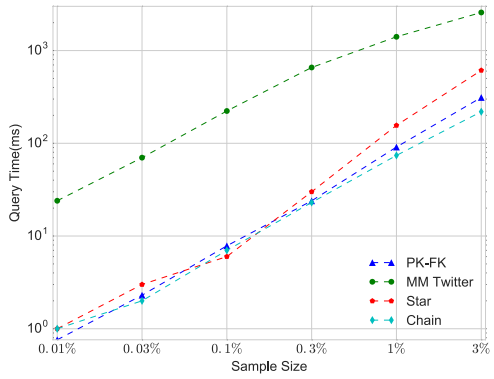


**Figure 10: Join size estimation times on different queries**

The experiments were conducted on a machine with an Intel i7-4770 processor and 32 GB of main memory. Note that even on the largest data set (24GB Twitter data), the largest sample has size 720MB and comfortably fits in memory.

All sampling schemes operate in two s to estimate a join size: (1) scan through the sample and find all tuples satisfying the predicates; and (2) join the tuples and scale up the join size. The only difference among different sampling schemes is the scaling up formula, so the query times of different sampling schemes are almost identical. Thus we just report the running times of two-level sampling below.

Figure 10 shows the query time of two-level sampling for different queries over different sample sizes. It is clear that the query time has a linear dependency on the sample size (queries on the Twitter data is slower because the base data size is larger, so the absolute sample size is also larger). This is because step (1) is clearly linear; and step (2) is also linear as we can store the tuples in the join key order so that we can run sort-merge join in linear time. Also, even for many-many joins, we do not have to actually enumerate all join results in order to compute the join size. Our scaling up formula just needs the number of tuples in the sample for each join key value. The figure shows that for most cases, the query time is on the order of milliseconds. It can get larger for larger sample size, but our previous experiments already show that a sample size of 0.1% already provides enough accuracy for most cases.

To put these numbers into context, we also ran the same queries (in full, without indexes) in Oracle Database 12c. For the PK-FK query, the time is 150 seconds; for the many-many join on the Twitter data, it takes more than 10 hours. It means that the milliseconds spent in join size estimation are really negligible in the overall query time, and may be well worthy.

## 5.5 Applicability and Coverage

Although it is still not clear how two-level sampling can be applied to an arbitrary join, the current set of joins supported (two-table joins, chain joins, and star joins) already provide good coverage. To quantify this, we examined all the 22 queries in the TPC-H benchmark. Among them, 11 of them involve joins with 3 or more tables, for which the join order optimization problem arises. For each such query, we enumerated all its join plans, and checked whether every intermediate join size of each plan can be estimated by two-level sampling. Meanwhile, we note that the TPC-H benchmark data has two tiny dimension tables `nation` and `region`. We can sample all their tuples such that any intermediate join involving these tiny tables can be estimated without extra error. Then, it turns out that two-level sampling can cover nearly all the 11 queries of interest. The only exception is Q5, which has 2 join plans (out of a total of 8) containing intermediate joins that cannot be estimated by two-level sampling.

Finally, to see the potential impact on query optimization, we ran experiments to test the ability of these sampling methods to find the optimal join plan for a given query. Here, we made the simplifying assumption that the cost of a join plan depends solely on the intermediate join sizes; in practice, many other factors should be taken into account, such as the availability of indexes, which tables are cached in memory and which are on disk, what join algorithms can be used, etc, which are beyond the scope of this work. We chose two typical queries. The first is a 3-table PK-FK join `lineitem` ⋈ `orders` ⋈ `supplier` on the TPC-H benchmark data, which has 2 join plans. The second is a many-to-many triangle self-join on the Twitter data, which has 3 join plans. To make the costs of these 3 plans different, we used an asymmetric triangle join: `person1` → `person2`, `person2` → `person3`, `person1` → `person3`. We repeat each query 500 times and see how many times the estimates of for the intermediate join sizes lead to the optimal plan being selected. The results in Table 3 again confirm our previous analysis: Correlated sampling performs better on PK-FK joins, while Bernoulli sampling works better on many-to-many joins. Two-level sampling, combining the benefits of both worlds, performs the best in either case.

# 6. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we have introduced a new sampling algorithm for join size estimation called two-level sampling, which combines the advantages of previous sampling algorithms and outperforms them on a variety of joins. One important direction for future work is to integrate it into large-scale distributed query processing engines. Recent studies [5] on complex join processing have suggested that the intermediate result size plays a critical role in determining the best join algorithm to use, especially in a distributed environment. Therefore, it would be very interesting to see how the proposed algorithm can contribute to large-scale distributed query processing engines.

# 7. REFERENCES

[1] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query

answering. In *ACM SIGMOD Record*, volume 28, pages 275–286. ACM, 1999.

[2] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.

[3] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '99, pages 10–20, New York, NY, USA, 1999. ACM.

[4] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *The VLDB Journal*, 10(2-3):199–223, 2001.

[5] S. Chu, M. Balazinska, and D. Suciu. From theory to practice: Efficient join query evaluation in a parallel database system. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2015.

[6] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *Proc. International Conference on Very Large Data Bases*, 2005.

[7] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. In *Proc. International Conference on Very Large Data Bases*, 2008.

[8] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[9] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 61–72. ACM, 2002.

[10] C. Estan and J. F. Naughton. End-biased samples for join cardinality estimation. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 20–20. IEEE, 2006.

[11] S. Ganguly, P. B. Gibbons, Y. Matias, and A. Silberschatz. Bifocal sampling for skew-resistant join size estimation. In *ACM SIGMOD Record*, volume 25, pages 271–281. ACM, 1996.

[12] P. J. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Proc. Ninth Intl. Conf. Scientific and Statistical Database Management*, 1997.

[13] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. Quickr: Lazily approximating complex ad-hoc queries in big data clusters. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2016.

[14] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, 2010.

[15] F. Li, B. Wu, K. Yi, and Z. Zhao. Wander join: Online aggregation via random walks. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2016.

[16] R. J. Lipton and J. F. Naughton. Query size estimation by adaptive sampling. In *Proc. ACM Symposium on Principles of Database Systems*, 1990.

[17] A. Metwally, D. Agrawal, and A. Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems*, 31(3):1095–1133, 2006.

[18] F. Rusu and A. Dobra. Pseudo-random number generation for sketch-based estimations. *ACM Transactions on Database Systems*, 32(2), Article 11, 2007.

[19] F. Rusu and A. Dobra. Sketches for size of join estimation. *ACM Transactions on Database Systems*, 33:1–46, 2008.

[20] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation.

[21] D. Vengerov, A. C. Menck, M. Zait, and S. P. Chakkappen. Join size estimation subject to filter conditions. *Proceedings of the VLDB Endowment*, 8(12):1530–1541, 2015.

[22] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

# APPENDIX

# A. DETAILED ANALYSES

## A.1 Proof of Unbiasedness of $\hat{J}$

PROOF. For each $v$ such that $a_v b_v \neq 0$, we have:

$$\begin{aligned}\mathbb{E}[\hat{J}_v] =& p \cdot \mathbb{E}[\hat{J}_v \mid h(v) < p] \\ =& \mathbb{E}\left[\left(\frac{|S_A(v)|}{q} + 1\right) \cdot \left(\frac{|S_B(v)|}{q} + 1\right) \mid h(v) < p\right] \\ =& \left(\frac{q(a_v - 1)}{q} + 1\right) \cdot \left(\frac{q(b_v - 1)}{q} + 1\right) = a_v b_v.\end{aligned}$$

For each $v$ such that $a_v b_v = 0$, we have $\mathbb{E}[\hat{J}_v] = 0 = a_v b_v$.

It follows that:

$$\mathbb{E}[\hat{J}] = \sum_v a_v b_v = |A \bowtie B|.$$

□

## A.2 Proof of Variance of $\hat{J}$

PROOF. For each $v$ such that $a_v b_v = 0$, we know that $\mathrm{Var}[\hat{J}_v] = 0$.

For each $v$ such that $a_v b_v \neq 0$, we define $X_v = pq^2 \cdot \hat{J}_v$ and define $Y_v = (|S_A(v)| + q) \cdot (|S_B(v)| + q)$. We have that:

$$X_v = \begin{cases} Y_v, & \text{if } h(v) \leq p, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Note that $X_v$ equals $Y_v$ with probability $p$ and 0 otherwise. And we now have $\mathrm{Var}[\hat{J}] = \sum_v \mathrm{Var}[\hat{J}_v] = \sum_{v:a_v b_v \neq 0} \mathrm{Var}[\hat{J}_v] = \sum_{v:a_v b_v \neq 0} \mathrm{Var}(X_v)/p^2 q^4$.

By Law of Total Variance, $\mathrm{Var}[X_v] = \mathbb{E}_{Y_v}[\mathrm{Var}(X_v \mid Y_v)] + \mathrm{Var}_{Y_v}[\mathbb{E}[X_v \mid Y_v]]$.

We have

$$\mathbb{E}_{Y_v}[\mathrm{Var}[X_v \mid Y_v]] = \mathbb{E}_{Y_v}[p(1-p)Y_v^2] = p(1-p)(\mathrm{Var}[Y_v] + \mathbb{E}^2[Y_v]),$$

and

$$\mathrm{Var}_{Y_v}[\mathbb{E}[X_v \mid Y_v]] = \mathrm{Var}_{Y_v}[pY_v] = p^2 \mathrm{Var}[Y_v].$$

It follows that

$$\mathrm{Var}[X_v] = p\mathrm{Var}[Y_v] + p(1-p)\mathbb{E}^2[Y_v].$$

We have

$$\mathbb{E}[Y_v] = \mathbb{E}\left[(|S_A(v)| + q) \cdot (|S_B(v)| + q)\right] = q^2 a_v b_v,$$

and

$$\begin{aligned}\mathrm{Var}[Y_v] =& \mathbb{E}\left[(|S_A(v)| + q)^2\right] \cdot \mathbb{E}\left[(|S_B(v)| + q)^2\right] - \\ & \mathbb{E}^2\left[(|S_A(v)| + q)\right] \cdot \mathbb{E}^2\left[(|S_B(v)| + q)\right].\end{aligned}$$

Since $|S_A(v)| \sim \mathrm{Binomial}(a_v - 1, q)$ and $|S_B(v)| \sim \mathrm{Binomial}(b_v - 1, q)$, we have

$$\mathbb{E}[|S_B(v)|] = q(a_v - 1),$$

$$\mathbb{E}[|S_B(v)|^2] = q^2(a_v - 1)^2 + (a_v - 1)q(1-q).$$

It follows that

$$\mathrm{Var}[\hat{J}] = \sum_{v:a_v b_v \neq 0} \mathrm{Var}[\hat{J}_v] = \sum_{v:a_v b_v \neq 0} \frac{1}{p}\sigma_1^2(v) + \sigma_2^2(v),$$

where $\sigma_1^2(v)$ and $\sigma_2^2(v)$ are as claimed in the lemma. □

## A.3 Proof of Unbiasedness with Selection Predicates

For each $v$ such that $a_v^{c_A} b_v^{c_B} = 0$, we have $\mathbb{E}[\hat{J}_v] = 0 = a_v^{c_A} b_v^{c_B}$.

For each $v$ such that $a_v^{c_A} b_v^{c_B} \neq 0$, conditioned on $h(v) < p$ we have that:

$$\begin{aligned}&\mathbb{E}\left[\frac{|S_A^{c_A}(v)|}{q} \mid h(v) < p\right] \\ =& \mathbb{E}\left[\frac{|S_A^{c_A}(v)|}{q} \mid I_A^{c_A}(v) = 0\right] \cdot Pr[I_A^{c_A}(v) = 0] \\ &+ \mathbb{E}\left[\frac{|S_A^{c_A}(v)|}{p_A} \mid I_A^{c_A}(v) = 1\right] \cdot Pr[I_A^{c_A}(v) = 1] \\ =& \frac{a_v^{c_A}}{a_v}(a_v - 1),\end{aligned}$$

$$\mathbb{E}[I_A^{c_A}] = \frac{a_v^{c_A}}{a_v},$$

$$\begin{aligned}\mathbb{E}[\hat{J}_v] =& p \cdot \mathbb{E}[\hat{J}_v \mid h(v) < p] \\ =& p \cdot \mathbb{E}\left[\frac{|S_A^{c_A}(v)|}{q} + I_A^{c_A}(v) \mid h(v) < p\right] \\ & \cdot \mathbb{E}\left[\frac{|S_B^{c_B}(v)|}{q} + I_B^{c_B}(v) \mid h(v) < p\right] = a_v^{c_A} b_v^{c_B}.\end{aligned}$$

It follows that $\hat{J}$ is an unbiased estimator.

$$\mathbb{E}[\hat{J}] = \sum_v a_v^{c_A} b_v^{c_B} = |A \bowtie_{c_A, c_B} B|.$$

## A.4 Proof of Variance with Selection Predicates

PROOF. For each $v$ such that $a_v^{c_A} b_v^{c_B} = 0$, we know that $\mathrm{Var}[\hat{J}_v] = 0$. For each $v$ such that $a_v^{c_A} b_v^{c_B} \neq 0$, we define $X_v = pq^2 \cdot \hat{J}_v$ and define $Y_v = (S_A^{c_A}(v) + qI_A^{c_A}(v)) \cdot (S_B^{c_B}(v) + qI_B^{c_B}(v))$:

$$X_v = \begin{cases} Y_v, & \text{if } h(v) \leq p; \\ 0, & \text{otherwise.} \end{cases}$$

Note that $X_v$ equals $Y_v$ with probability $p$ and 0 otherwise. And we now have $\mathrm{Var}[\hat{J}] = \sum_v \mathrm{Var}[\hat{J}_v] = \sum_{v:a_v b_v \neq 0} \mathrm{Var}[\hat{J}_v] = \sum_{v:a_v b_v \neq 0} \mathrm{Var}[X_v]/p^2 q^2 q^2$ as $\hat{J}_v$'s are pair-wise independent.

By Law of Total Variance, $\mathrm{Var}[X_v] = \mathbb{E}_{Y_v}[\mathrm{Var}(X_v \mid Y_v)] + \mathrm{Var}_{Y_v}[\mathbb{E}[X_v \mid Y_v]]$.

We have:

$$\mathbb{E}_{Y_v}[\mathrm{Var}[X_v \mid Y_v]] = \mathbb{E}_{Y_v}[p(1-p)Y_v^2] = p(1-p)(\mathrm{Var}[Y_v] + \mathbb{E}^2[Y_v]),$$

and

$$\mathrm{Var}_{Y_v}[\mathbb{E}[X_v \mid Y_v]] = \mathrm{Var}_{Y_v}[pY_v] = p^2 \mathrm{Var}[Y_v].$$

It follows that:

$$\mathrm{Var}[X_v] = p\mathrm{Var}[Y_v] + p(1-p)\mathbb{E}^2[Y_v].$$

$$\begin{aligned}\mathrm{Var}[Y_v] =& \mathrm{Var}[(S_A^{c_A}(v) + qI_A^{c_A}(v)) \cdot (S_B^{c_B}(v) + qI_B^{c_B}(v))] \\ =& \mathbb{E}[(S_A^{c_A}(v) + qI_A^{c_A}(v))^2]\mathbb{E}[S_B^{c_B}(v) + qI_B^{c_B}(v))^2] \\ & - \mathbb{E}^2[(S_A^{c_A}(v) + qI_A^{c_A}(v))]\mathbb{E}^2[S_B^{c_B}(v) + qI_B^{c_B}(v))]\end{aligned}$$

We have that:

$$\mathbb{E}[(S_A^{c_A}(v)+qI_A^{c_A}(v))^2] = q\left(a_v^{c_A}-\frac{a_v^{c_A}}{a_v}\right)+ q^2\left((a_v^{c_A})^2-a_v^{c_A}+\frac{a_v^{c_A}}{a_v}\right), \tag{12}$$

and

$$\mathbb{E}[(S_B^{c_B}(v)+qI_B^{c_B}(v))^2] = q\left(b_v^{c_B}-\frac{b_v^{c_B}}{b_v}\right)+q^2\left((b_v^{c_B})^2-b_v^{c_B}+\frac{b_v^{c_B}}{b_v}\right).$$

It follows that:

$$\mathrm{Var}[\hat{J}] = \sum_{v:a_vb_v\neq 0}\mathrm{Var}[\hat{J}_v] = \sum_{v:a_vb_v\neq 0}\frac{1}{p}\sigma_1^2(v)+\sigma_2^2(v).$$

□

## A.5 Optimal Parameters

$$\begin{aligned}\text{minimize}\quad & \sum_v \frac{1}{p}\left(\frac{1}{q}(a_v-1)+a_v^2\right)\\ \text{s.t.}\quad & n_A+n_B=n\\ & n_A = p\cdot(q(\|a\|_1-\|a\|_0)+\|a\|_0)\\ & n_B = p\|b\|_1\\ & p,q\in[0,1]\end{aligned}$$

We have that

$$\begin{aligned}&\sum_v \frac{1}{p}\left(\frac{1}{q}(a_v-1)+a_v^2\right)\\ =&\left(\frac{1}{q}(\|a\|_1-\|a\|_0)+\|a\|_2^2-\|a\|_1+\|a\|_0\right)\\ &\cdot(\|b\|_1+\|a\|_0+q(\|a\|_1-\|a\|_0))\\ =&\frac{1}{q}(\|a\|_1-\|a\|_0)\cdot(\|b\|_1+\|a\|_0)+\|a\|_2^2(\|a\|_1-\|a\|_0)q+c\end{aligned}$$

According to Cauthy–Schwarz Inequality, it is minimized when $q=\sqrt{\frac{\|a\|_0+\|b\|_1}{\|a\|_2^2-\|a\|_1+\|a\|_0}}$. Considering that $p,q\in[0,1]$, we have $q\in\left[\frac{n-\|a\|_0-\|b\|_1}{\|a\|_1-\|a\|_0},1\right]$. Hence, the optimal parameters are

$$q=\begin{cases}\frac{n-\|a\|_0-\|b\|_1}{\|a\|_1-\|a\|_0} & \text{if } \sqrt{\frac{\|a\|_0+\|b\|_1}{\|a\|_2^2-\|a\|_1+\|a\|_0}}<\frac{n-\|a\|_0-\|b\|_1}{\|a\|_1-\|a\|_0};\\ 1, & \text{if } \sqrt{\frac{\|a\|_0+\|b\|_1}{\|a\|_2^2-\|a\|_1+\|a\|_0}}>1;\\ \sqrt{\frac{\|a\|_0+\|b\|_1}{\|a\|_2^2-\|a\|_1+\|a\|_0}} & \text{otherwise,}\end{cases}$$

and

$$p=\frac{n}{\|b\|_1+\|a\|_0+q(\|a\|_1-\|a\|_0)}.$$

## A.6 Proof of Unbiasedness of $\hat{\psi}$

Proof. For each $v$ such that $h(v)<p$, we have

$$\mathbb{E}[\hat{\psi}_v] = p\mathbb{E}[\hat{\psi}_v \mid h(v)<p] = p\frac{1}{a_vb_v}\mathbb{E}\left[\sum_{i,j}X_{i,j}\mid t_a,t_b \text{ are sentries.}\right]$$

where

$$X_{i,j}=\begin{cases}\frac{f(t_i,t_j)}{p_ip_j} & \text{with probability } p_ip_j;\\ 0, & \text{otherwise.}\end{cases}$$

, where $t_i$( resp. $t_j$) denotes the $i$-th (resp. $j$-th) tuple in $A(v)$( resp. $B(v)$) and $p_i$ (resp. $p_j$) is the sampling probability for $t_i$ (resp. $t_j$) in second level. More precisely, the sampling probability is 1 for sentry and is $q$ for other tuples in second level.

$$\begin{aligned}\mathbb{E}[\hat{\psi}_v] &= p\mathbb{E}\left[\sum_{i,j}X_{i,j}\mid t_a,t_b \text{ are sentries.}\right]\\ &= p\sum_{i,j}\mathbb{E}[X_{i,j}\mid t_a,t_b \text{ are sentries.}]\\ &= p\sum_{i,j}f(t_i,t_j) = pf(A(v),B(v)).\end{aligned}$$

It follows that $\mathbb{E}[\hat{\psi}]=f(A,B)$. □

## A.7 Proof of Unbiasedness of $\widehat{\mathrm{Var}}[\hat{J}_{2lvl}]$

Proof. We only need to proof the unbiasedness of $\widehat{a_v^{c_A}}$ and $\widehat{(a_v^{c_A})^2}$ conditioned on $v$ is sampled. Note that the following proof are under the condition that $v$ is sampled.

The unbiasedness of $\widehat{a_v^{c_A}}$ has been proved in Appendix A.3.

For the unbiasedness of $\widehat{(a_v^{c_A})^2}$, we have $(a_v^{c_A})^2 = \frac{1}{q^2}\left(\mathbb{E}[(S_A^{c_A}(v)+qI_A^{c_A}(v))^2]-q\left(a_v^{c_A}-\frac{a_v^{c_A}}{a_v}\right)+q^2\left(a_v^{c_A}-\frac{a_v^{c_A}}{a_v}\right)\right)$, according to Equation 12.

And we have shown that $\mathbb{E}\left[\frac{|S_A^{c_A}(v)|}{q}\mid h(v)<p\right] = a_v^{c_A}-\frac{a_v^{c_A}}{a_v}$ in Appendix A.3.

Recall that

$$\widehat{(a_v^{c_A})^2} = \left(\frac{S_A^{c_A}(v)}{q}+I_A^{c_A}(v)\right)^2+\left(1-\frac{1}{q}\right)\frac{|S_A^{c_A}(v)|}{q}.$$

According to the linearity of expectation and the upper three equations, we finally have $E[\widehat{(a_v^{c_A})^2}] = (a_v^{c_A})^2$.

$$\mathbb{E}[(S_A^{c_A}(v)+qI_A^{c_A}(v))^2] = q\left(a_v^{c_A}-\frac{a_v^{c_A}}{a_v}\right)+q^2\left((a_v^{c_A})^2-a_v^{c_A}+\frac{a_v^{c_A}}{a_v}\right), \tag{13}$$

□

## A.8 Proof of Variance of $\hat{\psi}$

Proof. Because $h$ is pair-wise independent, we have $\mathrm{Var}[\hat{\psi}] = \sum_{v:a_vb_v\neq 0}\mathrm{Var}[\hat{\psi}_v]$. Now we only need to focus on $v$ with $a_vb_v\neq 0$ and $h(v)<p$. As defined before, let $X_v=\sum_{i,j}X_{i,j}$ and we know that

$$\hat{\psi}_v=\begin{cases}X_v, & h(v)<p\\ 0, & \text{otherwise.}\end{cases}$$

By law of Total Variance, we have $\mathrm{Var}[\hat{\psi}_v] = \frac{1}{p}\mathrm{Var}[X_v]+(\frac{1}{p}-1)\mathbb{E}^2[X_v]$.

By law of Total Variance, we have:

$$\begin{aligned}\mathrm{Var}[X_v] =&\mathbb{E}\left[\mathrm{Var}[X_v\mid t_a,t_b \text{ are sentries.}]\right]\\ &+\mathrm{Var}[\mathbb{E}[X_v\mid t_a,t_b \text{ are sentries.}]]\\ =&\mathbb{E}[\mathrm{Var}[X_v\mid t_a,t_b \text{ are sentries.}]]\\ =&\frac{1}{a_vb_v}\sum_{t_a\in A(v),t_b\in B(v)}\mathrm{Var}[X_v\mid t_a,t_b \text{ are sentries.}]\end{aligned} \tag{14}$$

Define $Y_v$ as $X_v \mid \{t_a, t_b \text{ are sentries.}\}$ and define $Y_{i,j}$ as $X_{i,j} \mid \{t_a, t_b \text{ are sentries.}\}$.

$$\begin{aligned}\mathrm{Var}[Y_v] =& \mathrm{Var}\left[\sum_{t_i \in A(v), t_j \in B(v)} Y_{i,j}\right] \\ =& \sum_{t_i \in A(v), t_j \in B(v)} \mathrm{Var}[Y_{i,j}] + \\ & \sum_{t_i \in A(v), t_j, t_k \in B(v), t_k \neq t_j} \mathrm{Cov}(Y_{i,j}, Y_{i,k}) \\ & + \sum_{t_i, t_k \in A(v), t_k \neq t_i, t_j \in B(v),} \mathrm{Cov}(Y_{i,j}, Y_{k,j}).\end{aligned}$$

We know that

$$\mathrm{Var}[Y_{i,j}] = \left(\frac{1}{p_i p_j} - 1\right) f^2(t_i, t_j),$$

$$\mathrm{Cov}(Y_{i,j}, Y_{i,k}) = \left(\frac{1}{p_i} - 1\right) f(t_i, t_j) f(t_i, t_k),$$

$$\mathrm{Cov}(Y_{i,j}, Y_{k,j}) = \left(\frac{1}{p_j} - 1\right) f(t_i, t_j) f(t_k, t_j).$$

It follows that

$$\begin{aligned}&\mathrm{Var}[X_v \mid t_a, t_b \text{ are sentries.}] \\ =& \left(\frac{1}{q^2} - 1\right) \sum_{t_i \in A(v) \setminus \{t_a\}, t_j \in B(v) \setminus \{t_b\}} f^2(t_i, t_j) + \\ & \left(\frac{1}{q} - 1\right) \sum_{t_i \in A(v) \setminus \{t_a\}} f^2(t_i, t_b) + \left(\frac{1}{q} - 1\right) \sum_{t_j \in B(v) \setminus \{t_b\}} f^2(t_a, t_j) + \\ & \sum_{t_i \in A(v), t_j \in B(v) \setminus \{t_b\}} \left(\frac{1}{q} - 1\right) f(t_i, t_j) f(t_i, B(v) \setminus \{t_j\}) + \\ & \sum_{t_i \in A(v) \setminus \{t_a\}, t_j \in B(v)} \left(\frac{1}{q} - 1\right) f(t_i, t_j) f(A(v) \setminus \{t_i\}, t_j)\end{aligned} \tag{15}$$

By using Equation 14 and 15, we can derive Equation 16.

□

# B. HANDLING OTHER AGGREGATES

In this section, we show how two-level sampling can be extended to handle more general aggregation queries over joins. An example is the following query on the TPC-H benchmark data, which finds the total revenue of all orders in a certain period from a certain supplier.

```
SELECT SUM(l_extendedprice * (1 - l_discount))
FROM orders, lineitem
WHERE l_orderkey = o_orderkey
      AND o_orderdate >= '2016-06-01'
      AND o_orderdate <= '2016-06-20'
      AND l_suppkey = 3
```

## B.1 Estimating SUM

When the aggregate is `SUM`, the problem can be more generally defined as follows. Let $f(t_a, t_b)$ be a real-valued function defined on a tuple $t_a \in A$ and a tuple $t_b \in B$. We wish to estimate $\psi = \sum_{t_a \in A, t_b \in B} f(t_a, t_b)$ using the sample. Note that since we allow a general $f$, we no longer need to consider the selection predicates explicitly, because we can simply define $f(t_a, t_b) = 0$ if $t_a$ or $t_b$ does not satisfy their respective predicates.

For the ease of presentation, we introduce the notation $f(X, Y) = \sum_{t_a \in X, t_b \in Y} f(t_a, t_b)$, thus our target is just $\psi = f(A, B)$. We also write $f(\{t\}, Y)$ simply as $f(t, Y)$, and $f(X, \{t\})$ as $f(X, t)$.

An unbiased estimator for the `SUM` is

$$\hat{\psi} = \sum_v \frac{1}{p_v} \hat{\psi_v},$$

where $\hat{\psi_v}$ is defined as:

$$\hat{\psi_v} = \begin{cases} \frac{1}{q^2} f(S_A(v), S_B(v)) + \frac{1}{q} f(S_A(v), s_B(v)) \\ + \frac{1}{q} f(s_A(v), S_B(v)) + f(s_A(v), s_B(v)), \\ \qquad\qquad \text{if } v \text{ is sampled,} \\ 0, \qquad\qquad \text{otherwise.} \end{cases}$$

We show that $\hat{\psi}$ is unbiased Appendix A.6.

Its variance is:

$$\mathrm{Var}[\hat{\psi}] = \sum_{v: a_v b_v \neq 0} \mathrm{Var}[\hat{\psi_v}] = \sum_{v: a_v b_v \neq 0} \frac{1}{p_v} \sigma_1^2(v) + \sigma_2^2(v), \tag{16}$$

where

$$\begin{aligned}& a_v b_v \sigma_1^2(v) = \\ & \left(\frac{1}{q^2} - \frac{2}{q} + 1\right)(a_v - 1)(b_v - 1) \sum_{t_a \in A(v), t_b \in B(v)} f(t_a, t_b)^2 \\ & + \left(\frac{1}{q} - 1\right) b_v (a_v - 1) \sum_{t_a \in A(v)} f(t_a, B(v))^2 \\ & + \left(\frac{1}{q} - 1\right) a_v (b_v - 1) \sum_{t_b \in B(v)} f(A(v), t_b)^2,\end{aligned}$$

and $\sigma_2^2(v) = \left(\frac{1}{p} - 1\right) f(A(v), B(v))^2$.

We give the detailed derivation of the variance in Appendix A.8. Note that when setting all $f(t_a, t_b) = 1$, i.e., the `SUM` degenerates into `COUNT`, and the formula of the variance becomes exactly the same as (1).

## B.2 Estimating AVG

We can estimate the `AVG` simply using the estimator of the `SUM` divided by the estimator of the `COUNT`:

$$\hat{\mu} = \frac{\hat{\psi}}{\hat{j}}.$$

This estimator is not unbiased, but it converges to the true `AVG` almost surely as $J \to \infty$.

Other aggregates that can be expressed using `SUM` and `COUNT` can also be handled. For example, `STDDEV` can be computed using the `SUM` of squares a given attribute, together with `COUNT` and `AVG`. Confidence intervals can be computed using the delta method, similar as in [12]. We omit the details.