Confidence Bounds for Sampling-Based GROUP BY Estimates

FEI XU, CHRISTOPHER JERMAINE, and ALIN DOBRA
University of Florida, Gainesville

Sampling is now a very important data management tool, to such an extent that an interface for database sampling is included in the latest SQL standard. In this article we reconsider in depth what at first may seem like a very simple problem—computing the error of a sampling-based guess for the answer to a GROUP BY query over a multitable join. The difficulty when sampling for the answer to such a query is that the same sample will be used to guess the result of the query for each group, which induces correlations among the estimates. Thus, from a statistical point-of-view it is very problematic and even dangerous to use traditional methods such as confidence intervals for communicating estimate accuracy to the user. We explore ways to address this problem, and pay particular attention to the computational aspects of computing "safe" confidence intervals.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Query processing, Relational databases*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Multivariate statistics*

General Terms: Algorithms, Theory, Reliability

Additional Key Words and Phrases: Approximate query processing, multiple hypothesis testing, sampling

ACM Reference Format:

Xu, F., Jermaine, C., and Dobra, A. 2008. Confidence bounds for sampling-based GROUP BY estimates. ACM Trans. Datab. Syst. 33, 3, Article 16 (August 2008), 44 pages. DOI = 10.1145/1386118.1386122 http://doi.acm.org/10.1145/1386118.1386122

16

1. INTRODUCTION

Over the last decade, sampling and other statistical approximation techniques have been widely proposed as a way to speed up query processing in relational database systems. Dozens of papers have been written on the topic; a few a listed in the references section of this article [Acharya et al. 1999a; Dobra

Authors' addresses: Department of Computer and Information Sciences and Engineering, University of Florida, Gainesville, FL 32611-6120; email: cjermaine@cise.ufl.edu.

et al. 2002; Haas and Hellerstein 1999; Ganti et al. 2000; Chaudhuri et al. 2001; Jermaine et al. 2005; Gibbons and Matias 1998]. The attractiveness of statistical approximation is clear: during analytic query processing where most queries are statistical in nature, a statistical synopsis of the database can be used to provide a good guess as to the actual query answer in a fraction of the time that may be required to compute an exact query answer. The increase in query processing speed is related to the much smaller size of the synopsis, which generally decreases both the CPU time required to answer the query, as well as the amount of data that must be loaded from disk into main memory.

The obvious downside of this speed-up is some inaccuracy in the approximate query result. Fortunately, while end-users may be hesitant to accept nonprecise answers, the pitfalls associated with returning an approximate answer can be obviated using standard statistical techniques. For example, it is often possible to accompany the guess with accuracy guarantees in the form of probabilistic confidence bounds [Sarndal et al. 1992]: "With probability p, the final answer to the query is between l and h."

The specific problem that we consider in this article is that of developing "safe" confidence bounds for a SUM/GROUP BY query whose answer estimated using a sample of the database. Sampling has quite a few advantages compared to other estimation methods, including the fact that a simple random sample can be used to evaluate almost any multitable join, regardless of the selection and/or join predicates present in the query. We are primarily interested in an application where a small sample of a large database has been precomputed or is obtained in response to a specific SUM/GROUP BY query, though our methods are also applicable to online estimation [Haas and Hellerstein 1999].

The fundamental motivation behind this article is that classic, univariate statistical analyses are not applicable to providing sampling-based confidence bounds for even the simplest GROUP BY queries—an issue that has been ignored in the database literature to date. The reason for this is that the same sample is used to evaluate the underlying query for each group, resulting in correlations among the various, groupwise estimates. Obtaining independent samples for each group is not a practical option. With 100 groups, an independent, 1% database sample for each group (resulting in 100 different 1% samples) would expectedly cover most of the database tuples, and then negate the benefit of sampling.

To illustrate the problem in detail, we first consider the query: "What were the total sales per region in 2004?" This query may be written in SQL as

```
SELECT SUM(p.COST), s.REGION
FROM PRODUCT p, SALES s
WHERE p.PROD = s.PROD AND s.YR = 2004
GROUP BY s.REGION
```

Imagine that the answer to the query is guessed at by using random samples from the following two database tables:

PRODUCT		SALES		
PROD	COST	PROD	YR	Region
thingy	$10	thingy	2004	Asia
gadget	$2	widget	2003	Asia
widget	$3	thingy	2004	USA
dohicky	$4	gadget	2004	USA
		widget	2003	USA
		thingy	2004	Europe
		dohicky	2004	Europe

The random samples are used as input into a sampling-based estimator for the answer to the query [Haas and Hellerstein 1999]. In order to apply the simplest estimator, the sample from each of the two tables would be joined, and the result scaled up by the inverse of each sampling fraction. The problem we face is that the same sample of relation PRODUCT is effectively used to simultaneously answer three queries, one for each group—in this case, there is one group associated with the region Europe, one with the region USA, and one with the region Asia. If the first estimate was inaccurate, then the second and third estimates are likely to be inaccurate as well. Note that each of the three regions described in the database has experienced a sale of the thingy product in 2004, and thingy is the most expensive product available. As a result, if thingy has not been sampled from the PRODUCT table, it is likely that the estimates for Europe, Asia, and USA will simultaneously be too low.

Such correlation may be of great concern if an end-user will make decisions based upon the results of several correlated approximations. A user may ask a query that returns answers for 10 different groups, and receive 10 answers with 90% confidence bounds for each. The user may then assume that since each estimate is correct with 90% probability, that most of the approximations will be correct. However, if the various estimates are strongly correlated, the statistical reality may be that there is a 10% chance that *each and every one* of the 10 approximations is incorrect.

1.1 Our Contributions

While such issues have been considered in statistics for decades [Miller 1981], they have been ignored in the data management literature. This article makes the following specific technical contributions:

—We identify the abstract problem of predicting the number of estimates that are "incorrect" in the sense that the true query answers are outside of the provided confidence intervals. We call this problem the *sum-inference* (SI) problem.

—We demonstrate the use of the SI problem for providing "safe" confidence bounds for GROUP BY queries.
—We provide three possible solutions to the SI problem, and thus to GROUP BY query estimation. One is a nonparametric Monte Carlo solution, one is a parametric solution, and one is a parametric Monte Carlo solution. We explore the benefits and shortcomings of the various solutions.
—We carry out experiments to validate the theoretical developments and show that SI-based solutions are computationally practical.

A final contribution is that, while the article presents a detailed examination of how to handle sampling-based estimates over SUM-based GROUP BY queries, the techniques we consider are quite general and can be applied more widely to other estimators. In fact, one reason for defining the SI problem and studying it in a general setting (outside of the context of GROUP BY queries alone) is the potential for applying the SI problem to other types of queries. We elaborate on how the SI problem might be applied more widely in Section 7.

1.2 Article Organization

In Section 2 we give a brief introduction to terms and techniques from statistics that are used throughout the article. The technical contribution starts in Section 3 where we introduce the GROUP BY estimation problem. The major technical difficulty in designing estimators for GROUP BY queries is dealing with simultaneous statistical inference. To this end, we introduce the SI problem—which isolates the statistical analysis of simultaneous inference from database estimation—and show how it can be applied to GROUP BY. In Section 4 we propose three methods to solve the SI problem. Section 5 considers computational issues, such as how to efficiently compute the covariance of two estimates. Section 6 presents results of an empirical study of the proposed techniques. We indicate other possible uses of the SI problem in Section 7, discuss related work in Section 8, then conclude in Section 9.

The SI problem assumes that the covariances of the various estimates are available. Obtaining such covariances is often a challenging problem by itself. Thus, Appendix A of the article specifically considers the problem of deriving and estimating covariances over a set of arbitrary join queries computed over samples from one or more database tables.

The notation used throughout (except the Appendix, which is self-contained) is summarized in Table I.

2. BACKGROUND: ESTIMATORS AND CORRELATIONS

In this section we discuss a few statistical preliminaries.

2.1 Estimators and their Properties

Let M be a parameter of interest that we are trying to estimate. In this article the "parameter of interest" is typically the final answer to a query. An estimate $\overline{M}$ of M is a single number computed from a sample or other statistical

Table I. Variables and Notations Used in this Article

μ	Mean. When it is a vector, μ_i is ith component.
σ^2	Variance of one dimensional random variables.
Σ^2	Covariance matrix.
$\mathrm{Cov}(X, Y)$	Covariance between random variables X and Y.
$\widetilde{M}$	Estimator of parameter M.
$X_{i,\ldots}$	Bernoulli(zero/one) random variable for component $i, \ldots$
N	Multivariate normal random variable in an SI problem. N_i indicates the ith component.
n	Number of estimators in the SI problem.
$p_i(x)$	Probability density function of N_i in N.
$p_{i,j}(x, y)$	Joint probability density function of N_i and N_j in N.
l	Lower bound. When a vector, l_i indicates the ith component.
h	Upper bound. When a vector, h_i indicates the ith component.
s	Score vector, s_i indicates the ith element.
ζ	The random variable of an SI problem.
F	The distribution of ζ.
ρ	Coefficient for simultaneous confidence intervals.
$\widetilde{F}$	The approximate distribution of ζ.
ϵ	A small threshold.
$I(A)$	Indicator function for predicate A. $I(A) = 1$ if A is true, and 0 otherwise.
m	Total number of non-diagonal entries in a covariance matrix.
T	The set of all non-diagonal entries in a covariance matrix.
S	A sample from T.
$V_{..}$	Components used in computing covariance matrix entries.

summary, which serves as a guess of the value of M. Note that if the estimation process is repeated many times, many different estimates will be observed. These estimates are typically characterized using a random variable. The random variable whose observed value is used to estimate M is called an *estimator* and denoted $\widetilde{M}$.[1]

For example, consider a SUM query over a single database table. Specifically,

```
SELECT SUM(f(R))
FROM R
```

The function f can encode any mathematical function over tuples from R, and can encode an arbitrary selection predicate. Also, a query of this form can be extended to handle a clause of the form GROUP BY $att1, att2, \ldots$ by first identifying all of the n groups induced by the GROUP BY clause, and then rerunning the query n times. During the ith run, $f(R)$ is modified to accept only tuples from the ith group (f evaluates to zero for any tuple that is not from that group). For example, imagine that we add a clause GROUP BY gender to the above query, and gender has values male and female. We could simply define a function f_{male} where $f_{male}(R) = f(R)$ if R.gender is male and zero otherwise, as well as a function f_{female} where $f_{female}(R) = f(R)$ if R.gender is female. Then, running the query twice (once with each function) gives an answer to the GROUP BY query.

[1]When M is a parameter to be estimated, we will use $\widetilde{M}$ to denote the associated estimator; we will also make use of the standard convention of using a capital letter such as N to denote a random variable.

Given such a query, the final answer can be computed as

$$M = \sum_{r \in R} f(r) \tag{1}$$

The simplest estimator for M may be random sampling without replacement. To estimate M using this method, denote by R' the set of samples from R and let n_R denote the sample's size. If N_R is the number of tuples in relation R, we can then express $\widetilde{M}$ by introducing a set of Bernoulli (zero/one) random variables that indicate whether tuples from R are not/are in R'. Let X_k be the variable that indicates whether the kth tuple from R is in R'. With this, a natural estimator for the aggregate query would be

$$\widetilde{M} = \frac{N_R}{n_R} \sum_{k=1}^{N_R} X_k f(k) \tag{2}$$

This estimator simply sums the elements in the sample set and scales the result according to the inverse of the sampling fraction.

The estimator $\widetilde{M}$ is called an *unbiased estimator* because $E[\widetilde{M}] = M$, where $E[\widetilde{M}]$ is the expected value of the estimator. That is, if the estimator were used many times in succession, the average would be exactly equal to the true answer.

For an unbiased estimator, the *variance* is usually used as the criteria to indicate how good the estimator $\widetilde{M}$ is, where $\sigma^2(\widetilde{M}) = E[\widetilde{M}^2] - E^2[\widetilde{M}]$. The variance of this estimator is

$$\sigma^2(\widetilde{M}) = \frac{N_R}{n_R} \frac{n_R - 1}{N_R - 1} \sum_{k=1}^{N_R} \sum_{l=1, l \neq k}^{N_R} f(k) f(l) - \frac{N_R}{n_R} \sum_{k=1}^{N_R} f^2(k) \tag{3}$$

If an unbiased estimator has small variance, then there is only a small chance that its value is far from the true value for the parameter M.

This very simple estimator can be extended to a join over an arbitrarily large number of tables in a straightforward way. We refer to the resulting estimator as the *Haas-Hellerstein* estimator [Haas and Hellerstein 1999]. This generalization is the estimator used by the UC Berkeley Control project [Hellerstein et al. 1999], and if one samples directly from a precomputed join, the simpler, one-table version of the Haas-Hellerstein estimator is the estimator used by the AQUA project [Acharya et al. 1999b] by their join synopsis method [Acharya et al. 1999a].

To extend the estimator, we first assume that $T_1, T_2, \ldots, T_k$ are k database tables and that we wish to guess an answer to a query of the form

```
SELECT SUM(f(T1, T2, ... , Tk))
FROM T1, T2, ... , Tk
WHERE pred(T1, T2, ... , Tk)
```

We let $g(T_1, \ldots, T_k)$ be a function that returns $f(T_1, \ldots, T_k)$ if *pred* is true, 0 otherwise. We assume the numbers of tuples in these tables are $N_1, N_2, \ldots, N_k$, and the sample sizes are $n_1, n_2, \ldots, n_k$, respectively. We define

Bernoulli random variables $X_{w1}, \ldots X_{wk}$ that govern whether or not the wth tuple from $T_1, \ldots, T_k$ are sampled, respectively. The following then serves as the Haas-Hellerstein estimator for the sum of f:

$$\widetilde{M} = \frac{N_{1\ldots k}}{n_1 \ldots n_k} \sum_{w1,\ldots,wk} X_{w1} \ldots X_{wk}\, g(w_1, \ldots, w_k) \tag{4}$$

2.2 Errors and Confidence Intervals

A *confidence interval* (CI) for M is an interval $[l, h]$ with a user-defined probability $1 - p$ associated with it. This is the probability that the interval that is chosen actually contains the parameter of interest. Generally, $1 - p$ is called the *confidence level* and p is called the Type-I error of the estimator $\widetilde{M}$.

Properties such as an estimator's variance and unbiasedness can be used to compute a CI. Define

$$err = \widetilde{M} - M \tag{5}$$

to be the error of some estimator $\widetilde{M}$. Since the estimator is unbiased, the mean of err is 0 and the variance is equal to $\sigma^2(\widetilde{M})$. According to the Central Limit theorem, the distribution of err is asymptotically normal for the Haas-Hellerstein estimator considered in this article, as long as a large-enough database sample is used (see Haas and Hellerstein, Sections 5.2.1 and 6 for more details [Haas and Hellerstein 1999]). Since (as discussed in the previous section) a GROUP BY query can be encoded as a number of Haas-Hellerstein estimators, the estimate for each group in a SUM-based GROUP BY query is also normally distributed.

If the error follows a normal distribution, a CI at confidence level $1 - p$ is $[-z_{\frac{p}{2}}\sigma(\widetilde{M}), z_{\frac{p}{2}}\sigma(\widetilde{M})]$ where $z_{\frac{p}{2}}$ is a coefficient that is determined by the cumulative density function for the normal distribution and $\sigma(\widetilde{M})$ is the standard deviation of $\widetilde{M}$, which is the square root of the variance $\sigma^2(\widetilde{M})$. Thus, the probability that $\overline{M} - M$ is in this interval is $1 - p$. By manipulating this expression, we obtain

$$Pr\left[M \in \left[\overline{M} - z_{\frac{p}{2}}\sigma(\widetilde{M}), \overline{M} + z_{\frac{p}{2}}\sigma(\widetilde{M})\right]\right] = 1 - p \tag{6}$$

Therefore,

$$\left[\overline{M} - z_{\frac{p}{2}}\sigma(\widetilde{M}), \overline{M} + z_{\frac{p}{2}}\sigma(\widetilde{M})\right] \tag{7}$$

is a CI for M with confidence level $1 - p$.

2.3 Correlations

This article considers how correlations between estimates computed using the same synopsis may be taken into account. In general, the *covariance* between two variables provides a measure of the correlation between them. The covariance for two random variables X and Y is defined as

$$\mathrm{Cov}(X, Y) = E[XY] - E[X]E[Y] \tag{8}$$

For uncorrelated variables, the covariance between them is 0. If $\mathrm{Cov}(X, Y) > 0$, then Y tends to increase as X increases, and if $\mathrm{Cov}(X, Y) < 0$, then Y tends

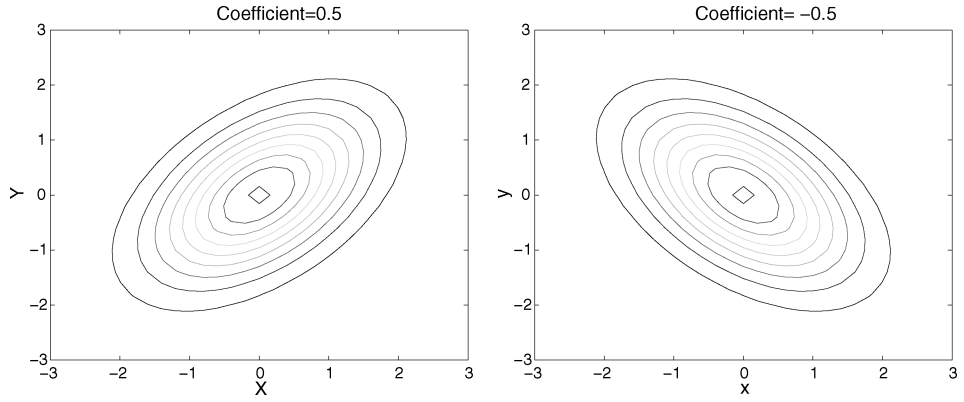

Fig. 1. Standard bivariate normal(coef = 0.5(left) and −0.5(right)).

to decrease as X increases. Figure 1 shows the contours of standard bivariate normal distribution with covariance 0.5 and −0.5, respectively. From the figure, we see that for a negative covariance the distribution's contours are a set of ellipses, the major axes of which fall into the second and fourth quadrants, respectively. This indicates that Y tends to decrease as X increases. For the positive covariance the contours are a set of ellipses, the major axes of which fall into the first and third quadrants. This indicates that Y tends to increase as X increases.

In general, such multivariate distributions are specified using a covariance matrix Σ and a mean vector μ. For the joint distribution of the error of a set of estimators $\widetilde{M}_i$, $\mu_i = 0$ and $\Sigma_{i,j} = \mathrm{Cov}(\widetilde{M}_i, \widetilde{M}_j)$.

3. THE SI PROBLEM AND GROUP BY QUERIES

This particular section first describes what information or statistics should be communicated to a user in the case of an approximate GROUP BY answer, so that he or she is fully aware of the accuracy of an approximation. We then formally define the abstract problem that must be solved in order to compute this information, which we call the *SI Problem*. The section concludes by describing how a solution to the SI problem can be used to compute "safe" GROUP BY bounds.

3.1 GROUP BY and Multiple Inference

A GROUP BY query may be viewed as a large set of individual queries asked simultaneously, one for each group. Unfortunately, in this situation a traditional confidence bound such as

"With probability p, the answer to group i is within the range l_i to h_i."

has only a very narrow correct interpretation. As soon as a user looks at the results for a second group j, statistically speaking the user needs to "forget" that she or he ever saw the bounds for group i and consider group j in complete isolation! When two or more groups are considered together or compared with one another, the accuracy parameter p is meaningless, and hence dangerous.

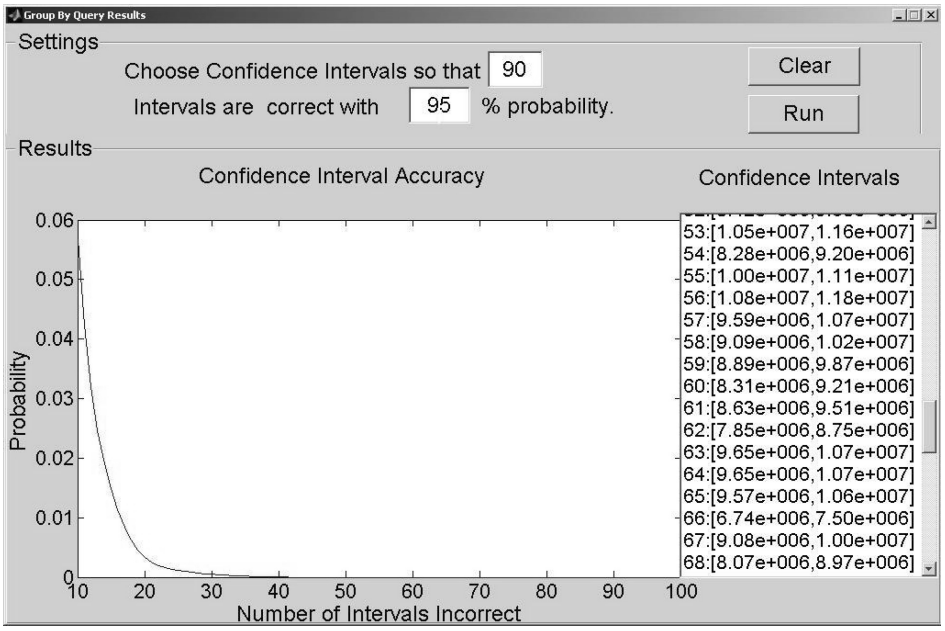

Fig. 2. A GUI for GROUP BY query.

Rather than viewing confidence bounds individually, we propose to extend the traditional guarantee and instead present a guarantee taking the form

"With probability p, at least k of n groups are within the ranges (l_1 to h_1), (l_2 to h_2), ..., and (l_n to h_n), respectively."

In the case where $n = k$, these new bounds are exactly equivalent to the classical method from statistics for dealing with simultaneous estimates: the bounds are altered so that, with high probability, each one of the estimates are within the specified ranges [Miller 1981]. However, statisticians have begun to acknowledge that this is overly restrictive [Benjamini and Hochberg 1995]. For example, the classic method for controlling simultaneous error is to assume that the error is additive (this is known is the Bonferroni correction or the union bound). In order to provide 100 bounds that are all correct with probability 0.99, each one would be correct with probability 0.9999. This can result in needlessly wide confidence bounds. Our generalization of the bound to include the parameter k allows the user to control the extent to which group-wise correlations can affect the number of incorrect estimates. No longer is a bound derived by computing the probability that it is incorrect; rather, it is derived by computing the probability that a set of bounds are *simultaneously* incorrect.

In order to make this a bit clearer, Figure 2 gives a primitive user interface that could be used to present such bounds to the user for a GROUP BY query. The scroll box at the right of the figure gives a confidence bound for each of the 100 groups that are found in the query result. Using the interface, the user chooses p and k, and the bounds are computed accordingly. In addition, the

user is supplied with a plot that shows how likely it is that $n - k$ or more of the bounds computed are wrong. For example, Figure 2 shows that there is a very small but nonzero chance that at least 30 of the 100 bounds given are incorrect, and there is only about a 0.003 chance that at least 20 of the 100 bounds are incorrect. Since these are relatively small values, the user can be sure that (in this case) the chance of a catastrophic error in the query result is small.

3.2 The SI Problem

The issue at the heart of this article is how such simultaneous bounds can be computed in an efficient, principled fashion. This section describes the *Sum-Inference problem*, or SI problem for short, which can be used to quantify the extent to which correlation among statistical estimators for database queries can cause many estimates derived from the same synopsis to be incorrect. This problem will form the basis of our computation of safe `GROUP BY` bounds. The main reason for formulating the abstract SI problem instead of directly introducing and analyzing estimates for `GROUP BY` queries is the fact that, as explained in Section 7, there is the potential that the SI problem can be applied to other types of queries. A side benefit is that the analysis `GROUP BY` queries can be dissociated from the particular method used to estimate the total for each individual group.

The SI problem models the situation where we have n random variables. The random variable N_i corresponds to a possible answer to the the ith query or group. We will assume that N_i is normally distributed (which is true asymptotically for a sample-based `SUM`/`GROUP BY` queries due to the Central Limit theorem; see Section 2 for details). Also given as input into the SI problem is a set of n valid ranges, with one range for each random variable; each range corresponds to a confidence bound on the answer at each query. The ith range is defined by the bounds l_i and h_i. If the ith range or bound is wrong – that is, if N_i happens to fall outside of the range from l_i to h_i – then we incur a penalty or score s_i. If we are simply trying to count the number of incorrect ranges, then $s_i = 1$; in general, s_i may take a different value. We then define the following random variable:

$$\zeta = \sum_{i=1}^{100} I(N_i \notin [l_i, h_i])s_i \tag{9}$$

In this expression, I is a function that returns 1 if its boolean argument evaluates to *true* and 0 if it is *false*. If we define a distribution function F such that $F[k]$ gives the probability that the above expression evaluates to k, then given F one can check the probability that the total penalty (or total number of incorrect confidence bounds) meets or exceeds k, which can be used as a solid indicator as to the accuracy of the given bounds.

Formally, the SI problem is defined as follows:

The Sum-Inference (SI) problem. Given

—a multidimensional normal random variable $N = \langle N_1, N_2, \ldots, N_n \rangle$ with mean vector μ and covariance matrix Σ,

—a vector s of n scores,

—a vector l of n lower bounds, and

—a vector h of n upper bounds

the SI problem is the problem of inferring the distribution function F, where $F[k] = P(\sum_i I(N_i \notin [l_i, h_i])s_i = k)$, i.e. $F[k]$ is the probability of observing a total penalty of k due to incorrect intervals.

Unfortunately, it is easy to show that it is likely impossible to infer characteristics of the distribution of F directly:

LEMMA 3.1. Hardness of the SI problem. *If $F[k]$ can always be computed in polynomial time, then $P = NP$.*

PROOF. Given a set of n integers $S = \{e_1, e_2, \ldots, e_n\}$ and a number k, the subset-sum problem asks whether there is a subset of S such that the summation of this subset is k. This is known to be NP-complete. We reduce this problem to the SI problem. To solve an instance of the subset-sum problem using the SI problem, we construct a random variable $N = \langle N_1, N_2, \ldots, N_n \rangle$, where $N_1, N_2, \ldots, N_n$ are independent, normally distributed random variables with mean 0 and variance 1. We also construct a vector of scores $s = [e_1, e_2, \ldots, e_n]^T$ such that each element in S is in this score vector once and only once. We also construct two vectors of lower bounds and upper bounds l and h where the elements are the extent of 95%, two-sided confidence intervals. We then solve the resulting SI problem.

Since each confidence interval is correct with probability 95%, there is a 5% chance that each random variable is outside the interval, thus there is a nonzero probability that any subset of incorrect predictions is obtained. This means that all subsets are possible with greater than zero probability, and so determining whether $F[k] \neq 0$ is actually a question of whether a subset with sum k exists. Thus, the subset sum problem has a solution if and only if $F[k] \neq 0$ for the resulting SI problem. □

Subsequent sections of this article will consider appropriate methods for computing approximate solutions to the SI problem.

3.3 Applying the Sum Inference Problem to GROUP BY Queries

In this section we describe a simple algorithm (Function GetIntvls) that makes use of the SI problem to compute the information necessary to display the user interface described previously. As described, the user supplies parameters p and k such that with probability p, at least k of n groups are within the intervals or bounds that are computed; intervals are then chosen automatically to ensure the desired accuracy.

The algorithm makes the assumption that the selected intervals for every group should have the same accuracy. That is, we assume that the ratio of the width of the ith interval to the variance of the ith interval's standart deviation is always ρ. This is reasonable, since always using the same ratio gives all groups the same priority; without this restriction the number of acceptable confidence intervals is infinite. A binary search is then performed on the possible ρ values.

At each iteration, an SI problem is constructed that can be used to check if a given ρ produces an actual p that is too high or too low for the user's chosen k value.

$Function\ GetIntvls\ (k, p, n, \mu, \Sigma)$: $[l, h]$
1. Build a score vector s where $s_i = 1$ forall i
2. Choose an initial upper bound ρ_u for ρ
3. Set $\rho_l = 0$
4. Set $\rho = \frac{\rho_u + \rho_l}{2}$
5. Set $l_i = \mu_i - \rho \times \Sigma_{i,i}^{1/2}$
6. Set $h_i = \mu_i + \rho \times \Sigma_{i,i}^{1/2}$
7. Give μ, Σ, s, l, h as input to an SI problem.
8. If $|p - \sum_{j=0}^{n-k} F[j]| \leq \epsilon$ goto 11
9. If $p - \sum_{j=0}^{n-k} F[j] > \epsilon$ then $\rho_u = \rho$; goto 4
10. If $(\sum_{j=0}^{n-k} F[j]) - p > \epsilon$ then $\rho_l = \rho$; goto 4
11. Return l and h as the final confidence intervals.

There are a few considerations regarding the algorithm that warrant additional discussion. First, the procedure described above requires that a different instance of the SI problem be solved at each iteration of the algorithm. With a binary search algorithm of this type, 30 or so iterations of the algorithm may be required. This may be a concern, since the SI problem can be expensive to solve (see the next two sections). Fortunately, it turns out that in practice, almost all of the computations required to solve the SI problem can be reused across iterations, rendering the cost of subsequent SI solutions negligible compared to the cost of the first one. This is due to the fact that the underlying random variables $N_1, N_2, \ldots, N_n$ do not change across iterations; only the bounds l_i and h_i change.

Second, the algorithm as presented leaves open the question of how to choose an initial upper bound ρ_u. In our implementation we choose $\rho_u = 0.1$, and until the *goto* in line 9 is used, every time that the *goto* in line 10 is used we double ρ_u rather than setting $\rho_l = \rho$. We chose $\rho_u = 0.1$ because it is close to the error that a user might find acceptable in the final answer to the query; this tends to reduce the number of iterations of the binary search required. However, we point out that the initial choice of ρ_u is not critical, due to the exponentially-fast convergence of the binary search and the fact that additional SI solutions (past the first one) tend to have little cost.

4. SOLVING THE SI PROBLEM

Given the intractability of the SI problem, developing computationally feasible methods for solving it (or approximately solving it) in a database environment is mandatory if it is to be a practical abstraction. In this section we consider three different methods for solving the SI problem:

(1) a Monte Carlo solution that samples directly from F in order to learn the distribution;
(2) a second method that makes use of moment analysis and an appropriate, parametric model for F in order to approximate it; and

(3) a related method that instead performs approximate moment analysis and is more computationally efficient for very large instances of the SI problem.

Each of the solutions assumes that we have access to the mean μ and covariance matrix Σ that are provided as input into the SI problem. In the case of the GROUP BY query computation described in the previous section, computing μ requires only that we compute the Haas-Hellerstrin estimate for the answer to the GROUP BY query. However, the computation of Σ is substantially more involved, and is discussed in detail in Section 5.

4.1 A Solution Using Monte Carlo Resampling

The first solution we consider approximates F via Monte Carlo sampling. We repeatedly and directly sample from the multivariate normal or Gaussian distribution defined by μ and Σ. For each sample from the multivariate normal, we compute the total penalty obtained due to incorrect intervals. $F[k]$ is then estimated by computing the fraction of the time that the observed penalty is exactly k. For example, if 1000 samples are taken and a penalty of 13 was incurred 55 times, $\widetilde{F}[13] = 0.055$, where $\widetilde{F}$ is our estimate for F. The observed penalty for each trial or sample is computed by totaling the penalty over each of the n variables $N_1, \ldots, N_n$. For a given trial, if N_i happens to be outside of the range defined by l_i and h_i, then a penalty of s_i is included in the total for that trial.

While this method is fairly simple, there are a few technical questions that need to be considered. First, we need to be able to sample from the Gaussian distribution defined by μ and Σ. It turns out that there are well-known methods for this that involve first sampling a number of standard normal random variables, using those to form a random vector, and the rotating and translating the vector using μ and Σ and standard methods from linear algebra ([Dobra et al. 2002], Section 4).

A second key question that must be answered is how many Monte Carlo samples are required in order to obtain a sufficiently accurate approximation for F? In order to obtain a guideline for this number, we sample N times,[2] where N is chosen so that the expected Euclidean distance from $\widetilde{F}$ to F is less than ϵ for some user-supplied ϵ. In other words, we take enough samples so that

$$E\left[\sum_{v}(\widetilde{F}[v] - F[v])^2\right] < \epsilon^2 \tag{10}$$

Computing the number of samples required so that this inequality holds requires a bit of mathematics. We assume that $\sum_k s_k$ is max (that is, the total possible penalty for any given trial is max) and without losing generality we assume that there are no negative penalties (this simplifies the exposition by allowing summations over all possible penalty values to start at 0).

[2]In the remainder of this section N refers to the number of Monte Carlo trials. We choose not to subscript N in order to clarify this (using N_{MC}, for example) in order to simplify the presentation of the formulas. As in the previous section, N_i still refers to the ith component of the multivariate normal that makes up the SI problem.

Let $F[v]$ evaluated at $v = 0, 1, \ldots, max$ be $p_0, p_1, \ldots, p_{max}$. That is, the set of values $p_0, p_1, \ldots, p_{max}$ answers the SI problem correctly. We then define random variables $X_{1,0}, \ldots, X_{N,max}$, where $X_{u,v}$ indicates whether or not the total penalty incurred in the uth trial was exactly v. That is, $X_{u,v}$ is one if $v = \sum_k s_i I(N_i \notin [l_i, h_i])$ for the uth of N multivariate Gaussian samples, and $X_{u,v}$ is zero otherwise.

Note that $E[X_{u,v}] = p_v$ for $u = 1, \ldots, N$. That is, on expectation, X_{uv} is exactly the desired value p_v. Thus, we have

$$\begin{aligned}
E\left[\sum_{v=0}^{max}(\widetilde{F}[v]-F[v])^2\right] &= \sum_{v=0}^{max} E\left[\left(\frac{\sum_{u=1}^{N} X_{u,v}}{N} - p_v\right)^2\right] \\
&= \sum_{v=0}^{max} E\left[p_v^2 - 2p_v\frac{\sum_{u=1}^{N} X_{u,v}}{N} + \left(\frac{\sum_{u=1}^{N} X_{u,v}}{N}\right)^2\right] \\
&= \sum_{v=0}^{max}\left(p_v^2 - 2p_v^2 + \frac{1}{N^2}E\left[\sum_{u=1}^{N} X_{u,v}^2 + \sum_{u=1}^{N}\sum_{w=1_{w\neq u}}^{N} X_{u,v}X_{w,v}\right]\right) \\
&= \sum_{v=0}^{max}\left(-p_v^2 + \frac{1}{N^2}\left(Np_v + N(N-1)p_v^2\right)\right) \\
&= \sum_{v=0}^{max}\frac{p_v - p_v^2}{N} \\
&= \frac{1}{N}\left(1 - \sum_{v=0}^{max} p_v^2\right) \leq \epsilon^2
\end{aligned} \tag{11}$$

Therefore, if

$$N \geq \frac{1 - \sum_{v=0}^{max} p_v^2}{\epsilon^2} \tag{12}$$

then the expected Euclidean distance between F and $\widetilde{F}$ is less than ϵ. Note that this is guaranteed to be the case if $N \geq \frac{1}{\epsilon^2}$, because $1 - \sum_{v=0}^{max} p_v^2$ is bounded by 1. As a result, 10000 samples are enough to guarantee an expected Euclidean distance of no greater than 0.01. These methods will be considered experimentally later on in this article.

Reusing Monte Carlo SI Computations for `GROUP BY`. When the SI problem is applied to computing bounds for a `GROUP BY` query, the algorithm to solve the SI problem will be invoked repeatedly to perform a binary search. Fortunately, it is possible to reuse almost all of the computations from the first SI solution when computing subsequent solutions, because only the bounds l_i and h_i change across iterations, and all bound widths are controlled by the parameter ρ (see Section 3.3 for details). This means that subsequent SI solutions are almost "for free," so that the actual multivariate sampling only needs to be performed the very first time that the SI problem is solved.

To reuse computations, we use all of the N Monte Carlo trials to build a list of $(\rho^-, trialno, i)$ triples. Each triple means that for the specified trial, an

ρ value greater than ρ^- will cause N_i to fall outside of the resulting range l_i to h_i. These triples are constructed for every i, and then all of them are sorted based upon the ρ^- value during the first SI solution.

Subsequently, for any given ρ, it becomes very efficient to check the corresponding $\widetilde{F}[k]$. We scan this sorted array from front to back, and keep going as long as ρ^- does not exceed ρ. For every value of *trialno*, we count the number of triples found. Let *cnt* be the number of trials values where the number of triples found is k. Then returning cnt/N gives us $\widetilde{F}[k]$ for the given value of ρ. This process can be made even faster by precomputing summary statistics and storing them in the array so that the entire lower end of the array need not be rescanned for each ρ and k pair that is queried.

4.2 Solving the SI Problem Using Moment Analysis

Because it is distribution-free, Monte Carlo may be preferred, especially if the computational resources required are minimal. However, for an SI problem with a very large problem size—that is, with a very large number of underlying groups or variables—the Monte Carlo method may not be practical because it requires a large number of samples from the underlying Gaussian. Since the generative model may have arbitrary pairwise correlations, each Gaussian sample requires multiplying a length n vector by an n-by-n matrix, which will be expensive if n is very large.

Thus, our second method for solving the SI problem is quite different. It relies on calculating exactly the first and second central moments of the distribution function F and then choosing an appropriate parametric distribution to approximate F. Let ζ be a random variable whose distribution is precisely F. Then recall from Section 2 that the first and second central moments of F are the mean $E[\zeta]$ and variance $E^2[\zeta] - E[\zeta^2]$ of ζ.

The justification for resorting to such a parametric method is straightforward. Note that the valid domain of F (or range of ζ) is known and easily computed. That is, in linear time we can easily upper-bound and lower-bound the values of i for which $F[i]$ can possibly have nonzero probability by computing the smallest and largest sums over subsets of the s_i values input into the SI problem—the smallest possible sum is $min\{0, \sum_{s_i<0} s_i\}$, and the largest possible sum is $max\{0, \sum_{s_i>0} s_i\}$, where each s_i is the penalty value associated with the ith group. Given an upper bound, a lower bound, and a mean and variance, the possible shapes that a reasonably well-behaved distribution can take are severely constrained, and so choosing a parametric distribution that accepts these four parameters as a surrogate for F should not incur much inaccuracy.[3] As we show experimentally, the accuracy of this method is quite remarkable when it is used in conjunction with the Beta distribution [Casella and Berger 2002], which accepts exactly this set of four parameters. The Beta distribution can take a very wide variety of shapes (see Figure 3 at left). To give the reader an idea of how good the Beta distribution approximation is for our application, we show in Figure 3 (right) the empirical distribution of $F[k]$ in a

[3]As evidence of this, the well-known and widely-used Chevyshev's inequality states exactly how much just two parameters—mean and variance—constrain a distribution.

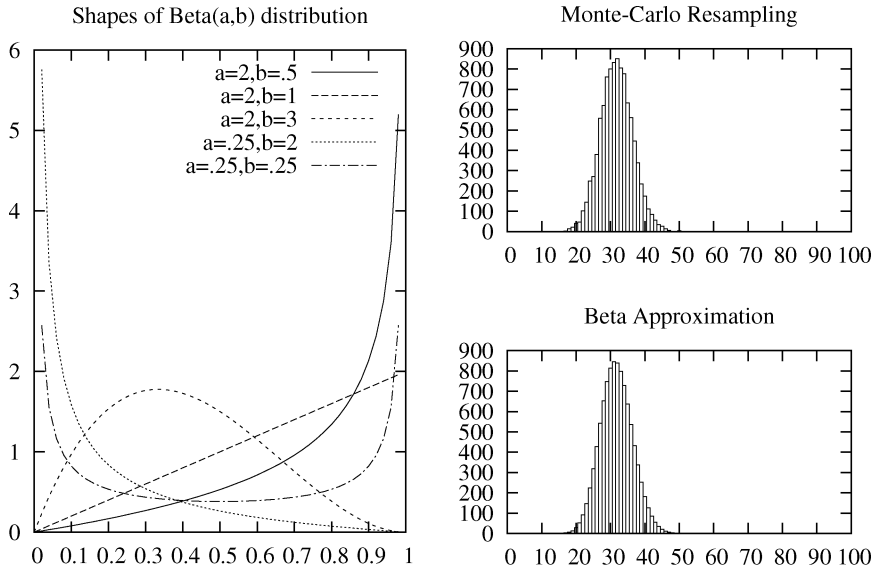

Fig. 3. Shapes of Beta distribution (left); the empirical estimate of pdf of $F[k]$ for 100 groups based on 10000 Monte Carlo samples (top right); and the approximation of the pdf of $F[k]$ using Beta(31.123,66.6907) (bottom right).

typical scenario obtained using 10000 Monte Carlo samples together with the approximation using Beta distribution that our method produces. The two distributions are virtually indistinguishable. The quality of the Beta distribution for use as a solution to the SI problem will be considered experimentally in depth in Section 6.

We note that while such parametric methods may be quite common in statistics and statistical machine-learning, there may be skepticism towards such methods in the part of data management practitioners. Guidelines on where to use such parametric distributions can be found in the statisics literature [Johnson et al. 1995]. If there *is* a concern regarding the accuracy, a principled approach to checking the accuracy of such an approximation would be to compute or estimate the third moment and (perhaps) the fourth moment using techniques similar to those that we will present shortly. These moments are the so-called skew and kurtosis of F, respectively. If both the skew and the kurtosis also match the parametric model chosen, then with six matching parameters the possible distributions are so constrained that no reasonable statistician would ever question the applicability of the model! However, as we will show experimentally, using the four-parameter method, it is possible to obtain excellent results.

Of course, all of this requires that we be able to perform an appropriate analysis on the variable ζ. Computing $E[\zeta]$ given the input to the SI problem is a relatively easy matter:

$$E[\zeta] = \sum_i \int_{\Re - [l_i, h_i]} p_i(x) s_i \, dx \tag{13}$$

where $p_i(x)$ is the probability density function of the estimate associated with the ith estimator, which is normal with mean $\mu[i]$ and variance $\Sigma[i,i]$. This integral for each i simply computes the probability that a trial over N_i results in a value that incurs a penalty, and multiplies that value by the incurred penalty in order to arrive at the expected penalty for the ith estimate.

We next turn our attention to calculating $E[\zeta^2]$. This is the expected *squared* penalty over all i:

$$\begin{aligned}
E[\zeta^2] &= \left(\sum_i I(N_i \notin [l_i, h_i])s_i\right)^2 \\
&= \left(\sum_i I(N_i \notin [l_i, h_i])s_i\right)\left(\sum_j I(N_j \notin [l_j, h_j])s_j\right) \\
&= \sum_i \left(I(N_i \notin [l_i, h_i])s_i\right)^2 + \sum_i \sum_{j \atop i \neq j} I(N_i \notin [l_i, h_i])s_i I(N_j \notin [l_j, h_j])s_j \\
&= \sum_i \left(\int_{\Re - [l_i, h_i]} p_i(x)s_i\, dx\right)^2 \\
&\quad + \sum_i \sum_{j \atop i \neq j} \int_{\Re - [l_i, h_i]} \int_{\Re - [l_j, h_j]} p_{i,j}(x, y)s_i s_j\, dx\, dy
\end{aligned} \tag{14}$$

In this equation, we have simply broken the expected squared penalty into two sums. The first sum considers the case where the penalty for a single N_i is squared, and the second sum considers the case where the penalty for two different N_i variables is multiplied. In the equation, $p_i(x)$ is again the probability density function of N_i, and $p_{i,j}(x, y)$ is the probability density function of the bivariate normal with

$$\mu' = [\mu[i], \mu[j]]^T; \Sigma' = \begin{bmatrix} \Sigma[i,i] & \Sigma[i,j] \\ \Sigma[i,j] & \Sigma[j,j] \end{bmatrix} \tag{15}$$

In other words, $p_{i,j}(x, y)$ is the joint density function for N_i and N_j. Evaluating this equation requires that we be able to compute the required integral over $p_{i,j}(x, y)$. Fortunately, there is a good deal of existing work that considers how to compute an integral over a bivariate normal. If Monte Carlo integration is used [Robert and Casella 2005], then it is easily possible to reuse the SI solution for the first iteration of the GROUP BY computation of Section 3.3 for subsequent iterations (see below).

Applying the Beta Distribution. After calculating the first two moments, we can then use an appropriate parametric distribution to (approximately) calculate any $F[k]$. As mentioned previously, in our implementation we make use of the four-parameter Beta distribution to serve as a surrogate for F. The four parameters for the Beta are the two shape parameters α and β, as well as the lower and upper bound *min* and *max* for the range of the underlying random variable ζ.

If $min = 0$ and $max = 1$, straightforward calculations show that given $\mu(\zeta)$ and $\sigma^2(\zeta)$, α and β should be chosen so that

$$\alpha = \frac{\mu^2(\zeta) - \mu^3(\zeta) - \mu(\zeta)\sigma^2(\zeta)}{\sigma^2(\zeta)}$$
$$\beta = \frac{\mu(\zeta) - 2\mu^2(\zeta) + \mu^3(\zeta) - \sigma^2(\zeta) + \mu(\zeta)\sigma^2(\zeta)}{\sigma^2(\zeta)}$$

Generally $min = 0$ and $max = 1$ are not satisfied. However a simple linear transformation can map this to a space such that $min = 0$ and $max = 1$. The mean and variance are transferred accordingly. Assuming the mean and variance in the original space are μ' and σ'^2, respectively, then the following equations hold:

$$\mu = \frac{\mu' - min}{max - min}$$
$$\sigma^2 = \frac{\sigma'^2}{(max - min)^2}$$

This ensures that the first two moments of the resulting distribution $\widetilde{F}$ both match $\mu(\zeta)$ and $\sigma^2(\zeta)$, respectively. Given this fit, to provide our solution to the SI problem we use $\widetilde{F}[k] = \int_{k-0.5}^{k+0.5} Beta(x)\,dx$.

Reusing Moment-Based SI Computations for `GROUP BY`. In the same way that the Monte Carlo solution for the first iteration of the `GROUP BY` algorithm can be reused, the moment-based solution can be reused to solve multiple SI problems with the same underlying variables. Assuming that Monte Carlo integration is used for each integral over $p_{i,j}(x, y)$ and $p_i(x)$, a number of samples are taken from each $p_{i,j}(x, y)$ and $p_i(x)$, and the fraction of samples within each corresponding bound is computed to approximate the integral. Using methods very similar to those in the previous section for reusing the pure Monte Carlo solution, these samples can be used to compute an efficient mapping from every possible ρ to a given variance and mean for ζ. For example, consider the mean of ζ. To handle this, a number of records of the form (ρ^-, i, μ) are computed for each i, one corresponding to each sample from p_i. This record indicates that N_i's contribution to the mean of ζ for $\rho > \rho^-$ is at least μ. The records are then sorted on the ρ^- values. To compute the mean of ζ for a given ρ, the records are scanned from front to back for each ρ^- that does not exceed ρ. For each i, the last μ observed is then used to compute ζ's mean.

4.3 Solution Using Approximate Moment Analysis

While the previous solution may be faster than the Monte Carlo solution, it is still linear in the size of the covariance matrix Σ. In the case of a `GROUP BY` query, the number of entries in Σ is quadratic in the number of groups. Since the number of groups may be many many thousands in a realistic scenario, a sublinear algorithm in the size of Σ is highly desirable.

For a very large SI problem, it is possible to increase the speed of the moment analysis method substantially by sampling from the covariance matrix. Recall that the first two moments of the distribution resulting from the SI problem are given as Eqs. (13) and (14). In order to make use of the method of moments, we are typically not interested in the second uncorrected moment as much as we are in the variance, which is related to the second moment via the relationship $\sigma^2(\zeta) = E[\zeta^2] - E^2[\zeta]$. To write an expression for $\sigma^2(\zeta)$, we define

$$C = \frac{\sum_i \left(\int_{\Re - [l_i, h_i]} p_i(x) s_i \, dx\right)^2 - \left(\sum_i \int_{\Re - [l_i, h_i]} p_i(x) s_i \, dx\right)^2}{\sum_i \sum_{j\, i \neq j} 1} \tag{16}$$

Then, we have

$$\sigma^2(\zeta) = E[\zeta^2] - E^2[\zeta] = \sum_i \sum_{j\, i \neq j} \left(\int_{\Re - [l_i, h_i]} \int_{\Re - [l_j, h_j]} p_{i,j}(x, y) s_i s_j \, dx \, dy + C \right) \tag{17}$$

The main reason that we define C is that in order to calculate C, we only need to calculate the diagonal entries of the covariance matrix, a cost that scales linearly with respect to the number of estimators. Thus we can compute C exactly, and the problem of calculating the variance reduces to the problem of computing the value of the double summation given above, which we can estimate effectively using sampling. Assuming the total number of estimators is n, the total number of nondiagonal entries in the double summation is $m = n(n-1)$. Let S be a random sample without replacement from

$$T = \{(i, j) | 1 \leq i \leq m, 1 \leq j \leq m, i \neq j\} \tag{18}$$

where $|S|$ is the sample size and $|T| = m$. The following formula then gives an unbiased estimator of the variance of ζ:

$$\widetilde{\sigma^2}(\zeta) = \frac{m}{|S|} \sum_{(i,j) \in S} \left(\int_{\Re - [l_i, h_i]} \int_{\Re - [l_j, h_j]} p_{i,j}(x, y) s_i s_j \, dx \, dy + C \right) \tag{19}$$

In order to compute how many samples from the covariance matrix are needed to accurately estimate $\sigma^2(\zeta)$, we can also derive a formula for the variance of $\widetilde{\sigma^2}(\zeta)$.

We define

$$\begin{aligned} a &= \frac{m}{|S|} \\ b &= \frac{m-1}{|S|-1} \\ h(i, j) &= \left(\int_{\Re - [l_i, h_i]} \int_{\Re - [l_j, h_j]} p_{i,j}(x, y) s_i s_j \, dx \, dy - C \right) \end{aligned} \tag{20}$$

If sampling without replacement is used, the variance of $\widetilde{\sigma^2}(\zeta)$ is

$$\sigma^2(\widetilde{\sigma^2}(\zeta)) = \left(\frac{a}{b} - 1\right) \sum_{(i,j)\in T} \sum_{(k,l)\in T} I((i,j) \neq (k,l))h(i,j)h(k,l) + (a-1) \sum_{(i,j)\in T} h(i,j)^2 \quad (21)$$

An unbiased estimator of the variance that is only based on the samples is

$$\widetilde{\sigma^2}(\widetilde{\sigma^2}(\zeta)) = ab\left(\frac{a}{b} - 1\right) \sum_{(i,j)\in S} \sum_{(k,l)\in S} I((i,j) \neq (k,l))h(i,j)h(k,l) + a(b-1) \sum_{(i,j)\in S} h(i,j)^2 \quad (22)$$

Given these formulas, we may easily design an algorithm that repeatedly samples from the nondiagonal entries in the covariance matrix and calculates the estimated value of $\widetilde{\sigma^2}(\zeta)$ based on the current sample set S. The algorithm stops sampling when the standard deviation of this estimator is less than some percentage of the estimated value, computed using an appropriate bound (such as the Central Limit theorem or Chebyshev's inequality).

The resulting algorithm is shown in Figure 4. Function *GetEstimatedVariance* implements the algorithm. The first two lines initialize some variables. Lines 3 to 8 calculate the constant C given in Eq. 16. The *Do*... *While* loop in lines 9 to 18 continues sampling and updating the estimated variance using Eq. 19 until the stopping condition described above is satisfied. The function *CalculateVariance* calculates the estimated variance using Eq. 22.

5. COMPUTATIONAL CONSIDERATIONS FOR COVARIANCE MATRIX ENTRIES

Regardless of which one of the three methods for solving the SI problem is used, entries in the covariance matrix must be computed. The Appendix gives a detailed derivation of the covariance formulas and associated unbiased estimators for sampling multitable joins over `GROUP BY` queries. Unfortunately, if one were to simply implement them directly using the obvious nested loops computations, it would be prohibitively expensive to compute even a single entry in the covariance matrix. This section will consider how these formulas can be implemented efficiently.

For ease of exposition, we will consider how to compute the covariance of the following two queries:

```
SELECT sum(f1(p1,l1))
FROM PART AS p1,LINEITEM l1
SELECT sum(f2(p2,l2))
FROM PART AS p2,LINEITEM l2
```

Function $GetEstimatedVariance\ (T, n)$
1. $Exp = 0, Vpart = 0, C = 0, m = n(n-1)$
2. $SampleSize = 0, Estimate = 0, Total = 0$
3. For $i = 1$ to M:
   4. $A = \left(\int_{\Re - [l_i, h_i]} p_i(x) s_i dx\right)$
   5. $Exp = Exp + A$
   6. $Vpart = Vpart + A^2$
7. $C = \frac{Vpart - Exp^2}{m}$
8. Do:
   9. Sample entry (i, j) from T
   10. $h(i,j) = \left(\int_{\Re - [l_i, h_i]} \int_{\Re - [l_j, h_j]} p_{i,j}(x,y) s_i s_j dxdy + C\right)$
   11. $SampleSet.add(h(i,j))$
   12. $Total = Total + h(i,j)$
   13. $SampleSize++$
   14. $Estimate = \frac{m}{SampleSize} Total$
   15. $\sigma^2 = CalculateVariance(SampleSet, N)$
   16. $\sigma = sqrt(\sigma^2)$
17. While$(Std \times 10 > Estimate)$
18. Return $Estimate$

Function $CalculateVariance\ (S, m)$
1. $NonEqPart = 0, EqPart = 0$
2. $a = \frac{m}{|S|}, b = \frac{m-1}{S-1}$
3. For any two different elements $s, t \in S$
   4. $NonEqPart = NonEqPart + s \times t$
5. For any element $s \in S$
   6. $EqPart = EqPart + s^2$
7. Return $ab(\frac{a}{b} - 1)NonEqPart + a(b-1)EqPart$

Fig. 4. Algorithm to calculate the estimated variance.

In these queries, f_1 computes the total aggregate value for one group, while f_2 computes the total aggregate value for a second group. f_1 and f_2 also compute any selection and/or join conditions. Subsequently, we assume that both functions correspond to queries that can be evaluated using a hash-join or a sort-merge-join.

Upon examining the formulas in Appendix A, the key statistics needed to estimate the covariance between the two estimators given above are the following:

$$V_{TT} = \sum_{p_1 \in P} \sum_{l_1 \in L} f_1(p_1, l_1) f_2(p_1, l_1)$$

$$V_{TF} = \sum_{p_1 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L, l_2 \neq l_1} f_1(p_1, l_1) f_2(p_1, l_2)$$

$$V_{FT} = \sum_{p_1 \in P} \sum_{p_2 \in P, p_2 \neq p_1} \sum_{l_1 \in L} f_1(p_1, l_1) f_2(p_2, l_1)$$

$$V_{FF} = \sum_{p_1 \in P} \sum_{p_2 \in P, p_2 \neq p_1} \sum_{l_1 \in L} \sum_{l_2 \in L, l_2 \neq l_1} f_1(p_1, l_1) f_2(p_2, l_2)$$

$rid(p_1)$	$rid(l_1)$	$f_1(p_1, l_1)$
1	1	2
1	3	5
3	2	4
2	1	3

Fig. 5. Example results for query 1.

$rid(p_2)$	$rid(l_2)$	$f_2(p_2, l_2)$
3	1	3
1	2	2
2	3	3
3	2	1

Fig. 6. Example results for query 2.

In the previous summations, P and L refer to our samples of PART and LINEITEM, respectively. Equivalent formulas for these statistics can be obtained by making use of the identity function I:

$$V_{TT} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 = p_2) I(l_1 = l_2) f_1(p_1, l_1) f_2(p_2, l_2)$$

$$V_{TF} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 = p_2) I(l_1 \neq l_2) f_1(p_1, l_1) f_2(p_2, l_2)$$

$$V_{FT} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 \neq p_2) I(l_1 = l_2) f_1(p_1, l_1) f_2(p_2, l_2)$$

$$V_{FF} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 \neq p_2) I(l_1 \neq l_2) f_1(p_1, l_1) f_2(p_2, l_2)$$

Note that these two relations have four statistics, where the subscript *TT* (for example) requires a double summation over the result sets for f_1 and f_2, considering identical source record pairs. The subscript *TF* requires a summation over the two result sets for nonzero f_1 and f_2 values where the source records from P are the same, but those from L differ. *FT* is similar. The subscript *FF* requires a summation over the two result sets where the source records from both P and L differ.

In order to calculate each of these statistics, a reasonable method would be to make use of a double summation over the result set from both of the queries. First, we would use a hash join to compute the set of all nonzero f_1 values, since those are the only ones that may contribute to the actual value for the statistic. Then, in a similar fashion, we would use a hash join to compute the set of all nonzero f_2 values. In order to determine whether each nonzero value can contribute to one of the statistics, for each result tuple we also need to store the record IDs (RIDs) of the two tuples that were used to produce the value. For example, for V_{FF}, only those pairs of record pairs where $I(p_1 \neq p_2)$ and $I(l_1 \neq l_2)$ both evaluate to 1 can contribute to the total; this can be determined if we have the necessary RIDs. Once this data has been computed and stored in an array, then a nested summation can be used to compute each statistic. For example, consider Figures 5 and 6, which give example nonzero f_1 and f_2

values along with the source record IDs for the two queries. Given such a data structure, the following algorithm then computes V_{FF}:

Function CalculateV_{FF} (S_1,S_2)
1. $V_{FF} = 0$
2. For every record in S_1
    3. For every record in S_2
        4. $V_{FF} = V_{FF} + I(p_1 \neq p_2)I(l_1 \neq l_2)f_1(p_1, l_1)f_2(p_2, l_2)$
5. Return V_{FF}

If S_1 is the set of nonzero f_1 values and S_2 is the analogous set for f_2, this algorithm would require $O(n|S_1||S_2|)$ time. For a larger number of relations, the complexity increases quickly. In general, over n relations, the total number of statistics required is 2^n, and the time complexity for relations S_1 to S_n is $O(2^n \prod |S_i|)$.

By careful implementation, we can reduce the time complexity in the two-relation case to to $O(|S_1|+|S_2|)$ (and, by extension, we can reduce the complexity over n relations to $O(2^n \sum |S_i|)$). Continuing with the same example, the basic idea is as follows:

(1) First, it is easy to calculate V_{TT}. We simply to a hash join on both the RID's of PART and LINEITEM.

(2) Second, once we have calculated V_{TT}, it is easy to calculate V_{TF}. Let

$$V_{T*} = \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 = p_2) f_1(p_1, l_1) f_2(p_2, l_2) \tag{23}$$

Instead of checking the equality of both the pairs of RIDs from both relations, V_{T*} only checks the equality of the two RIDs from PART. This can be computed efficiently using a hash join on the RID from PART. Then, we notice that

$$\begin{aligned} V_{T*} &= \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 = p_2) f_1(p_1, l_1) f_2(p_2, l_2) \\ &= \sum_{p_1 \in P} \sum_{p_2 \in P} \sum_{l_1 \in L} \sum_{l_2 \in L} I(p_1 = p_2)(I(l_1 = l_2) + I(l_1 \neq l_2)) f_1(p_1, l_1) f_2(p_2, l_2) \\ &= V_{TF} + V_{TT} \end{aligned} \tag{24}$$

As a result, we can compute V_{TF} as $V_{TF} = V_{T*} - V_{TT}$.

(3) Third, we can calculate V_{FT} in a similar way. We first calculate V_{*T} using a hash join on the RID from LINEITEM; then, $V_{FT} = V_{*T} - V_{TT}$.

(4) Finally, it is easy to calculate $V_{**} = V_{TT}+V_{TF}+V_{FT}+V_{FF}$. This is equivalent to calculating

$$\sum_{p_1 \in P} \sum_{l_1 \in L} f_1(p_1, l_1) \sum_{p_2 \in P} \sum_{l_2 \in L} f_2(p_2, l_2) \tag{25}$$

This calculation only requires a single scan of both S_1 and S_2. Then,

$$V_{FF} = V_{**} - V_{TT} - V_{TF} - V_{FT} \tag{26}$$

The time complexity is $O(|S_1| + |S_1|)$ for each step. Therefore, the overall time complexity is $O(|S_1| + |S_1|)$.

We illustrate the process with an example. S_1 and S_2 are shown in Figures 5 and 6, respectively. By executing each step, we have

$$\begin{aligned} V_{TT} &= 4 \\ V_{T*} &= 39 \\ V_{*T} &= 42 \\ V_{**} &= 126 \end{aligned}$$

Therefore,

$$\begin{aligned} V_{TF} &= 39 - 4 = 35 \\ V_{FT} &= 42 - 4 = 38 \\ V_{FF} &= 126 - 4 - 35 - 38 = 49 \end{aligned}$$

It is not hard to extend this idea to any number of relations. For example, if the two queries both have three relations, the first step is to calculate V_{TTT} using a hash join on the RIDs from all three relations. The next step is to calculate V_{TT*}, V_{T*T}, and V_{*TT}. We can then get V_{TTF}, V_{TFT}, and V_{FTT}. Similarly, we calculate V_{T**}, V_{**T}, and V_{*T*} which allows us to obtain V_{TFF}, V_{FFT}, and V_{FTF}. Finally, we calculate V_{***} and use this to compute V_{FFF}. The only difficulty is that we need to apply the inclusion/exclusion rule carefully to ensure that no values are double-counted. For example, if we want to calculate V_{TFF}, we notice that $V_{T**} = V_{TTT} + V_{TTF} + V_{TFT} + V_{TFF}$. In order to compute V_{TFF}, we need to subtract the other three totals from V_{T**}.

6. EXPERIMENTAL EVALUATION

In this section we experimentally evaluate the utility of the SI-based simultaneous `GROUP BY` bounds. There are two questions we consider. First, the purpose of sampling from a database is to obtain an approximate answer quickly. However, the additional computation required to produce the simultaneous `GROUP BY` bounds will require extra time, which may perhaps mitigate the benefit of sampling. Thus, the first question we wish to address is: For each of the three methods that we have proposed, how high is the time required to obtain the SI-based simultaneous `GROUP BY` bounds compared to the time required to execute the query over the entire database?

Second, all three methods that we have proposed are approximate methods, and do not solve the SI problem exactly. Therefore, the second question we wish to address is: How well do these three methods bound the correctness of an approximate `GROUP BY` answer in reality?

6.1 Running Time Experiments

To evaluate the utility of the SI-based simultaneous `GROUP BY` bounds, we are interested in three different times:

(1) the time to complete the query over the entire database;
(2) the time to complete the query over a sample set in order to produce an approximate answer; and
(3) the time to compute simultaneous confidence bounds using the SI problem.

If sampling is useful for reducing the running time of a query, we would expect that the summation of the last two times to be much smaller than the first time. Otherwise, we should simply answer the query exactly. In the ideal case, we would also expect the third time to be much smaller than the second one, so that the simultaneous confidence bounds are produced at no extra cost compared to the time required to estimate the answer. However, even if the third time is larger than the second one, this only indicates that there is room for improvement and does not preclude the use of our methods, because the total sampling time may still be less than evaluating the query exactly.

6.1.1 *Experimental Setup.*

(a) *Query and Schema Tested.* In our experiments, the following SQL statement is used:

```
SELECT SUM(o.Profit)
FROM Order o, Employee e, Branch b
WHERE o.EmpID = e.EmpID AND e.OfficeID = b.OfficeID
GROUP BY b.BranchID
```

The three relations in the query are

```
Order(EmpID, Profit, Other)
Employee(EmpID, OfficeID, Other)
Branch(BranchID, OfficeID, Other)
```

The `Other` field in each table refers to other possible attributes that could appear. In our experiments, this field is occupied by a random string so that the total size of each record is 100 bytes. We set up the query so that the join between `Order` and `Employee` is a primary-key to foreign-key join, and the join between `Employee` and `Branch` is a many-to-many join.

(b) *Parameters.* In this set of experiments, we want to test how different parameters may affect the three times described above. The parameters we consider are the database size, the skewness of the attributes in the join operation, the number of groups in the query, and the sampling ratio.

The *database size* determines the size of each relation. For simplicity, we will treat the size of the largest relation `Orders` as being approximately the same as the database size. The sizes of relations `Employee` and `Branches` are 0.1 and 0.01 times as large as the size of relation `Orders`. We test 10 GB, 1 GB and 0.1 GB databases. The default database size is 10 GB. Given the fact that the size of each record is 100 bytes, the default number of records in `Orders`, `Employee`, and `Branches` in a 10 GB database is 100 million, 10 million, and 1 million, respectively.

The *skewness* of the join attributes affects the join in two ways: first, a join operation takes more time if the join attributes are skewed; and second, the correlations between different groups increase as the skewness increases. We use a zipf coefficient to control the skewness of the three attributes `Employee.OfficeID`, `Branch.OfficeID`, and `Order.EmpID`. The default zipf coefficient is 0.

The *number of groups* defines the number of distinct values for the attribute Branch.BranchID, which appears in the GROUP BY clause of the query. This parameter does not affect the first two times much because the grouping operation is near the end of query execution. However, increasing the number of groups increases the number of concurrent estimators, and therefore increases the size of the covariance matrix that is input into the SI problem. Thus the time to solve the SI problem is affected. The default number of groups is 100.

The *sample ratio* determines the number of samples obtained from the database. Given the fact that all other parameters are fixed, increasing the sample ratio increases the sample size. Thus, both the second and third times described above should be affected. We sample so that if we obtain a $p\%$ sample from Orders, we will obtain a $5p\%$ sample from Employee, and a $20p\%$ sample from Branches. By default, we obtain a 1% sample from Orders. The sampling without replacement policy is used in each experiment.

(c) *Data Generation.* Given a set of parameter values, we generate the data set relation by relation. First, records in Order are generated as follows:

(1) A Profit is randomly generated uniformly between 1 and 100.
(2) An EmpID is produced by a zipf distribution with a domain size 0.1 times as large as the number of records in Order, and a zipf coefficient specified by the skewness of the join attributes. Because the total number of records in Employee is 0.1 times as large as the total number of records in Order, we generate EmpID from this domain to guarantee that the primary-key to foreign-key join between Order and Employee over EmpID is preserved.
(3) A random string is generated to make the size of the record 100 bytes.

Second, records of relation Employee are generated. For each record, the following steps are performed:

(1) A unique EmpID is assigned (from 1 to 0.1 times as large as the number of records in Order). EmpID is then the primary key of Employee, and guarantees the domain of EmpID in Employee and Order are the same to preserve the correctness of the primary-key to foreign-key join.
(2) An OfficeID is produced by a zipf distribution with a domain size 0.01 times as large as the number of records in Order, and a zipf coefficient specified by the skewness of the join attributes. We generate OfficeID from this domain for both Employee and Branch to preserve the correctness of the many-to-many join between these two relations over OfficeID.
(3) A random string is generated to make the size of the record 100 bytes.

Third, records of relation Branch are generated. For each record, the following steps are performed:

(1) An OfficeID is produced by a zipf distribution with a domain size 0.01 times as large as the number of records in Order, and a zipf coefficient specified by the skewness of the join attributes. This preserves a many-to-many join between Employee and Branch over OfficeID. Furthermore, this domain is as large as the total number of records in Order.

(2) A BranchID is randomly selected (uniformly distributed) from value 1 to the total number of groups.
(3) A random string is generated to make the size of the record 100 bytes.

(d) *Hardware and Software.* All the experiments are run over a DELL desktop with an Intel core-dual 1.8G CPU and 2GB memory using Open SUSE Linux 10.1.[4] Postgres 8.1[5] is used as the backend DBMS to store all the data and perform queries (both queries over the whole database and queries over the samples). The codes are written in C++ and compiled using gcc 4.1.0. Libpqxx 2.6.7[6] is used to connect PostgreSQL in C++. GSL1.9[7] is used for scientific computation and random number generation.

(e) *Experimental Procedure.* We prepare four different sets of databases and samples for the experiments. Each time, we fix three parameters using the default values and vary the fourth. In the first set of experiments we vary database size to generate 100MB, 1GB, and 10GB databases. In the second set of experiments we generate different data sets using zipf coefficients 0, 0.3, and 0.6. In the third set of experiments we generate data sets with 100, 1000, and 10000 groups. In the last set of experiments we generate a database and obtain three different samples using sample ratios 0.5%, 1%, 2% for Order. The two other relations follow the rule described above. For each database and sample generated, we then run the experiments five times using each of the three proposed solutions to the SI problem and compute the average of the five runs.

In each experiment, the time required to solve the SI problem includes exactly the time required to compute F once the join over the sample has been computed and is sitting in main memory.

The samples used in each experiment were precomputed and stored within the database. Since the samples were relatively small with respect to the available main memory, it is reasonable to assume that they were buffered entirely within main memory by the database system and did not need to be read from disk.

6.1.2 *Results.* The results are shown in Figure 7. Running times are given in seconds. Numbers are rounded to the nearest second. When the number of groups is 10000, both the moment analysis method and the Monte Carlo resampling method do not finish in a reasonable time, resulting in an N/A in the figure. This is because these two methods require operations over a 10000-by-10000 covariance matrix, which is not possible using our hardware.

In order to put the sample sizes that we tested into some sort of context, the 95% confidence interval width for sample ratio 0.5%, 1%, and 2% averaged 43%, 20%, and 8% of the estimated answer, respectively, if all other parameters take their default value. Thus, a 2% sample produces errors that are less than 10% (which should be quite reasonable for many data exploration applications)

[4]http://www.opensuse.org.
[5]http://www.postgresql.org.
[6]http://pqxx.org.
[7]http://www.gnu.org/software/gsl.

Running time under database size

	100MDB	1GDB	10GDB
Query time	8	85	1053
Estimate time	1	2	12
Moment analysis	11	12	23
Approximate moment analysis	1	1	1
Monte Carlo resampling	66	111	221

Running time under different skewness

	skew0	skew0.3	skew0.6
Query time	1053	1604	29164
Estimate time	12	16	27
Moment analysis	23	26	406
Approximate moment analysis	1	1	5
Monte Carlo resampling	221	245	435

Running time under different number of groups

	100 groups	1K groups	10K groups
Query time	1053	1057	1086
Estimate time	12	15	16
Moment analysis	23	215	N/A
Approximate moment analysis	1	1	57
Monte Carlo resampling	221	18055	N/A

Running time under different sample ratio

	0.5%	1%	2%
Query time	1053	1053	1053
Estimate time	8	12	32
Moment analysis	15	23	28
Approximate moment analysis	1	1	3
Monte Carlo resampling	219	221	230

Fig. 7. Running time under different parameter values.

and the error bounds seem to shrink by one-half or more with a doubling of the sample size.

6.1.3 *Discussion.* Several interesting results can be observed, we point out a few significant ones here.

First, we notice that the time to actually compute the sample-based estimate increases approximately linearly as the sample ratio increases (which is expected), while the time required to solve the SI problem is generally independent of the sample size.

Second, the running time for Postgres to complete the query over the entire database is from around 16 minutes to more than 8 hours. On the other hand, Postgres never needed longer than a minute to compute an approximate query answer over the samples in the worst case. Thus, sampling itself seems to be a valuable method in terms of reducing computation time, even over multitable joins.

Third, as long as the number of groups is 100, even for the slowest method (Monte Carlo resampling), we notice that the running time to produce simultaneous `GROUP BY` confidence intervals in the worst case is only 20% as long as the running time to execute the query over the entire database when the data has

no skew. If the data is very skewed, this extra time is only 2% as long as the query time. This indicates that all three methods are valuable for 100 groups or less. However, when the number of groups is 1000, Monte Carlo resampling takes around 20 times as long as the running time to execute the query over the entire database, and thus is not practical to use. Both moment analysis and approximate moment analysis still require less time than the time to execute the query over the entire database. When the number of groups is 10000, the only practical method is approximate moment analysis.

We notice that the third time for both the moment analysis and the Monte Carlo resampling method is generally larger than the time to complete the query over the samples. This indicates that these two methods are not ideal, though they are still useful. However, the approximate moment analysis method was nearly ideal in terms of its computational time. Most of the time, the time to obtain simultaneous GROUP BY bounds was much smaller than the time to compute the query result over a sample. The only time where this was not the case was when there were 10000 groups (the computation required 57 seconds). However, even in this case, the method takes just 4 times as long as it does to compute the query result over the sample, and it takes just 1/20 as long to obtain the simultaneous bounds as it does to compute the exact result. Therefore approximate moment analysis can be used virtually for free in most cases in terms of the extra computation that is required.

Finally, we acknowledge that our experiments consider what is really the best possible scenario for the application of sampling: an arbitrary, ad hoc query is issued for which no precomputed computation is available. This favorable scenario is the main reason for the orders-of-magnitude speedup associated with sampling. In other scenarios, such as an incremental scenario where a user drills down into an already-computed answer set, sampling may look much less attractive. Still, the results do convincingly show that when sampling is applicable, computing SI-based simultaneous bounds should not be much more expensive than simply using the underlying sample to compute an estimate.

6.2 Correctness Experiments

6.2.1 *Experimental Setup.* In this section we experimentally evaluate the correctness of our three methods for solving the SI problem, including the applicability of the parametric solutions.

Testing the correctness of any confidence bound is nontrivial, and is generally done using Monte Carlo methods. For example, a traditional $p\%$ confidence bound tells us that a parameter of interest is within a specified range $p\%$ of the time. A Monte Carlo experiment to check the correctness of such a confidence bound would rerun the computation that produced the bound N independent times, and see if approximately $\frac{p}{100} \times N$ out of the N times the bounds contain the true answer. Under this regime, we can treat each repetition as a coin flip where we observe a heads when the bound contains the true answer, and a tails otherwise. If the confidence bound computation is correct, the total number of observed heads would follow a binomial distribution given by $Binomial(N, \frac{p}{100})$. As long as the actual number of observed heads is within a two-sided binomial

confidence interval computed using $Binomial(N, \frac{p}{100})$, we can be reasonably sure that this confidence bound is correct.

It is more challenging to design a test for our three solutions to the SI problem. In the case of a GROUP BY query, for a given set of confidence bounds, the solution to the SI problem is a function F such that for any given k, $F[k]$ is the probability of observing k incorrect intervals. For example, if $k = 12$ and $F[k] = 0.25$, we expect 25% of the time to observe 12 incorrect bounds. Thus, if $p \approx \sum_{i=0}^{k} F[i]$, then the SI solution asserts that the probability of observing k or less incorrect bounds is p.

For a given p, a Monte Carlo experiment can be designed to check the correctness of an SI solution by repeating the following procedure N times independently. First, we obtain a sample from the database tables and use that sample to estimate the result to a GROUP BY query. For each group, we construct a 95% confidence interval using the sample and solve the resulting SI problem. Using F, we find k such that the probability of observing k or less incorrect bounds is p. We then calculate the actual number of incorrect intervals by computing the actual answer of the query over the entire database and then counting the total number of intervals that do not contain the answers for the associated group. The "coin flip" associated with this SI solution is a heads if the actual number of intervals is less than or equal to k, and a tails otherwise. Therefore, if the method to solve the SI problem is correct, the total number of heads in N independent repetitions follows a binomial distribution given by $Binomial(N, p)$. As long as the actual number of observed heads is within a two-sided binomial confidence interval, we have strong evidence that the method for solving the SI problem is correct.

To actually implement such an experiment, we use the same query described in the previous section:

```
SELECT SUM(o.Profit)
FROM Order o, Employee e, Branch b
WHERE o.EmpID = e.EmpID AND e.OfficeID = b.OfficeID
GROUP BY b.BranchID
```

We build four different data sets for this experiment. In the first data set, all parameters take their default values. In the second data set, the sample ratio is changed to 0.5%. In the third data set, the skewness is changed to 0.3. In the fourth data set, the skewness is changed to 0.3, and the number of groups is changed to 20. For each of our three methods (Monte Carlo resampling, moment analysis, and approximate moment analysis), and for each of four data sets, we perform the procedure described above for p in $\{0.05, 0.10, \ldots, 0.90, 0.95\}$, and $N = 100$.

6.2.2 *Results.* The results we observed are given in Figures 8, 9, 10, and 11. The figures show the 12 results observed using the three different SI solutions and four different data sets. The x axis in each figure is the p tested. The y axis (shown as a bar) is the total number of heads we observed out of 100 trials. The error bar over each bar is the two-sided 95% confidence interval of

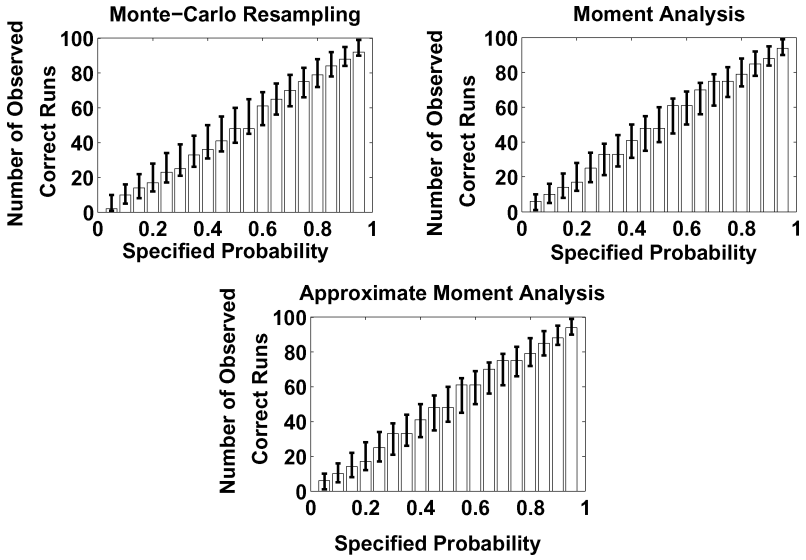

Fig. 8. Correctness of three methods over dataset one: all parameters use their default values. The error bar is a 95% two-sided confidence interval from the corresponding binomial distribution $Binomial(100, p)$, where each p value is shown in the x axis.

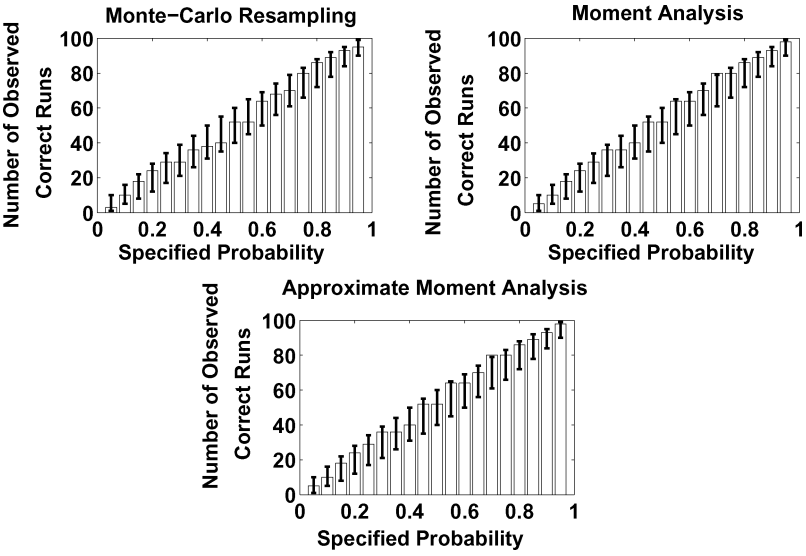

Fig. 9. Correctness of the three methods over dataset two: except for the fact that the sampling ratio is set to 0.5%, the three other parameters use their default values.

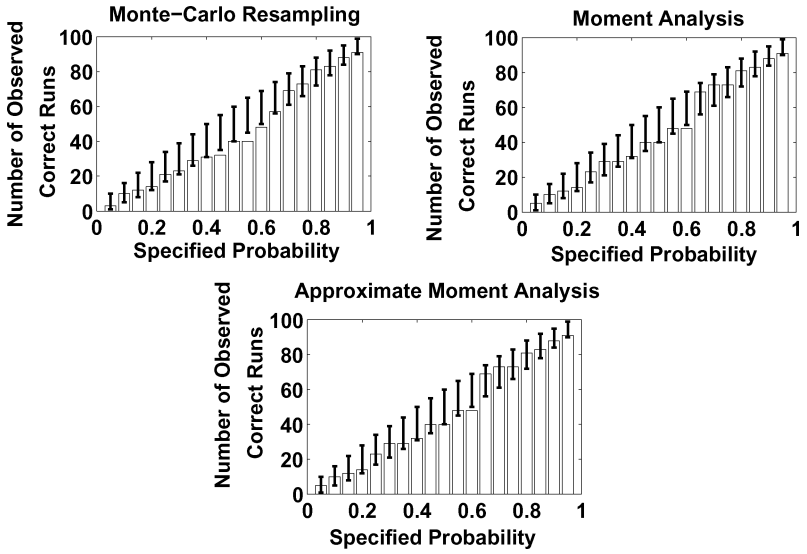

Fig. 10. Correctness of three methods over dataset three: except for the fact that the skew is set to 0.3, the three other parameters use their default values.

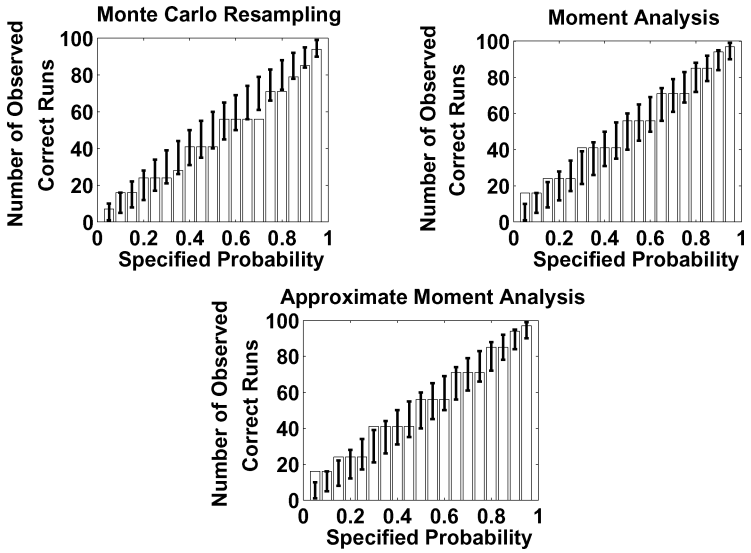

Fig. 11. Correctness of the three methods over data set four: 20 groups and a skew of 0.3 are used.

Binomial(100, p). In order to compute the interval, we compute the inverse cumulative distribution function values of probabilities 0.025 and 0.975 from *Binomial*(100, p) and treat them as lower bound and upper bound, respectively.

6.2.3 *Discussion.* If the SI solutions were unreliable in practice, we would expect to observe a number of heads outside the two-sided 95% confidence

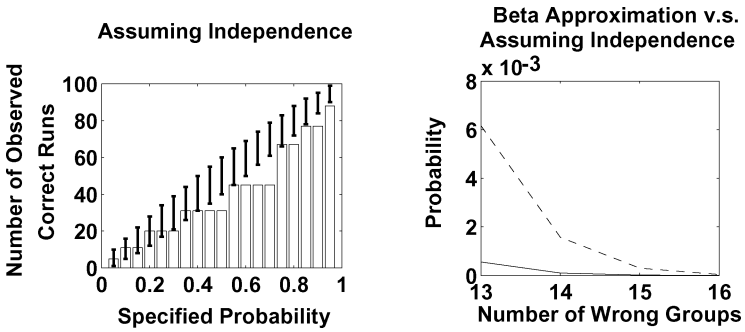

Fig. 12. Correctness obtained assuming independence on data set four. The right plot shows the difference between the SI solution obtained using independence (the solid line) and the Beta approximation (the dashed line).

interval of $Binomial(100, p)$ more than 5% of the time. In actuality, we observed that only 7 out of 240 results were outside of the two-sided 95% confidence interval of $Binomial(100, p)$. Therefore, we have strong evidence that the three methods for solving the SI problem do in fact produce correct and reliable results.

Furthermore, this low rate of error was observed for all three of the methods, including the two parametric solutions that rely on the Beta distribution—in fact, the rate of errors for the parametric, Beta-based solution was actually lower than for the nonparametric Monte Carlo method. This seems to be strong evidence for the assertion that using the four parameters in the parametric solution (lower bound, upper bound, mean, and variance) does constrain the space of possible distributions so much that a parametric assumption is quite reasonable, and seems to discount the need for a parametric solution that takes into account even higher moments such as the skew and kurtosis.

Finally, we note that while the results obtained in this set of experiments do seem to argue convincingly that the SI-based solution to the GROUP BY problem is accurate in practice, they do not necessarily mean that the SI-based solution is actually *necessary* in practice. It is still reasonable to ask "How dangerous would it be to simply ignore the correlations among groups, and assume that all groups are independent?"

Our first step toward answering this question is to repeat the procedure described above, while ignoring the correlations among the groups. Recall that in the Monte Carlo experiment, we first obtained a sample from the database. Then for each group, we constructed a 95% confidence interval using the sample. If we assume that the intervals are independent, then the probability of seeing that exactly k bounds are incorrect follows a Binomial distribution $Binomial(m, 0.05)$, where m is the total number of groups and 0.05 is the probability of seeing a single interval wrong. Thus, if we assume independence, the binomial distribution can be used to provide a very simple "solution" to the SI problem by using $F[k] = \sum_{i=0}^{k} Prob(Binomial(m, 0.05) = i)$. Figure 12 shows the accuracy obtained with this binomial "solution" to the SI problem, using exactly the same experimental setup for the other SI problem solutions with p

in $\{0.05, 0.10, \ldots, 0.90, 0.95\}$, and $N = 100$ over the fourth data set (the skewness is 0.3, the number of groups is 20). We see that for large p values, the observed value tends to be below the 95% error bar. For example, for $p = 0.6$, the 95% error bar covers the range from 50 to 70, but the observed value is only around 45.

Unfortunately, these results do not clearly demonstrate the extent to which a user can be mislead when the independence assumption is used to communicate the accuracy of a `Group by` querying the user. In Figure 12, if the independence assumption were valid, the total number of times that we would observe that the actual count of incorrect intervals is smaller than or equal to a given k (where $F[k] = p$) should be approximately $p \times 100$. This figure shows that for larger p values, the number of times that the count of incorrect intervals does not exceed k tends to be smaller than $p \times 100$. Thus, for a fixed k, when using the independence assumption we will obtain a estimated probability of seeing k or more incorrect intervals that are too small. The problem is that Figure 12 does not show exactly the extent to which this probability can be underestimated.

In order to investigate exactly how this probability is underestimated, we arbitrarily choose one of the 100 Monte Carlo trials from Figure 12, and plot the probability of seeing k or more incorrect groups obtained by using our Beta approximation, as well as the same probability obtained using the independence assumption. The results are shown on the right side of Figure 12. We only plot the tail portion of the cumulative density function because observing a large number of incorrect groups is the most worrisome outcome for the user of a sampling-based estimate. The figure shows that the probability of seeing 14 or more incorrect groups out of 20 is around 0.002 using our Beta approximation, which means that this can actually happen around once per 500 query executions. This is not insignificant, and not communicating this may be dangerous. However, when using the independence assumption, the computed probability is very close to 0. If a user takes this result at face value, he or she will assume that there is no chance of observing that the observed query result is largely useless.

Finally we point out that we generated the data using mild zipf skew in order to induce some correlations among the groups. In reality, correlations among groups can be much more significant than those induced by the zipf distribution. Thus the difference between the two probabilities can be even more significant in practice than what we show here, making the assumption of independence even more problematic in practice.

7. FUTURE WORK AND APPLICABILITY TO OTHER ESTIMATION PROBLEMS

In this article we have concentrated on sampling-based estimates for `SUM/GROUP BY` queries. However, many of the methods considered in this article are quite general and applicable to a much wider class of estimation problems than what we have considered here. To maintain focus and brevity, we have chosen not to consider these other applications in depth, and leave them to future work. However, we do give some hints in this section, as to how the SI problem can be applied beyond `SUM/GROUP BY` queries.

For example, it is not too difficult to extend our methods to deal with AVERAGEes over GROUP BY queries. An AVERAGE query is nothing more than a ratio of a SUM query to a COUNT query (which is itself a SUM query over a set of values that are always one). In this case, the definition of the SI problem must be changed slightly. Rather than having a single, normally-distributed random variable N_i associated with each estimate, there are two random variables N_i^{num} and N_i^{den} associated with the ith estimate—the first is associated with the SUM in the numerator of the AVERAGE query and the second with the COUNT in the denominator. The goal then becomes to infer the distribution of the number of times that the random variable $\frac{N_i^{num}}{N_i^{den}}$ is outside of the range l_i to h_i, given the covariances among all of them. Assuming sampling-based estimation, the Central Limit theorem still applies to each individual N_i^{num} and N_i^{den}. Since each individual variable is still normal, the methods considered in Section 4 are still applicable to solving the resulting, modified SI problem, with some slight modifications. For example, the Monte Carlo solution can easily be used by repeatedly sampling $N_1^{num}, N_1^{den}, N_2^{num}, N_2^{den}, \ldots, N_n^{num}, N_n^{den}$ and counting the number of times that $\frac{N_i^{num}}{N_i^{den}}$ is outside of the specified range. The two moment-based methods are also easily applied, though now the central task is computing the probability that both of the $\frac{N_i^{num}}{N_i^{den}}$, $\frac{N_j^{num}}{N_j^{den}}$ pairs are outside of their respective bounds for an arbitrary i and j.

It also seems possible to use the SI problem to produce a sampling-based estimate to so-called correlated aggregate queries. The following is an example correlated aggregate query:

```
SELECT SUM(EMP.SALARY)
FROM EMP
WHERE EMP.SALARY > (
   SELECT SUM(SALE.PRICE - PROD.COST)
   FROM SALE, PROD
   WHERE SALE.PROD=PROD.PROD AND
     AND SALE.EID=EMP.EID)
```

Such queries are characterized by a predicate over an inner aggregate query that is correlated with an outer aggregate query. This particular query finds the total salary "wasted" on employees who don't produce enough revenue to pay for themselves. One way to speed evaluation of this sort of query is to sample the SALE and PROD tables so that the predicate over each employee is evaluated approximately. Assuming that the SALE table is the largest database table and the join between SALE and PROD is expensive, this can speed query evaluation significantly. However, it then becomes unclear whether each employee should be included in the final total, and the distribution of the "penalty" or error incurred by including one or more employees when they should not be included (or vice-versa) can be calculated as an instance of the SI problem.

For another example of where the SI problem may be applied, consider a top-k GROUP-BY query where the goal is to compute an aggregate over the k "best" groups, where "best" is measured via the application of some aggregate

function over the group. If the database is sampled, then the aggregate total over each group is only an estimate, and computing the total error resulting from the inclusion (or exclusion) of groups that should not (or should) be there can likely be posed as an instance of the SI problem.

8. RELATED WORK

While not studied in the context of databases, the study of simultaneous or multiple inference has a long history in statistics. The classical reference material is Miller's book [Miller 1981]. Some newer summaries of the field are books by Hochberg and Tamhane [Hochberg and Tamhane 1987], Westfall and Young [Westfall and Young 1993], and Hsu [Hsu 1996].

Most results regarding simultaneous inference are related to *hypothesis testing*, which is the process of using observed data to make decisions about unobserved parameters of interest. In a hypothesis test, the tester defines a null hypothesis, a test statistic, and a cutting value. The *null hypothesis* is an assumption that an effect or result we may be interested in confirming is not true (for example, we may be interested in determining whether an HIV drug is effective; the natural null hypothesis assumes that the drug is not in fact effective). Once the null hypothesis is formulated, a *test statistic* is calculated using the observed data. The test statistic is used to compute the p-value, which is the probability that a test statistic would be obtained assuming the null hypothesis were true. If the p-value is less than the cutting value p, the null hypothesis is rejected. This results in a Type-I or false positive rate of p. The so-called Type-II error is the probability of not rejecting a null hypothesis that is in fact false, and describes the power of the test.

In multiple hypothesis testing, rather than dealing with a single null hypothesis, one instead has a large number of hypotheses that are to be evaluated using the same data. The problem is that since the same data is used for each test, an error on one test may increase the chance of an error on another. The easiest way to handle such potential correlations is to adjust the p-value associated with each test so that the overall chance of erroneously rejecting a hypothesis is kept manageable. There are a number methods for doing this: so called "single-step" procedures such as the classic Bonferroni procedure [Miller 1981] and Sidak's procedure (both outlined nicely in Dragici [2003]), "step-down" procedures such as Holm's procedure [Holm 1979], and "step-up" procedures such as Hochberg's procedure [Hochberg 1988]. Essentially, given a vector of p-values, these methods all attempt to reject some of the corresponding hypotheses so as to guarantee that the probability of making any Type-I is less than p. They vary in power (Type-II error) and how they decide to associate a cutting value with each p-value to be able to guarantee a low enough Type-I error. In practice, all of these methods tend to be rather conservative.

Despite the wealth of related work from statistics, only relatively recently have statisticians begun to address issues similar to those considered in this article. One common characteristic of all of the classic statistical methods for simultaneous inference (as well as those discussed thus far) is that the goal is to control the so-called "family-wise error rate"; that is, they seek to control the

probability of falsely rejecting *any* of the hypotheses. In this sense, these methods are very restrictive compared to the SI problem, which can be used to control the probability that *any number* of estimates are incorrect. Only in the last few years have statisticians considered related questions. Work (from a statistics point of view) that is considered groundbreaking in its attempt to address this is due to Benjamini and Hochberg [1995], who seek to control the expected fraction of rejected hypotheses that are actually valid in general-purpose hypothesis testing. This is somewhat similar to our approach in providing "safe" bounds for GROUP BY queries, where the user is allowed to specify an error rate for the different confidence intervals. Perhaps the work from statistics that is closest to our own is the recent (and also well-known) contribution of Storey [2002], who considers the problem of computing the expected error rate for a given set of p-values. In a sense, this is related to the SI problem itself in that we are attempting to infer the distribution of this statistic for a given confidence region.

Both the work of Benjamini and Hochberg and that of Storey has seen significant follow-up work. But while this new line of research in the statistical literature is related to the methods proposed in here, the differences are significant. First, there is the obvious difference that our emphasis is on database estimation problems. Beyond this, it would be incorrect to assert that this article is a straightforward application of existing work from statistics. In one sense, our work is more targeted, in that it deals specifically with confidence regions over normally distributed estimates where covariances are easily estimated, as one would expect in a database environment. The work of Storey and Benjamini and Hochberg and later follow-up research deals with general-purpose hypothesis testing and an unknown covariance structure. In another sense, our work generalizes this existing work in that we allow an arbitrary penalty value to be associated with each incorrect interval. Another key difference is that our work has a distinctly computational focus. Our interest is in efficiently scaling the analysis to tens of thousands of simultaneous estimates computed from millions of data points that are then joined using complex database operations. This computational emphasis is far from what one finds in the statistics literature.

This article has generally been concerned with database sampling. The study of sampling has a very long history in databases, with most prior research focusing on how to employ sampling in a relational database. Early work on using samples to produce estimates for answers to database queries is due to Hou et al. [1988, 1989]; Lipton et al. [1990], and many others. Early on, researchers also developed methods to obtain random samples from database files, with the most notable work due to Olken and Rotem [1989] and Olken et al. [1990].

However, the study of sampling specifically for the answer to GROUP BY queries is actually rather limited in the database literature. Acharya et al. [2000] proposed the use of so-called *congressional samples* to approximately answer GROUP BY queries with high accuracy. Congressional samples combine both uniform and precomputed nonuniform samples to maximize the accuracy of answering a GROUP BY query. However, Acharya et al.'s work tackles a quite different

problem from what we have considered here. Their focus was on increasing estimate accuracy, while we focus on accurately *describing* estimate accuracy. Hellerstein et al. [1997] also considered sampling for GROUP BY queries, and proposed the *index striding* technique that controls the sampling rates of the various groups so as to maintain fairness in the amount of system resources devoted to each group. This method is based upon the observation that the number of records in each group can be different, so it may be important to sample small groups more heavily than big ones. Charikar et al. [2000] addressed the problem of estimating the number of distinct values over a database table column, which is equivalent to estimating the number of groups while grouping over this column. In fact, sampling for GROUP BY queries was the motivation for their work, since they sought a method to give the user an idea of how many groups were present in the database, in case not all of the groups had been encountered in the sample. Charikar et al. proved that any estimator for this problem cannot guarantee a small error bound for all population distributions, unless a large portion of the records in the table are provided. They provide an optimal estimator that matches their lower error bound, and also provide several heuristic estimators that can work better for specific distributions.

9. CONCLUSION

We have considered in depth the statistical issues that must be addressed if a single sample is used to answer a GROUP BY query. The problem is that since the same sample is used for each group, it is unacceptable to use classic, univariate methods to quantify error since the groupwise estimates are not independent—if one bound is wrong, then it may be likely that all of the bounds are wrong. Statistically speaking, univariate bounds are only valid in isolation, and once a user has seen one of them he or she needs to forget about that bound before looking at the next one. Since this is unreasonable in practice, we have considered what information should be given to the user to safely quantify the accuracy of a GROUP BY query result, and also discussed in depth how to compute such information efficiently.

APPENDIX

A. Covariance Analysis

This section considers the problem of how to formally analyze the variance of a single query and the covariance for two queries using the Haas-Hellerstein estimator [Haas and Hellerstein 1999] for SUM queries over one or more database tables.

A.1 Analysis for SUM Over Two Database Tables

It is natural to begin by considering a SUM query over two database tables. Specifically,

SELECT SUM ($f(R, S)$)
FROM R, S

Note that the function f can encode any mathematical function over pairs from R and S (including a COUNT query). As discussed in Section 2, the above query may be used to perform the computation required for a single group in a GROUP BY query if f limits the sum to only those tuples belonging to the group in question.

Let $f_i(R,S)$ denote the function for estimator $\widetilde{M}_i$ that returns the value of $f(R,S)$ if the tuple (R,S) belongs to the ith group, and returns zero otherwise. We denote by N_R and N_S the number of tuples of relations R and S, respectively. Furthermore, we denote by R' and S' the samples from R and S, respectively, and use n_R and n_S for their respective sizes. In order to perform the analysis, it is helpful to introduce Bernoulli (zero/one) random variables that indicate whether tuples from relations are not/are in the sample. Let X_k and Y_l be variables that indicate whether the kth and lth tuples from R and S are in R' and S', respectively. With this, the ith natural Haas-Hellerstein estimator of the aggregate is

$$\widetilde{M}_i = \frac{N_R}{n_R}\frac{N_S}{n_S}\sum_{k=1}^{N_R}\sum_{l=1}^{N_S} X_k Y_l\, f_i(k,l) \tag{27}$$

where the double sum is really a sum over the tuples of R' and S' expressed using the Bernoulli variables.

Before we can characterize the estimate $\widetilde{M}_i$, we need to derive some useful properties of the random variables X_k and Y_l. We present the necessary results and proofs only for X_k; the results for Y_l are obtained simply by substituting Y_l for X_k and S for R in all equations. First, some notation:

$$\alpha_R = \frac{N_R}{n_R}; \quad \beta_R = \frac{N_R - 1}{n_R - 1} \tag{28}$$

In the developments that follow we could have used the *Kronecker delta* symbol [Jermaine et al. 2005], instead, we introduce a function that leads to more intuitive formulas:

$$\rho_{k,k'}(t) = \begin{cases} 1 & k = k' \wedge t = 0 \\ 1 & k \neq k' \wedge t = 1 \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

Intuitively, this function allows us to turn on/off terms that have equal/not equal values for the indices k, k' based on value of the variable t. The following properties of $\rho_{k,k'}$ function will be used extensively:

PROPOSITION A.1. *For any function $\mathcal{F}(k,k')$, we have*

$$\forall k,k', \quad \sum_{t=0}^{1} \rho_{k,k'}(t) = 1 \tag{30}$$

$$\sum_k \sum_{k'} \rho_{k,k'}\mathcal{F}(k,k') = \begin{cases} \sum_k \mathcal{F}(k,k) & t=0 \\ \sum_k \sum_{k'\neq k} \mathcal{F}(k,k') & t=1 \end{cases} \tag{31}$$

$$\sum_{t=0}^{1}\sum_k \sum_{k'} \rho_{k,k'}\mathcal{F}(k,k') = \sum_k \sum_{k'} \mathcal{F}(k,k') \tag{32}$$

PROOF. To prove the first identity, we observe that irrespective of whether $k = k'$, $\rho_{k,k'}(t)$ takes value 1 for one $t \in \{0, 1\}$ and value 0 for the other, thus the sum is always 1. The second identity follows directly from the definition of ρ by observing that some terms vanish since they have 0 coefficient due to ρ. The third identity follows directly from the second, since the two cases in Eq. (31) are summed up and they complete the double sum. □

With these, we now have the following result that characterizes the Bernoulli random variables:

PROPOSITION A.2. *If R' is a sample without replacement of R, then for the kth tuple of R we have*

$$E[X_k] = \frac{1}{\alpha_R} \tag{33}$$

$$\rho_{k,k'}(t)E[X_k X_{k'}] = \rho_{k,k'}(t)\frac{1}{\alpha_R \beta_R^t} \tag{34}$$

PROOF. The expected value of a zero/one random variable is the probability of the variable taking the value 1; in this case $\frac{1}{\alpha_R}$ is the probability that the kth tuple of R is in R'. When $k = k'$, since $X_k^2 = X_k$, $E[X_k X_{k'}]$ is $\frac{1}{\alpha_R}$. When $k \neq k'$, $E[X_k X_{k'}] = P[X_k = 1 \wedge X_{k'=1}] = P[k \in R']P[k' \in R' | k \in R']$. Since the conditional probability is $\frac{1}{\beta_R}$, then $E[X_k X_{k'}] = \frac{1}{\alpha_R \beta_R}$. Using the definition of $\rho_{k,k'}(t)$, the two cases can be easily encoded as in the statement of the proposition. Note that ρ has to appear on the right side as well as on the left side to ensure the right side is 0 whenever the left side is. □

In the formulas derived subsequently, a particular set of terms appear that deserve special attention. We introduce and characterize them first.

For $t_R, t_S \in \{0, 1\}$ and the ith, jth estimators, we define

$$P_{t_R,t_S} = \sum_{k=1}^{N_R}\sum_{l=1}^{N_S}\sum_{k'=1}^{N_R}\sum_{l'=1}^{N_S}\rho_{k,k'}(t_R)\rho_{l,l'}(t_S)f_i(k,l)f_j(k',l') \tag{35}$$

Depending on values of t_R and t_S, these terms can be rewritten using the second identity in Proposition A.1 as sums of the form $f_i(\cdot)f_j(\cdot)$. For example,

$$P_{0,1} = \sum_{k=1}^{N_R}\sum_{l=1}^{N_S}\sum_{l'=1,l'\neq l}^{N_S} f_i(k,l)f_j(k,l') \tag{36}$$

An extra property of these terms is given by

PROPOSITION A.3.

$$\sum_{t_R=0}^{1}\sum_{t_S=0}^{1} P_{t_R,t_S} = \sum_{k=1}^{N_R}\sum_{l=1}^{N_S}\sum_{k'=1}^{N_R}\sum_{l'=1}^{N_S} f_i(k,l)f_j(k',l') \tag{37}$$

PROOF. The proof follows directly from the second identity in Proposition A.1 and simple reorganization of the sums. □

THEOREM A.4. *For any arbitrary estimators $\widetilde{M}_i$ and $\widetilde{M}_j$ given by Eq.* (27)*, we have*

$$E\left[\widetilde{M}_i\right] = \sum_{k=1}^{N_R}\sum_{l=1}^{N_S} f_i(k,l) \tag{38}$$

$$\mathrm{Cov}\left(\widetilde{M}_i, \widetilde{M}_j\right) = \sum_{t_R=0}^{1}\sum_{t_S=0}^{1}\left(\frac{\alpha_R}{\beta_R^{t_R}}\frac{\alpha_S}{\beta_S^{t_S}} - 1\right) P_{t_R,t_S} \tag{39}$$

PROOF. Using linearity of expectation and the first identity in Proposition A.2, the unbiasedness of the expectation of $\widetilde{M}_i$ follows immediately. □

To compute the covariance, we first estimate $E\left[\widetilde{M}_i\widetilde{M}_j\right]$.

$$E\left[\widetilde{M}_i\widetilde{M}_j\right] = \alpha_R^2\alpha_S^2 \sum_{k=1}^{N_R}\sum_{k'=1}^{N_R}\sum_{l=1}^{N_S}\sum_{l'=1}^{N_S} E\left[X_k X_{k'}\right] E\left[Y_l Y_{l'}\right] f_i(k,l) f_j(k',l') \tag{40}$$

To rewrite this term, observe that by multiplying each term within the summation by $\sum_{t_R=0}^{1}\rho_{k,k'}(t_R)$ and $\sum_{t_S=0}^{1}\rho_{l,l'}(t_S)$ (both these multipliers are 1 by the first identity in Proposition A.1 thus they do not change the identity) and regrouping the sums, we obtain

$$E\left[\widetilde{M}_i\widetilde{M}_j\right] = \alpha_R^2\alpha_S^2 \sum_{t_R=0}^{1}\sum_{t_S=0}^{1}\sum_{k=1}^{N_R}\sum_{k'=1}^{N_R}\sum_{l=1}^{N_S}\sum_{l'=1}^{N_S} \rho_{k,k'}(t_R) E[X_k X_{k'} \rho_{l,l'}(t_S) E\left[Y_l Y_{l'}\right] f_i(k,l) f_j(k',l') \tag{41}$$

Now, using the second identity in Proposition A.2 and the definition of P_{t_R,t_S} we get

$$E\left[\widetilde{M}_i\widetilde{M}_j\right] = \sum_{t_R=0}^{1}\sum_{t_S=0}^{1}\frac{\alpha_R}{\beta_R^{t_R}}\frac{\alpha_S}{\beta_S^{t_S}} P_{t_R,t_S} \tag{42}$$

Using now the fact that $\mathrm{Cov}\left(\widetilde{M}_i\widetilde{M}_j\right) = E[\widetilde{M}_i\widetilde{M}_j] - E[\widetilde{M}_i]E[\widetilde{M}_j]$ and the result in Proposition A.3 we obtain the required result. □

Note that the above result does not require that i and j be different; thus, the result applies to the case when $i = j$. This will give us the variance of $\widetilde{M}_i$.

A.2 Analysis for SUM over Multiple Database Tables

We now extend the previous result to an arbitrary number of database tables. We assume that $T_1, T_2, \ldots, T_k$ are k database tables, and consider the queries of the form

```
SELECT SUM ( f(T1, T2, ... , Tk))
FROM T1, T2, ... , Tk
```

Just as before, we let $f_i(T_1, \ldots T_k)$ be a function that returns $f(T_1, \ldots T_k)$ if the tuple $(T_1, \ldots, T_k)$ belongs to group i, and 0 otherwise. The numbers of

tuples in these tables are $N_1, N_2, \ldots, N_k$, respectively. In addition, it is also assumed the sample sizes are $n_1, n_2, \ldots, n_k$, respectively. We define Bernoulli random variables $X_{w1}, \ldots X_{wk}$that will govern whether or not the wth tuple from $T_1, \ldots, T_k$ are sampled, respectively. The following then serve as the ith estimator for the sums of f:

$$\widetilde{M}_i = \frac{N_1 \ldots N_k}{n_1 \ldots n_k} \sum_{w1,\ldots,wk} X_{w1} \ldots X_{wk} f_i(w_1, \ldots, w_k) \tag{43}$$

As in the previous section, in order to derive the covariance of $\widetilde{M}_i$ and $\widetilde{M}_j$, for u in 1 to k we have

$$\alpha_u = \frac{N_u}{n_u}; \quad \beta_u = \frac{N_u - 1}{n_u - 1} \tag{44}$$

$$P_{t_1,\ldots,t_k} = \sum_{w_1=1}^{N_1} \sum_{w_1'=1}^{N_1} \cdots \sum_{w_k=1}^{N_k} \sum_{w_k'=1}^{N_k} \prod_{u=1}^{k} \rho_{w_u,w_{u'}}(t_u) \times f_i(w_1, \ldots, w_k) f_j(w_1', \ldots, w_k') \tag{45}$$

THEOREM A.5. ***The covariance for multiple database tables.*** *The covariance of $\widetilde{M}_i$ and $\widetilde{M}_j$ is*

$$Cov(\widetilde{M}_i \widetilde{M}_j) = \sum_{t_1=0}^{1} \ldots \sum_{t_k=0}^{1} \left(\prod_{u=1}^{k} \frac{\alpha_u}{\beta_u^{t_u}} - 1 \right) P_{t_1,\ldots,t_k} \tag{46}$$

PROOF. The proof mirrors the proof of Theorem A.4. k terms of the form $\sum_{t_u=0}^{1} \rho_{w_u,w_{u'}}(t_u)$ are multiplied with terms in the multiple summation of the expansion of $E\left[\widetilde{M}_i \widetilde{M}_j\right]$, then the results in Propositions A.2 and A.1, the definition of $P_{t_1,\ldots,t_k}$, the generalization of Proposition A.3 and summation regrouping give the required result. □

Although the analysis above was performed in the context that all the relations are used for each query, the queries do not need to operate over the same set of input relations.For example, imagine that query Q_1 computes a sum over f_1 and operates over relations T_1 and T_2, and Q_2 uses f_2 and operates over T_2 and T_3, where both use the same sample from T_2. In such a situation, the analysis may be used by re-writing both queries to operate over the union of the set of input relations (T_1, T_2, and T_3). Q_1 would be changed to compute the sum over a function $f_1{}'(t_1, t_2, t_3) = f_1(t_1, t_2)/|T_3|$, and Q_2 would use the function $f_2{}'(t_1, t_2, t_3) = f_2(t_2, t_3)/|T_1|$.

A.3 An Unbiased Covariance Estimator

In order to compute the covariances using the formulas described above, it is necessary to have access to the entire database table (which is obviously not feasible if sampling is performed). As a result, it is useful to obtain an unbiased estimator for the covariance of two estimators, given only the sample that is used to estimate the answer to the query.

Consider the following estimator:

$$\widetilde{\mathrm{Cov}}(\widetilde{M}_i, \widetilde{M}_j) = \sum_{t_1=0}^{1} \cdots \sum_{t_k=0}^{1} \left(\prod_{u=1}^{k} \frac{\alpha_u}{\beta_u^{t_u}} - 1 \right) \prod_{v=1}^{k} \alpha_v \beta_v^{t_v} \times \sum_{w_1=1}^{N_1} \sum_{w_1'=1}^{N_1} \cdots \sum_{w_k=1}^{N_k} \sum_{w_k'=1}^{N_k} \rho_{w_1,w_1'}(t_1) \ldots \rho_{w_k,w_k'}(t_k) \times X_{w_1} X_{w_1'} \ldots X_{w_k} X_{w_k'} f_i(w_1, \ldots, w_k) f_j(w_1', \ldots, w_k') \quad (47)$$

It is simple to prove that this estimator is unbiased for the covariance of two estimators. To show this, we note that the expected value of

$$\sum_{w_1=1}^{N_1} \sum_{w_1'=1}^{N_1} \cdots \sum_{w_k=1}^{N_k} \sum_{w_k'=1}^{N_k} \rho_{w_1,w_1'}(t_1) \ldots \rho_{w_k,w_k'}(t_k) \times X_{w_1} X_{w_1'} \ldots X_{w_k} X_{w_k'} f_i(w_1, \ldots, w_k) f_j(w_1', \ldots, w_k') \quad (48)$$

is

$$\frac{1}{\prod_{v=1}^{k} \alpha_v \beta_v^{t_v}} P_{t_1,\ldots,t_k} \quad (49)$$

Therefore, the expected value of $\widetilde{\mathrm{Cov}}(\widetilde{M}_i, \widetilde{M}_j)$ is the covariance of $\widetilde{M}_i$ and $\widetilde{M}_j$.

REFERENCES

ACHARYA, S., GIBBONS, P., POOSALA, V., AND RAMASWAMY, S. 1999a. Join synopses for approximate query answering. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*. ACM, New York, 275–286.

ACHARYA, S., GIBBONS, P. B., POOSALA, V., AND RAMASWAMY, S. 1999b. The aqua approximate query answering system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*. ACM, New York, 574–576.

ACHARYA, S., GIBBONS, P. B., AND POOSALA, V. 2000. Congressional samples for approximate answering of group-by queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*. ACM, New York, 487–498.

BENJAMINI, Y. AND HOCHBERG, Y. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. Royal Statisti. Soc. 57*, 289–300.

CASELLA, G. AND BERGER, R. L. 2002. *Statistical Inference.* 2nd Ed. Duxbury. CAS g^2 02:1 1.Ex.

CHARIKAR, M., CHAUDHURI, S., MOTWANI, R., AND NARASAYYA, V. R. 2000. Towards estimation error guarantees for distinct values. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, New York, 268–279.

CHAUDHURI, S., DAS, G., AND NARASAYYA, V. 2001. A robust, optimization-based approach for approximate answering of aggregate queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*. ACM, New York, 295–306.

DOBRA, A., GAROFALAKIS, M., GEHRKE, J., AND RASTOGI, R. 2002. Processing complex aggregate queries over data streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'02)*. ACM, New York, 61–72.

DRAGICI, S. 2003. *Data Analysis Tools for DNA Microarrays*. Chapman and Hall, CRC Press.

GANTI, V., LEE, M.-L., AND RAMAKRISHNAN, R. 2000. Icicles: Self-tuning samples for approximate query answering. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*. Morgan Kaufmann, 176–187.

GIBBONS, P. B. AND MATIAS, Y. 1998. New sampling-based summary statistics for improving approximate query answers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*. ACM, New York, 331–342.

HAAS, P. J. AND HELLERSTEIN, J. M. 1999. Ripple joins for online aggregation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*. ACM, New York, 287–298.

HELLERSTEIN, J. M., AVNUR, R., CHOU, A., HIDBER, C., OLSTON, C., RAMAN, V., ROTH, T., AND HAAS, P. J. 1999. Interactive data analysis: The control project. *Computer 32,* 8, 51–59.

HELLERSTEIN, J. M., HAAS, P. J., AND WANG, H. J. 1997. Online aggregation. In *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD'97)*. ACM, New York, 171–182.

HOCHBERG, Y. 1988. A sharper bonferroni procedure for multiple tests of significance. *Biometrika 75*, 800–802.

HOCHBERG, Y. AND TAMHANE, A. C. 1987. *Multiple Comparison Procedures*. Wiley, New York.

HOLM, S. 1979. A simple sequentially rejective multiple test procedure. *Scand. J. Stat 6*, 65–70.

HOU, W.-C., ÖZSOYOGLU, G., AND TANEJA, B. K. 1988. Statistical estimators for relational algebra expressions. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'88)*. ACM, New York, 276–287.

HOU, W.-C., ÖZSOYOGLU, G., AND TANEJA, B. K. 1989. Processing aggregate relational queries with hard time constraints. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'89)*. ACM, New York, 68–77.

HSU, J. 1996. *Multiple Comparisons: Theory and Methods*. Chapman and Hall, CRC Press.

JERMAINE, C., DOBRA, A., ARUMUGAM, S., JOSHI, S., AND POL, A. 2005. A disk-based join with probabilistic guarantees. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*. ACM, New York, 563–574.

JOHNSON, N. L., KOTZ, S., AND BALAKRISHNAN, N. 1995. *Continuous Univariate Distributions Vol. 2*, Wiley, New York.

LIPTON, R. J., NAUGHTON, J. F., AND SCHNEIDER, D. A. 1990. Practical selectivity estimation through adaptive sampling. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*. ACM, New York, 1–11.

MILLER, R. G. 1981. *Simultaneous Statistical Inference*, 2nd ed. Springer, Berlin, Germany.

OLKEN, F. AND ROTEM, D. 1989. Random sampling from b+ trees. In *Proceedings of the Conference on Very Large Data Bases (VLDB'89)*. 269–277.

OLKEN, F., ROTEM, D., AND XU, P. 1990. Random sampling from hash files. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*. ACM, New York, 375–386.

ROBERT, C. P. AND CASELLA, G. 2005. *Monte Carlo Statistical Methods.* Springer, New York.

SARNDAL, C., SWENSSON, B., AND WRETMAN, J. 1992. *Model Assisted Survey Sampling*. Springer, Berlin, Germany.

STOREY, J. D. 2002. A direct approach to false discovery rates. *J. Royal Statist. Soc. Series B 64*, 479–498.

WESTFALL, P. AND YOUNG, S. 1993. *Resampling-Based Multiple Testing*. Wiley, New York.

Received August 2006; revised June 2007, December 2007; accepted April 2008