Error-Constrained COUNT Query Evaluation in Relational Databases[*]

Wen-Chi Hou[+], *Gultekin Ozsoyoglu*[-], *and Erdogan Dogdu*[-]

ABSTRACT

An error-constrained COUNT query in relational algebra is of the form "Estimate COUNT(E), where E is a relational algebra expression, such that the error in the estimate is at most e". The general approach is to evaluate an estimator for COUNT(E) using a large enough sample from input relations such that the error is less than the specified bound with a certain confidence level.

We present an approach which utilizes double sampling for error-constrained COUNT queries. We compare this approach with another approach, called adaptive sampling, in terms of the sample sizes needed to obtain the desired error bound with a given confidence level.

We have implemented both approaches (i.e., adaptive sampling and double sampling) in a prototype, real-time DBMS called CASE-MDB. We present some of the results of our performance evaluation experiments.

1. Introduction

In this paper, we discuss the problem of estimating a COUNT(E) query having a guaranteed error bound with a certain confidence level, where E is an arbitrary relational algebra (RA) expression with Select, Join and Intersection operations. Such an estimate is useful in query optimization as well as in real-time databases as it takes significantly less time to obtain compared with evaluating COUNT(E) completely. To guarantee the desired error bound with the given confidence level, we use *double sampling* (or *two-phase sampling*) where in the first stage preliminary information is obtained from a small pilot sample taken from the input relations. And then, the sample size for the second stage is determined such that the estimator is guaranteed to produce an estimate with the desired error bound and the given confidence level.

There is only one other approach in the literature discussing the error-constrained COUNT(E) estimation problem. In [LiNa 89, 90, LNS 90], random sample tuples are repetitively taken until the number of output tuples meets a lower bound (i.e., the stopping condition) for a given relative error and a confidence level. The lower bound was first derived in [LiNa 90] and later refined in [LNS 90] by assuming a normal distribution for the outcomes of the samples. This refinement is based on the Central Limit Theorem. We will call the associated algorithm the LNS algorithm. Detailed comparisons between LNS algorithm and our approach are presented later.

In section 2, we briefly summarize the estimator we use. Section 3 discusses two basic sampling techniques, namely, simple random sampling and systematic sampling, which can be used as the underlying sampling methods for double sampling. Section 4 describes and compares adaptive sampling and double sampling. Section 5 improves the double sampling for the use of simple random sampling without (as opposed to "with") replacement. Section 6 describes the experimental results that we have run on our prototype DBMS, called CASE-MDB. Section 7 concludes.

2. Estimators for COUNT Queries

In [HoOT 88, HoOz 88], COUNT(E) is computed as $\sum_i (\pm) COUNT(E_i)$, where E is an arbitrary RA expression and E_i is an RA expression containing only Select, Join, Intersection, Project and Difference (if any[+]) operations. Different estimators are used for different COUNT(E_i) expressions. However, in this paper, we will restrict ourselves to RA expressions with only Select, Join and Intersection operations.

2.1. Point Space Model

A relation instance r with $|r|$ tuples is modeled as a finite one-dimensional space with a set of $|r|$ points. A Select-Join-Intersection (SJI-) expression E with n operand relations is modeled as a finite n-dimensional space, called *the point space of E*, such that

* This research is supported by the National Science Foundation under Grants IRI-8811057, IRI-9009897, and IRI-9008632.

+ Department of Computer Science, Southern Illinois University at Carbondale, Carbondale, IL 62901.

- Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH 44106.

+ If difference operators do appear in E_i, they are always preceded by projection operations.

(i) Each relation corresponds to one dimension : there are totally $\prod_{i=1}^{i=n} |r_i|$ points in the point space, where $|r_i|$ is the number of tuples in relation r_i, assuming that the n different operand relations are numbered from 1 to n. Each point uniquely maps to a sequence of n-tuples $(t_1, ..., t_n)$, where $t_i \in r_i$, $1 \leq i \leq n$.

(ii) A point in the point space assumes the value 1 if the set of corresponding n-tuples $(t_1, ..., t_n)$ produces an output tuple when substituted into E; otherwise the point has the value 0.

2.2. SJI-Expressions

For a Select-Join-Intersection (SJI-) expression E, computing COUNT(E) is equivalent to counting the number of points with the value 1 in the point space. In [HoOT 88, HoOz 88], a consistent and unbiased estimator, denoted by $\hat{Y}$, is proposed for estimating the number of 1's in the point space. $\hat{Y}$ is defined as $N \cdot (y / m)$, where N is the total number of points in the point space of E, m is the number of sample points, and y is the number of sample points with the value 1.

3. Sampling Techniques

In this section, we discuss two commonly used sampling techniques, namely, simple random sampling and systematic sampling [HaHM 53, Coch 77, Sukh 84]. These two sampling techniques (and others) have been implemented in two DBMSs, called CASE-DB and CASE-MDB, as the underlying sampling methods for an error-constrained query processing technique, called double sampling.

Simple random sampling (SRS) is a method of selecting m elements (sample size) out of N (the population size) such that each possible sample of m elements has an equal chance of being selected. If an element that is already selected is removed from the population for all subsequent draws, this method is also called simple random sampling *without replacement*. On the other hand, if the element drawn is placed back to the population for subsequent draws, the method is called simple random sampling *with replacement*. Simple random sampling has simple mathematical properties, and serves as a prelude to other sampling techniques.

Assuming that we have an ordering among the population elements, systematic sampling takes a unit at random from the first k elements and every k^{th} unit from there on. Systematic sampling (SS) is often said to follow the every k^{th} rule [Coch 77]. The procedure is simple; however, the performance depends on the properties of the population [Yama 67]. The technique may greatly improve the estimate, as compared to simple random sampling, for those populations in which elements are ordered, while it may deteriorate the estimate for those populations with periodic variations.

4. Error-Constrained Query Evaluation - Three Approaches

In order to compare error-constrained approaches, we assume that

(1) error constraints (or the precision requirements) are always specified in relative errors with confidence levels, e.g., a 10% error with a 95% confidence level, and

(2) simple random sampling with replacement is the underlying sampling method in this section.

However, in Section 5, we will revise the methodology to incorporate simple random sampling without replacement. In section 6, we will also use systematic sampling for comparison purposes.

4.1. Known Population Characteristics

Consider an ideal situation where the characteristics of the population (i.e., the variance and the mean) are known. Then, for a given error constraint, the required sample size can be easily obtained as follows.

Let N be the population size and m be the sample size. Let y_i be the value of an individual unit in the population, Y be the *population total* (i.e, $\sum_{i=1}^{i=N} y_i$), $\bar{Y}$ be the *population mean* (i.e., $\bar{Y} = Y / N$), and $\bar{y}$ be the *sample mean* (i.e., $\bar{y} = (\sum_{i=1}^{i=m} y_i) / m$). To obtain an estimate of the population total Y (or the population mean $\bar{Y}$) precise to a given relative error bound e with a confidence level l (where l equals $1-\alpha$), we have

$$Pr\left(\left| \frac{(N/m)\sum_{i=1}^{i=m} y_i - Y}{Y} \right| \geq e \right)$$

$$= Pr\left(\left| \frac{\bar{y}-\bar{Y}}{\bar{Y}} \right| \geq e \right)$$

$$= Pr(|\bar{y}-\bar{Y}| \geq e\bar{Y}) = \alpha \qquad (4.1)$$

where Pr denotes the probability, m is the required sample size, and α is the risk of having a higher relative error than e. If a sample of size m is randomly taken with replacement then the standard error $\sigma_{\bar{y}}$ is defined as

$$\sigma_{\bar{y}} = \frac{\sigma}{\sqrt{m}} \qquad (4.2)$$

where σ^2, the variance of the population, is defined as $\sum_{i=1}^{i=N}(y_i - \bar{Y})^2 / N$. Let us assume that the sample size is large enough for the Central Limit Theorem to apply (i.e., $\bar{y}$ is approximately normally distributed), we have

$$e\bar{Y} = t\sigma_{\bar{y}} \qquad (4.3)$$

where t is the abscissa of the standard normal curve that cuts off an area of α at the tails. Solving for m, we have

$$m = \left(\frac{t\sigma}{e\bar{Y}}\right)^2 \qquad (4.4)$$

Unfortunately, in an error-constrained environment, information such as variance and mean of the population is usually not available. In the tables of Section 4.4, we list the required sample sizes for various error constrains computed from equation (4.4) under the column entitled "Reference sizes". These columns can be used as an indices to show how efficient an error-constrained algorithm is. Note that any error-constrained algorithm without the knowledge of variance and mean can not have a better performance (i.e., a smaller sample size) than in the idea situation.

4.2. Sequential Sampling

Sequential sampling [Wald 45] has been used in error-constrained environments in social sciences, industry, etc., when there is no or little knowledge about the population at hand. It is characterized by its sequential sample gathering (i.e., taking the samples step by step) and its stopping condition. That is, sample units are taken one at a time. By checking the outcome of each sample unit, a decision (i.e., the stopping criterion) as to whether an additional sample unit is to be taken is made. The stopping criterion sets a lower bound on the required sample size for a given error constraint. Deriving stopping conditions is a major concern on the efficiency of a sequential sampling method, since, though an "oversized" sample may produce an estimate satisfying the required precision requirement, it may also result in a waste of resources. On the other hand, too small a sample may not be enough to produce an estimate satisfying the required precision.

4.2.1. Adaptive Sampling and LNS algorithm

Lipton and Naughton propose *Adaptive Sampling* [LiNa 89, 90, LNS 90] for error-constrained queries by taking a sample unit one at a time. Stopping conditions used to determine the required sample size are also presented. Clearly, adaptive sampling is a variant of sequential sampling.

Lipton and Naughton [LiNa 89, LiNa 90] use an urn model to lay down the theoretical framework for estimating the size of a Select-Join(-Intersection) query by sampling. The output tuples of the SJI-expression is subdivided into disjoint sets, and each disjoint set is represented by a ball in the urn model. Each ball has the size of the disjoint set as its value. In an SJI-expression with n operand relations $r_1, ..., r_n$, a ball corresponds to a tuple t in a designated relation, say r_1, and the ball value corresponds to the number of output tuples of an SJI-expression produced by tuple t and relations $r_2, ...,$ and r_n.

The algorithm proposed in [LiNa 89, 90, LNS 90] (i.e., the LNS algorithm) repetitively draws and evaluates sample tuples from a relation until the number of output tuples of the SJI-expression meets certain stopping conditions for a given error constraint. There are two stopping conditions in [LNS 90], one for the cases in which a "reasonable amount" of balls have non-zero values, and the other for those cases where few balls have non-zero values, e.g, an extremely low selectivity for the SJI-expression. Without loss of generality, here we compare only the first stopping criterion $y > k_1 \cdot b \cdot \frac{1}{e} \cdot (\frac{1}{e}+1)$ with our results (i.e, assuming a reasonably large selectivity), where y is the number of output tuples accumulated so far, k_1 is a value associated with a given confidence level, b is the maximum size of the disjoint sets, and e is the desired limit of a relative error. Unfortunately, the b value is usually unknown, which determines the efficiency of LNS algorithm. In such cases, a guess value, which may be much larger than the maximum size of the disjoint sets, has to be used.

When preliminary information about the population is available, the performance of LNS algorithm can be significantly improved. That is, instead of using a guess value for b in the stopping condition, one can use $\hat{V} \;/\; \hat{Y}$ for b [LiNS 90], where $\hat{V}$ is an estimate of the the population variance and $\hat{Y}$ is an estimate of the population mean. To obtain such information, one possible approach is to take a pilot sample, just as what we do in the first stage of double sampling (to be discussed in next section). Hereafter, we will call the LNS algorithm which uses preliminary information $\hat{V} \;/\; \hat{Y}$) to determine the stopping condition the *Improved LNS (ILNS) algorithm.* Performance of both LNS and ILNS will be presented in Section 4.4.

4.3. Double Sampling and Our Approach

In *Double sampling*, a sample is taken in two steps [Cox 52, Coch 77]. Cox [Cox 52] has shown that double sampling has a performance close to that of the optimal sequential sampling method, if exists. Another advantage is the reduced overhead, because a required sample is taken in two steps, rather than m steps as in a regular sequential sampling method (including adaptive sampling), where m is the sample size. In the first step of double sampling, a sample of size, say, m_1, is taken to obtain a preliminary information about the mean and variance of the population. Based on this preliminary information, the required sample size, say, m, that guarantees that the estimate meets the precision requirement with a certain confidence level is computed. In the second step, additional (i.e., $m - m_1$) sample units, are taken, and the final estimate is produced. Note that the first step sample is not only used to provide preliminary information, but it also constitutes a part of the final sample of size m.

An SJI-expression with n operand relations $r_1, ..., r_n$, is modeled as an n-dimensional point space in our point space model with points having values 0 and 1. The urn model (which is one-dimensional) can be considered as a simplification of the n-dimensional point space model by merging $n-1$ dimensions, say $r_2, ..., r_n$. Here, by merging $n-1$ dimensions we mean that all the points which have the same value for r_1 coordinate, regardless of their $r_2, ...,$ and r_n coordinate values, are merged into a single point (which then becomes a ball of the urn model). Each point after the merge (i.e., a ball) has as its value the sum of the values of the set of original points in the original n-dimensional space.

There are some differences worth mentioning between the point space model and the urn model.

(1) In the point space model, sample tuples can be drawn from any or all of the relations while, in the urn model, sample tuples can be drawn only from a single relation.

(2) The values of points in the point space model can be obtained simply by evaluating the SJI-expression using sample tuples as input without using index files on the join attributes. In the LNS approach, the value of a ball has to be obtained using index files. Index files are used to provide fast access to the entire relations. Therefore, from a sampling point of view, using index files of $r_2, \cdots$ and r_n is equivalent to using entire relations $r_2, \cdots$ and r_n as samples, as far as the sample size (in terms of points) is concerned. When index files are available, point space model can also take this advantage (by merging dimemsions) just as LNS's urn model.

In order to compare the efficiency of different error-constrained algorithms on the same basis. Hereafter, in the remainder of Section 4, we will assume that the urn model (a special case of the point space model) is the underlying model. That is, sample tuples (i.e., the balls) are taken from a single relation in both the double sampling and the adaptive sampling, and the ball values can be immediately determined using index files. In Figure 4.1, we present the error-constrained algorithm using double sampling, revised to sample only from r_1 so that it can be compared meaningfully with the LNS algorithms.

4.3.1. Determining the Sample Size

For double sampling and ILNS algorithm, the database designer has to determine how to compute the first step sample size. It can be determined based on precision requirements, experience or, simply, using a fixed number. For simplicity, we assume that the DBMS always chooses 100 tuples from r_1 as the first step sample size for the comparisons in Section 4.4.

When the error is specified as the *coefficient of variation* $\sqrt{C}$, defined as $\sigma_{\bar{y}} / \bar{Y}$, the formula for computing the total sample size m [Cox 52] is

$$m = \frac{v^2}{C\bar{y}^2}\left(1+8C+\frac{v^2}{m_1\bar{y}^2}+\frac{2}{m_1}\right) \qquad (4.5)$$

```
Algorithm Estimate-COUNT(E, e, l)
Input  : E: an SJI-expression with n operand
         relations;
         e : a given relative error;
         l : a given confidence level.
Output : an estimate of COUNT(E) satisfying the
         given error constraint.
Var      t : the abscissa of the normal curve that cuts
         off an area of 1 - l at the tails;
         $m_1$ : the first step sample size;
         $\bar{y}$ : sample mean of the first step sample;
         $v^2$ : sample variance of the first step sample;
         m : the overall sample size;
         N : population size;
         $\hat{Y}$ : an estimate of COUNT(E);

Begin
    Determine and draw the first step sample of size
    $m_1$ from relation $r_1$;
    Evaluate $y_i$ (ball) value, $1 \le i \le m_1$, for each
    sample tuple in $m_1$;
    $\bar{y} := \sum_{i=1}^{i=m_1} y_i / m_1$;
    $v^2 := \sum_{i=1}^{i=m_1} (y_i - \bar{y})^2 / m_1$;
    $m := (\frac{tv}{e\bar{y}})^2\left(1 + 8(\frac{e}{t})^2 + \frac{v^2}{m_1\bar{y}^2} + \frac{2}{m_1}\right)$;
    {Compute the overall sample size m}

    if $m > m_1$ then Evaluate additional $m - m_1$
              sample units as the second step
    else  $m := m_1$;
    $\hat{Y} := N \cdot \sum_{i=1}^{i=m} y_i / m$;
    Return $(\hat{Y}(1 - 2(\frac{e}{t})^2))$;    { adjust bias}
End.
```

Figure 4.1. COUNT Query Estimation Algorithm Using Double Sampling

where m is the total sample size, m_1 is the sample size at the first step, v^2, defined as $\sum_{i=1}^{i=m_1} (y_i - \bar{y})^2 / m_1$, is an estimate of the population variance σ^2 from the first sample.

We can rewrite equation (4.5) in terms of the relative error e and the confidence level l using equation (4.3) as

$$m = \left(\frac{tv}{e\bar{y}}\right)^2\left(1 + 8\left(\frac{e}{t}\right)^2 + \frac{v^2}{m_1\bar{y}^2} + \frac{2}{m_1}\right) \qquad (4.6)$$

Note that $m - m_1$ is the additional sample units to be taken at the second step, if $m > m_1$. When this is not true, i.e., $m \le m_1$, it means that the sample size is already large enough to obtain the required accuracy, and one can simply stop the process without going on to the second step and return the estimate from the first step sample result as the final result. Also one can see from equation (4.5) that, the larger the first sample size m_1 is, the smaller the total sample size m will be, assuming $m > m_1$.

The estimator Y becomes slightly biased; to adjust the bias, Cox suggests to take $Y(1-2(e\ /\ t)^2)$ as the final estimate for the size of the query, which is what the algorithm in Figure 4.1 returns. Please note the similarity between equations (4.4) and (4.6). That is, the first term in equation (4.6) is the sample size needed when the population variance and the mean are known (i.e., equation (4.4)), and the 2nd, 3rd and 4th terms of equation (4.6) are the adjustment terms (i.e., the extra cost) for not knowing the variance and mean in an error-constrained environment. That is, information about the population may help reduce the sample size.

Cox [Cox 52] also provides another formula for computing the sample size when the population has only 0 and 1 values. Clearly, the point space model meets this condition. The formula in [Cox 52], after some modifications as above, is written as

$$m = \frac{(\frac{t}{e})^2 (1-p_1)}{p_1} + \frac{3}{p_1(1-p_1)} + \frac{t^2}{e^2 p_1 m_1} \qquad (4.7)$$

where p_1 is the proportion of the sample units having the value 1 at the first step. This formula gives a better performance (i.e., a smaller total sample size m) than equation (4.6) when the population has only 0 and 1 values, which is the case in the point space model. However, equation (4.6) is more general, and can be applied to both the point space model and the urn model. The estimator is also slightly biased. To reduce the bias, Cox [Cox 52] suggests to take $N(p - (\frac{e}{t})^2 p\ /\ (1 - p))$ as the final estimate, where p is the proportion of the overall sample units with the value 1. We will use this formula for our experiments in Section 6.

4.4. Comparison of the Three Approaches

In this section, we compare the performance of the double sampling algorithm described in Figure 4.1 (i.e., we only sample from r_1) with LNS and ILNS algorithms. Since LNS and ILNS algorithms are developed for only select, join and intersection operators, here we only consider these operators.

4.4.1. Select Operator

Let us now consider the select operation, i.e., the query to be estimated is COUNT $(\sigma_F(r))$, where r is a relation with 10,000 tuples, and F is a selection formula. Note that, for a single select operation, the point space model and the urn model are exactly the same, i.e., each point or ball has only the value 0 or 1. We have chosen the selectivities to be 20% (i.e., 0.2) and 50% (i.e., 0.5) for the comparison that follows. In Table 4.1, we list the sample sizes (i.e., the number of input tuples) needed to obtain an estimate within an error of 10% (i.e., e = 0.1) with different confidence levels of 0.8, 0.9, 0.95 and 0.99.

First, let us consider the figures under the column "Reference sizes". For a population with the selectivity of 0.2 (i.e., variance $\sigma^2 = 0.2 \cdot 0.8 = 0.16$), the required sample size is $(\frac{1.28}{0.1})^2 \cdot \frac{0.16}{0.04} = 655$ for a 80% confidence level (t = 1.28).

Now let us consider LNS algorithm. For the select operation, b is equal to 1 in the stopping condition $y > k_1 \cdot b\ \frac{1}{e}\ (\frac{1}{e}+1)$. With the same error constraint, on the average, 1430 sample tuples are required for a population with selectivity 0.2. The sample size is obtained as follows. For a 0.8 confidence level, k_1 = 2.6 from the table in [LiNa 90b]; for a 10% error, e = 0.1; thus, y has to be larger than 2.6 · 10 · 11 = 286. Since the selectivity is 0.2, the expected sample size is 1/0.2 · 286 = 1430. The expected sample sizes are listed under the column "LNS sizes".

As for ILNS algorithm, as well as for double sampling, we have assumed that preliminary information is obtained from 100 sample tuples. On the average, from the first 100 sample tuples, $\hat{V}\ /\ \hat{Y}$ = 0.2 · 0.8 / 0.2 = 0.8 (< b =1 in LNS algorithm) for selectivity 0.2; and thus for a 10% error and a 80% confidence, the expected sample tuples in ILNS algotithm would be 0.8 · 1430 = 1144.

As for double sampling, with a sample size of 100 at the first step and selectivity 0.2, the average sample selectivity at the first step is 0.2. With a 10% (e = 0.1) error and a 80% (t = 1.28) confidence level, the overall sample size m is $(\frac{1.28}{0.1})^2 \cdot (\frac{0.16}{0.2^2})$ $(1 + 8(\frac{0.1}{1.28})^2 + \frac{0.16}{100 \cdot 0.2^2} + \frac{2}{100}) = 655 + 33 + 26 + 13 = 727$. Note that the first term in equation (4.6) is exactly the same as the equation (4.4), which is the required sample size for a situation where the variance and the mean of the population are known. The second, third terms can be considered as the extra cost for not knowing the population variance and mean. In the example, the extra cost is around 10% higher than the sample size from equation (4.6), and is much smaller than those for the LNS and ILNS algorithms.

4.4.2. Join Operation

Let us consider the join operation COUNT($r_1 \bowtie r_2$) Assume that both r_1 and r_2 have 10,000 tuples. For simplicity, we also assume that there is a single join attribute with a domain of integers between 1 and 10,000.

Selectivity = 0.2

confidence level	Reference sizes	LNS sizes	ILNS sizes	Our sizes
0.8	655	1,430	1,114	727
0.9	1,082	2,090	1,672	1,179
0.95	1,537	2,750	2,200	1,661
0.99	2,652	6,100	4,880	2,843

Selectivity = 0.5

confidence level	Reference sizes	LNS sizes	ILNS sizes	Our sizes
0.8	164	572	286	177
0.9	271	836	418	287
0.95	384	1,100	550	404
0.99	663	2,684	1,324	691

Error e=10%

Table 4.1. Sample Sizes Required for the Desired Error Bound For the Select Operation.

We have arbitrarily chosen to use normal distributions for the values of balls to compare the efficiency of different algorithms, since normal distributions have been claimed to be close to real life distributions. For the following comparisons, we assume that each tuple in r_1 has a distinct join attribute value between 1 and 10,000; and the join attribute values of r_2 have a normal distribution with mean 5,000, and variances 250^2 and $1{,}000^2$, respectively. Each tuple t in r_1 corresponds to a ball in the urn, and the ball has as its value the number of tuples in r_2 that have the same join attribute value as t; thus values of balls have a normal distribution. The total number of output tuples of $r_1 \bowtie r_2$ is 10,000. That is, the population mean (i.e., the average value of the balls) $\bar{Y}$ is 1. Based on 100 independently generated normal distributions for the join attribute values with variances 250^2 and $1{,}000^2$, the variance of the ball values are, on the average, 11.276 and 2.818, respectively; and the average b values are 27.84 and 11.76, respectively. Please note the difference between the variance of the join attributes values (i.e., 250^2 and $1{,}000^2$) and the variance of the ball values (i.e., 11.276 and 2.818). Corresponding to the above two distributions with ball value variances 11.276 and 2.818, there are about 1270 and 3700 balls (out of 10,000), respectively, having non-zero values, i.e., we do not deal with the cases where very few balls have non-zero values. Also, the former population has a sharp peak than the latter one.

Consider the normally distributed ball values with variance 11.276. For a given relative error of 10% ($e = 0.1$) and a confidence level 0.8 ($t = 1.28$), the expected sample sizes are obtained, for example, as follows. The "Reference sizes" is $(\frac{1.28}{0.1})^2 \cdot (\frac{11.276}{1^2}) = 1847$ from equation (4.4).

We assume that, in LNS algorithm, the lowest upper bound for the ball values, i.e., b, is known in the stopping condition $y > k_1 \; b \cdot \frac{1}{e} \cdot (\frac{1}{e} + 1)$. Note that this information is usually unknown and is not needed in the double sampling algorithm. When b is unknown, the sample size will be larger than the figures listed in Table 4.2 depending on how much larger the guessed b values are. With the same error constraint and the assumption that the b value is known, the y value has to be larger than $2.6 \cdot 27.84 \cdot 10 \cdot 11 = 7962$. Since $\bar{Y} = 1$, the expected sample size is $1 \cdot 7962 = 7962$.

As for ILNS algorithm, the average $\hat{V}(\hat{Y}) \;/\; \hat{Y}$ are 11.276 / 1 (compared to b = 27.84 in LNS) and 2.818 / 1 (compared to b = 11.76 in LNS) for the two normal distributions, respectively. Therefore, the expected sample sizes for a 10% error and a 80% confidence level for the same population is $7962 \cdot (11.276 \;/\; 27.84) = 3225$.

As for double sampling, the expected sample size, for the same error constraint, is $(\frac{1.28}{0.1})^2 \cdot \frac{11.276}{1^2} \cdot (1+8(\frac{0.1}{1.28})^2 + \frac{11.276}{100 \cdot 1^2} + \frac{2}{100}) = 2183$, which is much smaller than 7962 and 3225 required for the LNS and ILNS algorithms.

Variance = 11.276

Confidence levels	Reference sizes	LNS sizes	ILNS sizes	Our sizes
0.8	1,847	7,962	3,325	2,183
0.9	3,051	11,637	4,713	3,546
0.95	4,332	15,372	6,226	4,997
0.99	7,477	37,361	1,5132	8,559

Variance = 2.818

Confidence levels	Reference sizes	LNS sizes	ILNS sizes	Our sizes
0.8	462	3,363	806	507
0.9	763	4,916	1,178	822
0.95	1,083	6,468	1,550	1,157
0.99	1,869	15,782	3,769	1,981

Error e=10%

Table 4.2. Sample Sizes Required for the Desired Error Bound for the Join Operation.

4.4.3. Intersection Operation

Though the intersection operation is not discussed in [LiNa 90b], it can be easily modeled by the urn model or the point space model [HoOT 88, HoOz 88]. To model $r_1 \cap r_2$ in the urn model, each ball corresponds to a tuple t in a designated relation, say r_1. A ball has a value 1 if the corresponding tuple t is also in r_2; otherwise, the ball has a value 0.

Clearly, as far as the urn model is concerned, intersection is exactly the same as selection. Thus, Table 4.1 can be used as a source of comparison for intersection operations producing 2,000 and 5,000 output tuples.

5. Improvement Using Simple Random Sampling Without Replacement

In the discussions of Section 4, all the samples are taken with replacement (i.e., simple random sampling with replacement). From the tables, in many cases, the sample sizes can be very high, i.e., close to or even more than the population size. When the population size is not very large compared to the sample size, a better choice is to use simple random sampling without replacement. Simple random sampling without replacement may reduce the sample variance $\sigma_{\bar{y}}^2$ by a factor of $N-m$ / $N-1$ (which is called the *finite population correction*), and thus, may reduce the required sample size. When N is large, $N-1 \approx N$. Let us consider the equation (4.2). If simple random sampling without replacement is used, equation (4.2) should be written as

$$\sigma_{\bar{y}} = \sqrt{\frac{N-m}{N-1}}\frac{\sigma}{\sqrt{m}} \qquad (5.1)$$

To simplify the equation, let S^2 be $\sum_{i=1}^{i=N}(y_i - Y)^2$ / $N-1$. And then, the equation (4.4) will be

$$m = (\frac{tS}{eY})^2 \Big/ \left(1 + \frac{1}{N}(\frac{tS}{eY})^2\right) \qquad (5.2)$$

We now consider an example to see how simple random sampling without replacement can affect the efficiency. For a 10% error and a 99% confidence level, the required "Reference size" for simple random sampling without replacement is 7,477 / $(1 + \frac{7{,}477}{10{,}000}) = 4278$, which is much smaller than 7,477 from Table 4.2. Similarly, one can use the simple random sampling without replacement as the underlying sampling method for double sampling, and rewrite the equation (4.6) by first replacing the v^2 by $(N-m)\ s^2\ /\ N$, where s^2 is defined as $\sum_{i=1}^{i=m}{}^1(y_i - \bar{y})^2$ / $(m-1)$, and then solving for m. As for equation (4.7), which is the formula for a population with values only 0 and 1 (e.g., the point space model), one can replace the term $(1-p_1)$ (more accurately, the term $p_1\ (1-p_1)$) by $((N-m)\ /\ N)\ (1-p_1)$, where N is the population size (i.e., the number of points in the point space) and m is the overall sample size. To simplify the equation for the sample size m, let the first, second and third terms of equation (4.7) be x_1, x_2 and x_3, respectively. By solving the following quadratic equation, m can be obtained.

$$(N+x_1)m^2 - (N^2 + 2Nx_1 + Nx_3)m + (x_1 + x_2 + x_3)N^2 = 0 \qquad (5.3)$$

where N is the population size, i.e., the number of points in the point space. Equation (5.3) gives a tighter bound on the sample size than equation (4.6), and is implemented in CASE-DB, the prototype DBMS.

6. Experimental Results

Double sampling and adaptive sampling are incorporated into two database management systems, called CASE-DB [HoOT 89] and CASE-MDB [Liu 89], for error-constrained COUNT query estimation purposes. CASE-MDB is a main memory version of CASE-DB which is a disk-resident, real-time (or time-constrained) prototype database management system. Double sampling implemented in CASE-DB and CASE-MDB uses simple random sampling without replacement (SRS) and systematic sampling (SS) (as well as other sampling techniques such as cluster sampling) to draw sample tuples. Due to space limitations, in this paper, we present the experimental results for double sampling on single select, join and intersection operations in CASE-MDB. Estimators for projection operations and experimental results for arbitrary relational algebra expressions can be found in [Dogd 90].

6.1. Design of Experiments

Instead of using the number of tuples as a measure of the sample size, we choose to use the *sampling fraction f* as the measure, which is defined to be the ratio of the number of tuples to the total number of tuples in a relation. Furthermore, for the double sampling approach, equal sampling fractions are drawn from all the input relations, not just from r_1. Recall from Sections 2 and 4.3 that, in the point space model, sample tuples are drawn from (some or) all of the operand relations and no index files on join attributes are needed.

Experiments are performed on relations with 10,000 tuples each, and each data value in the tables of this section is obtained from 200 experiments. The experiments have been designed to show, for a given confidence level (here always 95%) and different relative error limits (e.g., 2%, 5%, 10%, etc.), the actual errors, confidence level and the required sample sizes for different sampling techniques. In the tables, the column f% gives the average sampling fraction from each input relation. The column e% gives the average of actual errors observed during 200 runs, which is to be compared with the specified error limit. And the column l% gives the ratio of the number of runs in which the relative errors are found to be less than or equal to the specified error limit e, which is to be compared with the specified confidence level l (in this section, always 95%).

6.2. Creation of Input Relations

Each input relation has two attributes, both with integer values. The first attribute contains unique random numbers to differentiate one tuple from another. The second attribute is the one of interest in the query, i.e., the attribute involved in the selection formula or the join attribute. To observe the effect of the distributions of attribute values, we have experimented with relations in which second attributes have either a uniform or a normal distribution. To observe the behavior of the double sampling using different sampling techniques, we have also run experiments with relations in which tuples are ordered with respect to the second attribute. Hereafter, we use *ordered* and *unordered* relations to differentiate relations with ordered and unordered tuples, respectively.

6.3. Experiments on a Single Selection and Join Operations

6.3.1. Selection Operation

The experiments are performed on a selection operator with selectivities 0.2 and 0.5. The sampling fraction used in the first step is always 2% of the total relation (i.e., f = 2% or 200 tuples here). As far as the COUNT estimator is concerned, either a tuple satisfies the selection formula or it does not; the actual attribute values are not important. Therefore, we do not discuss the distributions of attribute values for the selection operation since they will not affect the performance [HoOT 88]. For simple random sampling (SRS), the storage ordering of tuples in a relation has no effect on the performance because every tuple has an equal chance to be selected into a sample, and every combination of m tuples has an equal chance to be selected as a sample. Thus, by definition, any sample of size m randomly taken is a simple random sample, regardless of the storage ordering. However, the storage ordering has a significant influence on systematic sampling. In systematic sampling, not every combination of m tuples can be drawn. When a sample is a good representative of the whole population, the systematic sampling gives a precise estimate; otherwise, it gives a bad estimate. This phenomenon is observable in tables 6.1 and 6.2.

The f% columns give the average of sampling fractions for different relative errors (e.g., 2%, 5%). For simple random sampling without replacement, these values are quite close to the expected values computed from equation (5.3). For example, for a 10% relative error and a given 95% confidence level, the experimental average size is 14% for selectivity 0.2 from Table 6.1, and the computed value is also 14%. Also, the confidence levels (i.e., l% columns) are quite close to the specified confidence level 95%. This means that double sampling on top of simple random sampling without replacement behaves exactly as the formula (5.3) specifies.

Given Error	SRS unordered/ordered			Adaptive Sampling unordered/ordered		
	f%	e%	l%	f%	e%	l%
2%	80	1	97	100	0	100
5%	39	2	95	100	0	100
10%	14	4	96	28	3	100
20%	4	8	97	8	5	98

Given Error	Systematic Sampling unordered			Systematic Sampling ordered		
	f%	e%	l%	f%	e%	l%
2%	50	2	0	50	2	0
5%	38	4	83	38	49	6
10%	14	4	91	14	10	51
20%	4	8	98	4	8	100

Selectivity = 0.2

Table 6.1. Select Operation with Selectivity of 0.2.

Given Error	SRS unordered/ordered			Adaptive Sampling unordered/ordered		
	f%	e%	l%	f%	e%	l%
2%	49	1	96	100	0	100
5%	14	2	94	42	1	100
10%	4	4	97	11	2	100
20%	2	5	100	3	5	100

Given Error	Systematic Sampling unordered			Systematic Sampling ordered		
	f%	e%	l%	f%	e%	l%
2%	48	1	94	49	1	100
5%	13	2	98	13	3	80
10%	4	4	97	4	1	100
20%	2	6	100	2	0	100

Selectivity = 0.5;

Table 6.2. Select Operation with Selectivity of 0.5.

For systematic sampling (SS), only a limited number of possible samples can be drawn. For example, if the sampling fraction is 10%, there are only 10 possible samples can be drawn. A typical sample may consist of the k^{th} ,$k+10^{th}$, $k+20^{th}$, , tuples, where k has only 10 possible values, $1 \le k \le 10$. Therefore, the precision of an estimate is determined by a limited number of underlying samples. We have chosen to use a cut on the sampling fraction at 50% when the sampling fraction is larger than 50% for systematic sampling. When the relation is unordered, tuples are (more or less) randomly distributed. Therefore, in general, the results for systematic sampling should be similar to simple random sampling. Of course, whether this similarity occurs or not depends on how the underlying samples are constructed. This explains why in the case of systematic sampling there are cases with 1% values much lower than 95% and others with 1% values quite close to those observed in simple random sampling.

When the tuples are ordered, it is possible that at a certain sampling fraction, the underlying samples have exactly (or almost) the same selectivity as that of the entire relation while for some other sampling fractions, the underlying samples are not good representatives of the population. This explains why in tables 6.1 and 6.2 there are cases where the estimates are very precise while in others the estimates are very bad.

We now compare f% columns for double sampling (using simple random sampling) and adaptive sampling. Clearly, sample sizes for double sampling is much smaller than adaptive sampling, for the same given error constraint. Note that adaptive sampling has a higher confidence level, which is higher than the required 95% and considered unnecessary. This is due to unnecessarily large samples obtained from adaptive sampling.

6.3.2. Join Operation

Experiments are performed on a join operation producing 72,750 output tuples, i.e., selectivity $= 72{,}750/10{,}000^2 \approx 0.00073$ with operand relations having 10,000 tuples each. We assume that the first step sampling fraction is 5% from each relation, and there are no index files available. With 500 tuples drawn from each operand relation, one can construct 500 independent sample points, or 500^2 correlated sample points in the point space [HoOT 88, HoOz 88], in which each point corresponds to one pair of tuples from different operand relations. Clearly, with the same effort, the latter tremendously cuts down the cost of sampling, and thus is adopted in CASE-DB and CASE-MDB. Note that even though the sample tuples are drawn randomly, the sample points are no longer independent. The experimental results in [HoOz 88] have shown that the introduced correlation does not have a severe effect on the performance of the estimator. Also note that a 5% sample fraction from each relation corresponds to a sample of 0.25% (i.e., 0.05^2) of the point space.

Consider the point space of a join operation $r_1 \bowtie r_2$. The number of points with value 1 on the same row (or the same column) represents the number of tuples produced by a tuple in, say, r_1 and the entire relation r_2 in the join operation. If the join attribute values are not uniformly distributed, each row/column will have different number of points with value 1. Since sample points are no longer independently drawn, the distributions of join attribute values, and thus the distributions of points with value 1, will have an effect on the precision of estimates. This explains why a better performance is obtained from relations with join attribute values uniformly distributed than from normally distributed, since the points with value 1 is more evenly distributed over the point space when the join attribute values have a uniform distribution.

Since the sample points are correlated, larger variance in estimation may be obtained compared to a sample with independent points of the same size. However, the effects are not severe in our empirical evaluations.

As for systematic sampling, when the relations are not ordered, the results are close to those with simple random sampling, as explained in the selection operation. When the relations are ordered, there are some estimates that are very high and some that are very low. Systematic sampling applied to ordered tuples in join operations tends to give a large variance in estimates. As we explained earlier, this phenomena is due to having limited number of underlying samples for systematic sampling.

Given Error	SRS			SS					
	unordered/ordered			unordered			ordered		
	f%	e%	1%	f%	e%	1%	f%	e%	1%
2%	39	1	99	39	1	100	39	16	12
5%	19	1	99	19	1	98	19	5	53
10%	10	3	100	10	2	100	10	3	99
20%	5	6	100	5	6	100	5	6	100

Selectivity = 0.00073

Table 6.3. Join Operation with Join Attribute Having Uniform Distribution.

Given Error	SRS			SS					
	unordered/ordered			unordered			ordered		
	f%	e%	1%	f%	e%	1%	f%	e%	1%
2%	38.7	1	92	38.7	1	99	38.8	29	0
5%	18.5	2	92	18.5	2	97	18.5	8	5
10%	10.3	4	97	10.3	3	98	10.3	7	91
20%	5.0	8	95	5 0	6	100	5.0	3	100

Selectivity = 0.00073

Table 6.4. Join Operation with Join Attribute Having Normal Distribution.

6.3.3. Intersection Operation

Intersection operation can be considered as a special case of the join operation in which all the attributes (or the key attributes) are "join" attributes, with probably a low selectivity. In order to guarantee that we have some points with the value 1, we take a higher first step sampling fraction (10%) than in the join operation. Each row or column has at most one point with value 1. Here, we do not have to discuss different distributions of "join" attribute values, since each "join" attribute value is unique in a relation. Systematic sampling has a poor performance especially when tuples are ordered.

Given	SRS			SS					
Error	unordered/ordered			unordered			ordered		
	f%	e%	l%	f%	e%	l%	f%	e%	l%
2%	100	0	100	50	2	40	39	118	0
5%	75	1	99	50	2	100	19	159	0
10%	50	3	98	48	4	100	11	108	0
20%	30	5	99	30	7	85	10	188	0

First Stage Sampling Fraction = 10%

Table 6.5. Intersection Operation with 2,000 Output Tuples.

Given	SRS			SS					
Error	unordered/ordered			unordered			ordered		
	f%	e%	l%	f%	e%	l%	f%	e%	l%
2%	91	0	100	50	2	48	39	118	0
5%	57	1	100	50	2	100	18	145	0
10%	35	3	100	35	16	38	10	183	0
20%	20	5	100	20	9	94	10	183	0

First Stage Sampling Fraction = 10%

Table 6.6. Intersection Operation with 5,000 Output tuples.

7. Conclusions

We have presented a double sampling-based COUNT(E) query estimation approach that guarantees a desired error bound with a specified confidence level. The estimatos is unbiased and consistent, and performs very well with simple random sampling without replacement as the underlying sampling plan. We have also presented the experimental results we performed on our prototype DBMS called CASE-DB.

We have also implemented and compared another error-constrained COUNT(E) estimation approach, the (I)LNS algorithms based on adaptive sampling. Given an error constraint and a confidence level, the LNS approaches determine whether or not an additional sample unit is to be taken by observing the number of sample output tuples. We have shown that these approaches perform, at least under the assumptions we have tested, worse than the double sampling we propose. Also, there are some problems that remain to be solved with the adaptive sampling approach, e.g., the bias of the estimator for the size of the Select-Join(-Intersection) query, which is denoted by $n \cdot s/m$ in [LiNa 89, LiNa 90, LNS 90]. Note that the estimator $n \cdot s/m$ is basically the same as the one, denoted by Y, in [HoOT 88, HoOT 89]. The estimator is unbiased when the sample size is prefixed; however, it may become biased in an error-constrained environment [Cox 52, Coch 77]. The magnitude of the bias in the estimator, which depends on how the sample units are taken (e.g., checking the number of output tuples and then deciding whether an additional sample unit is to be taken - this is actually how the bias is introduced), needs to be worked out. More importantly, if the magnitude of the bias is unknown, the confidence levels will also be inaccurate.

8. References

[Coch 77] W. Cochran, "Sampling Techniques", Third Ed. John Wiley & Sons, 1977.

[Cox 52] D.R. Cox, "Estimation by double sampling", Biometrika, 39.

[Dogd 90] E. Dogdu, "Experimental results on Double sampling", Unpublished Manuscript, CWRU, 1990.

[HoOT 88] W-C. Hou, G. Ozsoyoglu and B. Taneja, "Statistical Estimator for Relational Algebra Expressions", ACM PODS, Austin, TX, 1988.

[HoOT 89] W-C. Hou, G.Ozsoyoglu and B. Taneja, "Processing Aggregate Relational Queries with Hard Time Constraints", ACM SIGMOD, Portland, OR, 1989.

[HoOz 88] W-C. Hou and G. Ozsoyoglu, "Statistical Estimator for Aggregate Relational Algebra Expressions", to appear in ACM TODS, submitted 1988.

[LiNa 89] R. Lipton and J. Naughton, "Estimating The Size of Generalized Transitive Closures", the 15th VLDB Conference, Amsterdam, The Netherlands, 1989.

[LiNa 90] R. Lipton and J. Naughton, "Query Size Estimation by Adaptive Sampling", ACM PODS, 1990.

[LNS 90] R. Lipton, J. Naughton and D. Schneider, "Practical Selectivity Estimation through Adaptive Sampling", ACM SIGMOD, 1990.

[Liu 89] Liu, Y-M., "A Main Memory Real-Time Database Management System--Implementation and Experiments", MS Thesis, CWRU, July 1989.

[Sukh 84] P. Sukhatme, etc.., "Sampling Theory of Surveys Application", Third Ed., New Delhi, India and Iowa State Univ. Press, Ames, Iowa, 1984.

[Wald 45] A. Wald, "Sequential Tests of Statistical Hypotheses", Ann. Math. Stat. Vol 16, 1945.

[Yama 67] T. Yamane, "Elementary Sampling Theory", Prentice-Hall, Inc , 1967.