Statistical Estimators for Relational Algebra Expressions†

*Wen-Chi Hou, Gultekin Ozsoyoglu, and Baldeo K Taneja**

Department of Computer Engineering and Science
and
Center for Automation and Intelligent Systems
Case Western Reserve University
Cleveland, Ohio 44106

ABSTRACT

Present database systems process all the data related to a query before giving out responses As a result, the size of the data to be processed becomes excessive for realtime/time-constrained environments A new methodology is needed to cut down systematically the time to process the data involved in processing the query. To this end, we propose to use data samples and construct an approximate synthetic response to a given query

In this paper, we consider only COUNT(E) type queries, where E is an arbitrary relational algebra expression We make no assumptions about the distribution of attribute values and ordering of tuples in the input relations, and propose consistent and unbiased estimators for arbitrary COUNT(E) type queries We design a sampling plan based on the cluster sampling method to improve the utilization of sampled data and to reduce the cost of sampling We also evaluate the performance of the proposed estimators

1. Introduction

The existing database technology is not directly applicable to realtime or time-constrained data processing environments For example, presently, database management systems are only used offline in manufacturing where manufacturing processes are controlled and manipulated in realtime (using programmable controllers+) or in a time-constrained manner Basic reasons for this phenomenon are

- Present state-of-the-art database systems cannot be easily fed realtime data because of their monolithic nature or their restricted way of interaction modes,
- Present database system software is very large with variety of components for concurrency control, optimization, etc , and is slow,
- Heavy utilization of secondary storage makes the existing database systems slow for realtime/time-constrained data manipulation environments

For the last problem, recently *main-memory databases* have been proposed Although main memory databases are very useful in cutting down time-costs of queries, they represent only a speedup and do not offer any solutions to the basic problem. Even with no secondary storage, the data involved with a given query may be so large that there may not be enough time to completely process the query Thus we list an additional problem.

- Present database systems process all the data related to a query before giving out responses As a result, the size of the data to be processed becomes excessive for realtime/time-constrained environments

We think that a new methodology is needed to cut down systematically the time for processing the data involved with the given query

† This research is supported by the National Science Foundation under Grant DCR-8605554

* Department of Mathematics and Statistics, Case Western Reserve University

+ Programmable controllers are special-purpose computers that control manufacturing processes using the present and the past history of some "input" manufacturing processes The data storage and manipulation features of the existing programmable controllers do not use any of the existing database technology

Our approach is as follows If a response to a query in a realtime/time-constrained environment is not possible in the given time units, one can use data samples and construct an approximate *synthetic response* to the requested query The theme of such synthetic query response construction can be rephrased as *statistical approximation of query responses* In principle, the actual query response may be an aggregate (e g , SUM, COUNT) or it may simply be a set of values (tuples) Clearly, any synthetic aggregate query response should also be accompanied by a probabilistic error analysis describing the precision (i e , the accuracy) of the approximation, the confidence intervals for synthetic (i e , approximate) values, etc In this paper, we

(a) find consistent and unbiased statistical estimators (based on simple random sampling and cluster sampling) for COUNT(E) where E is any arbitrary relational algebra (RA) expression, and
(b) evaluate the performance of these estimators by incorporating them into a prototype relational DBMS and running them using artificially created relation instances

An extended version of this paper and the proofs of lemmas and theorems in this paper can be found in [HoOT 87]

In the literature, Morgenstein [Morg 81] discusses the estimation of COUNT queries by taking samples from "randomized" relations, in which tuples are randomly stored Estimation for a limited class of COUNT queries that contain only one selection and one join operation are given

Papers [Olke 86, OlkR 86] are closely related to this work even though they do not discuss the estimation of aggregate queries Olken [Olke 86] describes several sampling methods for generating simple random samples from flat files Olken and Rotem [OlkR 86] further discuss how to obtain a simple random sample of the result of each relational algebra operator using its operand relations Sampling algorithms and their performance, in terms of the number of disk accesses, are discussed [OlkR 86] introduces acceptance/rejection sampling to adjust the effect of relational operators on inclusion probabilities of resulting tuples This adjustment needs the knowledge of the cardinality of the projected or join attribute values One disvantage of this approach is that, for queries that allow projection and join on multiple domains and that contain multiple projections and joins, it can be quite complicated (if possible at all) to determine the cardinality of the projected and join attribute values In comparison, our research discusses how to use samples, not necessarily simple random samples, to estimate COUNT queries That is, we investigate the relationship between the sample results and the complete query results, find estimators and discuss their properties Also, we design an efficient sampling plan, which is based on the cluster sampling, to reduce the cost of sampling and improve the utilization of sampled data. We deal with arbitrary relational algebra expressions without assuming any knowledge about the distribution of attribute values

Rowe [Rowe 85] adopts a different approach, called antisampling, by which estimation of a population characteristic is inferred from some prestored statistics on the supersets of the population concerned, not from the database itself It requires the construction of an auxiliary structure, called the database abstract, to collect a large set of relevant statistical information on the database The main advantage of this approach is its fast speed of calculation However, the precision of estimation really depends on how closely the statistics collected are related to the database Since there are infinite number of possible statistics on a finite database, even with strong complexity limits on queries, the combinatorial possibilities can be immense Only a small subset of those statistics may be incorporated into the database abstract. This may limit the types of queries that a user can ask Also, in order to reflect the changes in a database, an additional effort is needed to update the statistics in the database abstract. The details of this approach still need to be worked out.

In section 2, we briefly introduce some statistics terminology and definitions Given an arbitrary relational algebra expression E, section 3 proposes estimators for a COUNT(E) query based on the simple random sampling scheme This section also discusses the theoretical framework for sampling from a relational algebra expression In section 4, we introduce another sampling technique, called cluster sampling, to reduce the cost of sampling Section 5 reports the preliminary experimental results of the estimators proposed in section 4

2. Terminology and Definitions

Let ψ be a given parameter of interest, such as a population mean or a population total An estimate of the parameter ψ is a single number computed from the sample information, which serves as a guess for the value of ψ

The function of the observation, which is used to obtain an estimate is called an *estimator*, and is denoted by $\hat{\psi}$

We now briefly discuss the desirable properties of an estimator [Coch 77]. An estimator $\hat{\psi}$ is said to be an *unbiased estimator* of ψ if $E(\hat{\psi})=\psi$ for all possible values of ψ, where $E(\hat{\psi})$ represents the expected value of the $\hat{\psi}$ If $\hat{\psi}$ is not unbiased, the difference $E(\hat{\psi})-\psi$ is called the *bias* of ψ Another highly desirable property is the consistency An estimator is *consistent* if the probability that it is in error by more than any given amount approaches zero as the sample becomes large That is, for a consistent estimator, when the sample consists of the whole population, the estimate becomes exactly equal to the true population value The *Mean Square Error* (MSE) of an estimator $\hat{\psi}$ is defined to be $E(\hat{\psi}-\psi)^2=Var(\hat{\psi})+(bias)^2$, where Var denotes the variance The MSE is usually used as a criterion indicating how precise the estimator $\hat{\psi}$ is Clearly, a smaller MSE is always desirable

Since almost any parameter that we wish to estimate has as its set of possible values an entire interval of numbers, reporting only the estimated value is generally unsatisfactory One measure of describing how close a point estimate is to the true value is to specify a *confidence interval* associated with a certain *confidence level* [Coch 77] A confidence interval is an interval of plausible values for the parameter being estimated A confidence level is the degree of plausibility of such an interval.

Simple random sampling is a method of selecting m elements (sample size) out of N (the population size) such that each one of $\frac{N!}{m!(N-m)!}$ possible samples has an equal chance of being selected Since a unit (i e, an element) that is already selected is removed from the population for all subsequent draws, this method is also called the *random sampling without replacement*

3. Estimation of COUNT(E) Based on the Simple Random Sampling Technique

In this section, we develop a general methodology for estimating the aggregate query COUNT(E) by using simple random sampling, where E is an arbitrary relational algebra expression containing operators union ($\cup$), difference ($-$), intersection ($\cap$), selection (σ), projection (π) and natural join ($\bowtie$) We assume no knowledge of the distribution of attribute values, ordering of tuples (sorted or not), and the existence of index files

In the present practice of survey sampling, a sample is usually taken from an existing population However, in our context, the population, i e, the result of a relational algebra expression E, is not available since we assume that there is not enough time to completely evaluate the expression E Therefore, we develop a methodology to model the mapping between a resulting tuple of E and the set of operand tuples that generate it such that an estimate of COUNT(E) can be computed by taking samples from the operand relations of E and then applying appropriate estimators to the sample result (instead of first completely evaluating E and then taking a sample from it)

Intuitively, we may think that an unbiased estimate of COUNT(E) can be directly computed from the sample result obtained by substituting the samples of relations for the relations themselves and evaluating E Unfortunately, the relationship between the sample result and the complete query result is not that straightforward This is also pointed out in [OlkR 86] For example,

$S(r_1-r_2) \neq S(r_1) - S(r_2)$, $S(r_1 \cup r_2) \neq S(r_1) \cup S(r_2)$, and $S(\pi(r_1)) \neq \pi(S(r_1))$,

where $S(r_i)$ denotes a simple random sample of the relation r_i However, as will be seen in sections 3 1 and 3 2, selection, intersection, and natural join have some properties such that we can directly model and compute the COUNT of expressions involving only these operations Two approaches, direct and indirect, are proposed for estimating the COUNT of expressions involving union and difference operations The direct approach involves evaluating some operand relations completely (not just samples of them) For the other approach, only samples of operand relations are needed Projection exhibits different characteristics from the others since it may produce duplicate tuples However, Goodman's estimator [Good 49], to be discussed in section 3 4, can be used as an estimator for the COUNT of expressions involving projection operations

This section proposes an estimator $\hat{Y}$ for those expressions that contain only operators σ, $\cap$ and $\bowtie$, and another estimator, i e, Goodman's estimator, for those expressions that contain operators σ, $\cap$, $\bowtie$, $-$ and at least one π's Based on the above two estimators, we develop a general methodology for estimating COUNT(E), where E is an arbitrary relational algebra expression The methodology is summarized in the algorithm ESTIMATE-COUNT(E) and the details are discussed in sections 3 1 to 3 4

We start with an algebra expression E that contains the natural join, intersection, and selection, and then incorporate union, difference and projection into our approach A consistent and unbiased estimator for COUNT(E) is constructed at the same time

Algorithm ESTIMATE-COUNT(E)

* Input an arbitrary relational algebra expression E
* Output. an estimate of COUNT(E)

Begin

(1) transform each subexpression of E, say E_i, that begins with a π and contains operator $\cup$'s into the form $\pi(E_{i1}) \cup \pi(E_{i2}) \cup \pi(E_{im})$ (Lemma 3) such that E_{ik}, $1 \le k \le m$, does not contain any $\cup$'s Consider each such $\pi(E_{ik})$ as a whole, i e , a relation, in the following steps

(2) transform E into the form $E_1\ \theta_1\ E_2\ \theta_2 \cdots \theta_{n-1}\ E_n$ such that $\theta_i \in \{\cup, -\}$, $1 \le i \le n$, E_i does not contain any $\cup$ or $-$ (described in Lemma 1) Note that parentheses are allowed in the expression.

(3) compute COUNT(E) as $\sum_j (\pm) COUNT(E'_j)$ by applying the principle of inclusion and exclusion (Lemma 2) For each COUNT(E'_j), choose an appropriate estimator $\hat{C}_j$ depending on whether E'_j has π or not, Return $\sum_j (\pm)\hat{C}_j$

End.

Example 1. Estimate COUNT(E)
$=$COUNT($r_1 \bowtie (r_2 - \pi(r_3 \cup r_4 - r_5))$)

Step (1) of ESTIMATE-COUNT produces

$$r_1 \bowtie (r_2 - \pi(r_3 \cup r_4 - r_5))$$
$$= r_1 \bowtie (r_2 - ((\pi(r_3 - r_5)) \cup (\pi(r_4 - r_5))))$$

Let r_3^* and r_4^* denote $\pi(r_3 - r_5)$ and $\pi(r_4 - r_5)$, respectively Step (2) of ESTIMATE-COUNT produces

$$= (r_1 \bowtie r_2) - ((r_1 \bowtie r_3^*) \cup (r_1 \bowtie r_4^*))$$

For simplicity, let N denote COUNT The step (3) of ESTIMATE-COUNT rewrites the COUNT(E),i e , N(E), as

$$= N(r_1 \bowtie r_2) - N((r_1 \bowtie r_2) \cap ((r_1 \bowtie r_3^*) \cup (r_1 \bowtie r_4^*)))$$
$$= N(r_1 \bowtie r_2) - N(r_1 \bowtie (r_2 \cap r_3^*)) - N(r_1 \bowtie (r_2 \cap r_4^*)) + N(r_1 \bowtie (r_2 \cap r_3^* \cap r_4^*))$$

Let $\hat{N}(E_j)$ denote the estimator for $N(E_j)$ Then, the algorithm returns

$$= \hat{N}(r_1 \bowtie r_2) - \hat{N}(r_1 \bowtie (r_2 \cap r_3^*)) - \hat{N}(r_1 \bowtie (r_2 \cap r_4^*)) + \hat{N}(r_1 \bowtie (r_2 \cap r_3^* \cap r_4^*))$$

3.1. Incorporating Natural Join and Intersection

A relation instance r_i with k tuples can be mapped to a set of k points in a one-dimensional space d_i with each tuple t corresponding to a point $f_i(t)$ on d_i, where f_i is a one-to-one function mapping a tuple in r_i to a point in d_i Let $d_1 d_2 \cdots d_n$ denote an n-dimensional space with $d_1, d_2, \ldots, d_n$ for each dimension, and $r_1 r_2 \cdots r_n$ denote a finite n-dimensional subspace of $d_1 d_2 \cdots d_n$ in which each dimension d_i contains only those points corresponding to tuples in relation instance r_i The natural join (and also the intersection) operation on two relations r_1 and r_2 can be viewed as a process of assigning binary values, 0 and 1, to the points in the plane $r_1 r_2$ A point $p(f_1(t_1), f_2(t_2))$ assumes value 1 if and only if t_1 in r_1 and t_2 in r_2 have identical join (intersection) attribute values Otherwise, $p(f_1(t_1), f_2(t_2))$ is 0 Similarly, an expression E with n−1 $\bowtie$'s (or $\cap$'s) and n operand relations $r_1, r_2, \ldots, r_n$, can be modeled as an n-dimensional space $r_1 r_2 \cdots r_n$ (hereafter, called *the point space* of E) with points having values of 0 and 1 There is a one-to-one correspondence between the points of $r_1 r_2 \cdots r_n$ having value 1 and the resulting tuples Therefore, evaluating the aggregate query COUNT(E) is exactly the same as counting the number of 1's in the corresponding point space Actually, the intersection can be viewed as a special case of the natural join in which all the attributes are the join attributes Thus, an estimator for COUNT(E) where E consists of the natural join and/or intersection of n relations can be given as follows

Let E be an arbitrary relational algebra expression with only $\bowtie$'s, and $\cap$'s, and n relations $r_1, \ldots, r_n$ N_i denotes the number of tuples in r_i, and $N = N_1 \times N_2 \times \cdots \times N_n$ is the total number of points in the point space $r_1 r_2 \cdots r_n$ Assume that the points in the point space of E are numbered as $p_1, p_2, \ldots, p_N$ Let y_i be the value, 0 or 1, of a point p_i in the point space of E, and $Y(E) = y_1 + y_2 + \cdots + y_N$, be the total number of 1's in the point space Note that Y(E) = COUNT(E) because of the one-to-one correspondence between the points with value 1's and the resulting tuples An estimator $\hat{Y}(E)$ for Y(E) based on a simple random sample of m points from the space $r_1 r_2 \cdots r_n$ is then

$$\hat{Y}(E) = N \frac{\sum_{i=1}^{i=m} y_i}{m} \quad \text{where } N = N_1 \times N_2 \times \cdots \times N_n$$

Theorem 1 $\hat{Y}(E)$ is a consistent and unbiased estimate of COUNT(E)

Theorem 2 The variance of $\hat{Y}(E)$, denoted by Var($\hat{Y}$(E)), is

$$\mathrm{Var}(\hat{Y}(E)) = N^2 \frac{(1-f)}{m} \frac{\sum_{i=1}^{i=N}(y_i-\bar{Y})^2}{N-1}$$

where $f=\frac{m}{N}$ is the sample fraction, and $\bar{Y}=\frac{\sum_{i=1}^{i=N} y_i}{N}$

Theorem 3 An unbiased estimate of Var($\hat{Y}$(E)), denoted by $\hat{Var}(\hat{Y}(E))$, can be computed as

$$\hat{Var}(\hat{Y}(E)) = N^2 \frac{(1-f)}{m} \frac{\sum_{i=1}^{i=m}(y_i-\bar{y})^2}{m-1}$$

where $f=\frac{m}{N}$ is the sample fraction, and $\bar{y}=\frac{\sum_{i=1}^{i=m} y_i}{m}$

3.2. Incorporating Selection

Selection can be incorporated into our approach by taking into account the qualifications, specified by the selection formula of σ, during the 0/1 value assigning process As before, the relational algebra expression E that contains operators ⋈, ∩, σ and n operand relations $r_1, r_2, \ldots, r_n$ is modeled as an n-dimensional space $r_1 r_2 \ldots r_n$. A point $p(f_1(t_1), \ldots, f_n(t_n))$ assumes value 1 if and only if the set of operand tuples $t_1, \ldots, t_n$ produces a resulting tuple in $E(t_1, \ldots, t_n)$, where $E(t_1, \ldots, t_n)$ represents the expression E in which r_i, $1 \le i \le n$, contains only one tuple, namely, t_i Again, there is a one-to-one correspondence between the points p and the resulting tuples Clearly, Theorems 1, 2 and 3 hold for any relational algebra expression E that contains ⋈'s ∩'s and σ's

3.3. Incorporating Union and Difference

Consider the points p in the n-dimensional space corresponding to an expression E that contains operators σ, ⋈, ∩, and n operand relations Assume that the tuple t_i is in r_i, $1 \le i \le n$ The value of a point $p(f_1(t_1), f_2(t_2), \ldots, f_n(t_n))$ is completely determined by the set of tuples $\{t_i\}$ depending on whether $E(t_1, \ldots, t_n)$ can produce a resulting tuple or not. However, for an expression that contains ∪'s or −'s, the set of tuples $\{t_i\}$ alone may not be sufficient to determine if they can generate a resulting tuple in E or not. For example, consider E = $r_1 - r_2$, or E = $r_1 \cup r_2$, $t_1 \in r_1$ and $t_2 \in r_2$ Unless we check all the tuples in r_2, we can not determine if t_1 is in r_2 This suggests the approach that, instead of taking a sample from the space $r_1 r_2$, we take a sample from r_1 and check it with the whole relation r_2 Obviously, such an approach is expensive We will discuss this approach later in the project operation of the relational algebra We now discuss another approach

Let N denote COUNT The Principle of Inclusion and Exclusion [Liu 68] states that $N(r_1 \cup r_2) = N(r_1) + N(r_2) - N(r_1 \cap r_2)$ and $N(r_1 - r_2) = N(r_1) - N(r_1 \cap r_2)$ More generally, $N(E_1 \cup E_2) = N(E_1) + N(E_2) - N(E_1 \cap E_2)$ and $N(E_1 - E_2) = N(E_1) - N(E_1 \cap E_2)$, where E_i is any relational algebra expression This implies that we can compute $\mathrm{COUNT}(E_i \cup E_j)$ and $\mathrm{COUNT}(E_i - E_j)$ through $\mathrm{COUNT}(E_i)$, $\mathrm{COUNT}(E_j)$ and $\mathrm{COUNT}(E_i \cap E_j)$, indirectly

Let E be an expression with ∪'s and −'s As a first step to incorporate union and intersection into our approach, we decompose E (Lemma 1 below) into a set of subexpressions E_i's combined by ∪'s and −'s, where E_i's themselves do not contain ∪'s or −'s That is,

$$E = E_1\ \theta_1\ E_2\ \theta_2 \quad \theta_k\ E_k \qquad (1)$$

where θ_i, $1 \le k$, denotes ∪ or − Note that, for simplicity, we have omitted parentheses for specifying the precedence among θ_i's in the above expression By Lemma 2 below, which is based on the application of the principle of inclusion and exclusion and the associativity properties of relational algebra operators, COUNT(E) is computed as $\sum_j (\pm)\ \mathrm{COUNT}(E'_j)$, where E'_j does not contain ∪'s and −'s Note that E'_j is either E_i in (1) above or intersections of E_i's, which is due to the application of the principle of inclusion and exclusion

Lemma 1 Let E be a relational algebra expression containing operators ∪, −, ∩, ⋈, σ and π, except that projections do not operate on differences E can be rewritten as a set of subexpressions E_i containing only operators ∩, ⋈, σ and π, and combined by operators ∪ and −

Expressions that contain projections operating on difference are excluded from the above theorem because

$$\pi(E_1 - E_2) \neq \pi(E_1) - \pi(E_2)$$

Lemma 2 COUNT(E), where E is defined as in Lemma 1, can always be computed as

$$\mathrm{COUNT}(E) = \sum_j (\pm)\ \mathrm{COUNT}(E'_j)$$

where E'_j contains only operators ∩, ⋈, σ, and π

Example 2 Consider COUNT($(E_1 \cup E_2) \bowtie E_3$)
=N($(E_1 \cup E_2) \bowtie E_3$) where N denotes COUNT
=N($(E_1 \bowtie E_3) \cup (E_2 \bowtie E_3)$)
=N($E_1 \bowtie E_3$) + $N(E_2 \bowtie E_3)$ − N($(E_1 \cap E_2) \bowtie E_3$)

For an expression E that does not contain projections, from section 3 2, we have $\hat{Y}(E'_j)$ as an unbiased estimate of COUNT(E'_j) where E'_j is a subexpression or intersection of subexpressions decomposed from E, and therefore we have

$$\hat{Y}(E) = \sum_j (\pm)\, \hat{Y}(E'_j)$$

as a consistent and unbiased estimate of COUNT(E) Assuming that a simple random sample is taken independently for each $\hat{Y}(E'_j)$, an unbiased estimate of the variance of the estimator $\hat{Y}(E)$, denoted $\hat{Var}(\hat{Y}(E))$, is then

$$\hat{Var}(\hat{Y}(E)) = \sum_j \hat{Var}(\hat{Y}(E'_j))$$

For simple random sampling without replacement, it is usually assumed that the estimate $\hat{Y}(E'_j)$ is normally distributed about the corresponding population value when both the population size and sample size are large [Coch 77] Since any linear combination of normal random variables is normally distributed, $\hat{Y}(E)$ is also normally distributed, and, hence, a confidence interval of $\hat{Y}(E)$ is then

$$\hat{Y}(E) \pm z \times \sqrt{\hat{Var}(\hat{Y}(E))}$$

where z is the value of the normal deviate corresponding to the desired confidence level

For a complex sampling plan, such as the one discussed in section 4, the normality assumption may not be true However, Chebyshev's theorem [SMO 86] states that for any $k \geq 1$, at least $(1-\frac{1}{k^2})$ of the measurements, e g , $\hat{Y}(E)$, from any distribution must lie within k standard deviations of their mean This enables us to derive a confidence interval with the desired confidence level for those cases which the normality assumption is not true For example, setting k=2 yields that at least 75% of any set of measurements must lie within two standard deviations of their mean If a priori knowledge of the distribution of the measurement is known, a higher confidence level can be obtained

3.4 Incorporating Projection

Consider the query COUNT(π_{ATR}(r)), where ATR is a set of attributes in the scheme of relation instance r If ATR contains a key, $\pi_{ATR}(r)$ does not decrease the number of tuples in r Therefore, we can simply ignore the projection operation in the query as far as the estimator is concerned However, if ATR does not contain a key then each set of tuples in r that have the same values for the ATR attributes will generate only one resulting tuple Unfortunately, the estimator of section 3 1 fails to model this characteristic We need to find a new estimator for the projection operation, and more generally, for a query containing projections In the following discussion, we always assume that ATR does not contain a key

Goodman [Good 49] deals with a similar problem in which the estimation of the number of classes K in a population is of interest. A consistent and unbiased estimator for K is proposed in [Good 49], based on a simple random sampling scheme, as

$$\sum_{i=1}^{i=m} A_i x_i$$

where $A_i = 1-(-1)^i \frac{[N-m+i-1]^{(i)}}{m^{(i)}}$,

and x_i is the number of classes containing i elements in a sample of size m, N is the size of the population and

$$n^{(i)} = \begin{cases} n(n-1)\ (n-i+1) & \text{for } i>0 \\ 1 & \text{for } i=0 \end{cases}$$

Goodman also gives the formula for computing the variance of the estimator Below we use Goodman's estimator as the building block for an estimator of a query containing projections

Usually, during the evaluation of a query, duplicate tuples generated by a π_{ATR}(r) operation are removed immediately at the end of each π_{ATR}(r) operation due to the efficiency consideration of the subsequent operations However, it does not make any difference to the result of the query if all the duplicate removal processes are deferred to the last stage of processing the query Consider the temporary result immediately before the last stage of removing duplicates Each set of duplicates, called a group, eventually generates a single tuple in the result of the query, and thus, there is a one-to-one correspondence between the groups and the resulting tuples Therefore, estimating the number of resulting tuples of a query amounts to estimating the number of groups of tuples without removing the duplicates This suggests that the question of counting the number of resulting tuples can be transformed into the question of counting the number of groups of tuples without removing duplicates

A. Projection plus Selection, Natural Join, and Intersection Operators

Let E be an expression containing operators π, σ, $\bowtie$ and $\cap$, and n relations as operands As before, we consider tuples in a relation as points in a dimension A point in the n-dimensional space assumes the value 1 if and only if the set of operand tuples corresponding to the point can create a tuple in the result. Note that several points may correspond to the same resulting tuple because of the projection operations Applying Goodman's estimator to a simple random sample of the points gives an unbiased estimate of the number of groups of points, say K, assuming that all the 0's are in a single group Thus, K−1 is the number of distinct groups (of points with values 1), i e , the number of tuples in the response to the query

B. Projection plus Union and Difference Operators

For an expression E that contains operators $\cup$, $-$, $\cap$, σ, π, and $\bowtie$, except that projections do not operate on set differences, Lemmas 1 and 2 state that COUNT(E) can be computed as $\sum_j (\pm)\text{COUNT}(E'_j)$, where E'_j contains operators $\cap$, $\bowtie$, σ, and π Goodman's estimator serves as a consistent and unbiased estimate of COUNT(E'_j) when E'_j contains projections Hence, a consistent and unbiased estimate of COUNT(E) is obtained when E does not contain any projections operating on differences.

Now, we consider the cases in which projections operate on set differences The simplest case among them is $\pi(r_1-r_2)$ Note that, in section 3 3, we have proposed two approaches to estimating COUNT(r_1-r_2) One utilizes the intersection operations and the inclusion and exclusion principle The other involves checking a sample of r_1 completely with the whole of relation r_2 In order for the first approach to apply, an expression E has to be first transformed into a set of subexpressions E_i's combined by $\cup$'s and $-$'s while E_i's themselves do not contains unions or differences, since neither the Goodman's estimator nor $\hat{Y}$ can directly estimate an expression containing unions and differences Unfortunately, $\pi(r_1-r_2)\neq\pi(r_1)-\pi(r_2)$, and the first approach fails to apply As for the second approach, if we take a simple random sample from r_1, check it with the whole of relation r_2, and then apply Goodman's estimator, we can get an unbiased estimator for COUNT($\pi(r_1-r_2)$) However, this method is possibly expensive since it involves evaluating the whole relation r_2

We now extend the second approach to the general case $\pi(E)$, where E is an arbitrary relational algebra expression with difference operators First, by Lemma 3 below, we transform π(E) into the form,

$$\pi(E) = \pi(E_1) \cup \pi(E_2) \cup \quad \cup \pi(E_k)$$

where E_i, $1 \le i \le k$, does not contain any set union operators Assume that E_i contains difference operators and n operand relations r_1, ,r_m, ,r_n, where relations r_{m+1}, ,r_n are preceded by some difference operators, directly or indirectly To estimate COUNT($\pi(E_i)$), each sample unit taken from the space r_1 r_m, not from r_1 r_n, is evaluated with the whole of relations r_{m+1}, ,r_n in computing the query $\pi(E_i)$ A point $p(f_1(t_1),f_2(t_2),$,$f_m(t_m))$ in the space $r_1 r_2$ r_m has the value 1 if and only if the set of tuples $\{t_i\}$ and r_{m+1}, ,r_n evaluated together can produce a resulting tuple in $\pi(E_i)$ The reasoning behind this is that the set of all the resulting tuples of E_i are constructed from a subset of the space r_1, ·,r_m, and the set of relations r_{m+1}, , r_n are used to define such a subspace For example, $E_i = r_1 \bowtie (r_2-(r_3-r_4)) \subseteq r_1 \bowtie r_2$ The expression r_3-r_4 is used to remove some tuples from r_2, and hence remove some tuples from $r_1 \bowtie r_2$ A simple random sample is taken from $r_1 r_2$, and each sample unit (point) $p(f_1(t_1),f_2(t_2))$ is evaluated with the whole of relations r_3 and r_4 to determine if $\{t_i\}$ produces a resulting tuple in computing E_i

Goodman's estimate serves as a consistent and unbiased estimate for the number of groups of 1's in the space r_1 r_m, i e , COUNT($\pi(E_i)$) Again, by applying the principle of inclusion and exclusion to expression (2) above, we construct a consistent and unbiased estimate for COUNT(π(E)) as $\sum_j (\pm)$ COUNT(E'_j), where E'_j is $\pi(E_i)$ or intersections of $\pi(E_i)$ Please note that the point space for $\pi(E_i)\cap\pi(E_j)$ is r_{i+1} $r_{i+m} r_{i+m+1}$ r_{i+m+k}, where r_{i+1} r_{i+m}, and r_{i+m+1} r_{i+m+k} are the point spaces corresponding to E_i and E_j respectively

Lemma 3 Let E be an arbitrary relational algebra expression E can always be rewritten as

$E = E_1 \cup E_2 \cup \quad \cup E_n$

where E_i does not have any $\cup$'s

4. Estimation of COUNT(E) Based on Cluster Sampling

As far as the estimate of COUNT(E) is concerned, which is computed as $\sum_j (\pm)$COUNT(E'_j), a sample does

not have to be a simple random sample In practice, methods of sampling other than random sampling are at times preferable to simple random sampling on the grounds of convenience, cost, and increased precision There are several enhancements for reducing the cost of sampling

(1) *make the most use of the sampled data* A simple random sample of size m from an n-dimensional point space needs m × n tuple retrievals, m retrievals from each of the n operand relations However, with the same set of sampled data, we can construct m^n sample units (see section 4 2) This method greatly improves the utilization of sampled data On the other hand, this method makes it more complex to compute the variance since the sample units are not independent any more

(2) *use the same sample for all occurrences of an operand relation* A relation r_i may occur in more than one expressions of E'_j's Using the same sample for all occurrences of r_i may save time on retrieving separate samples, especially when disk accesses are considered to be expensive As in (1) above, this method also increases the CPU time cost in computing the covariance of two estimators, say $\hat{COUNT}(E'_k)$ and $\hat{COUNT}(E'_l)$, if E'_k and E'_l are to contain common operand relations

(3) *use a physical block, instead of a tuple, as a sample unit from a relation* When the database resides on disks, the cost of retrieving a tuple is very close to retrieving a block Taking a physical block as a sample unit (i e , cluster sampling) may increase efficiency and reduce the cost of sampling, assuming that several tuples fit into a single block

In this section, we discuss a sampling plan which incorporates the above listed enchancements Our implementation (section 5) also utilizes the sampling plan of this section+

4 1 Introduction to Cluster Sampling

Cluster sampling [Coch 77, Sukh 84] denotes the method of selections in which the sample unit is a cluster of elements Each cluster may contain different number of elements It is usually more convenient and cheaper to take a sample of given number of elements clustered in a few large units than take a sample of elements scattered over an entire area On the other hand, for a given sample size, small sample units often give higher precision than the large ones Since the average cost of obtaining an element is cheaper in cluster sampling, when the cost is balanced against precision, the larger unit may prove superior That is, for a given cost (time limit), a larger sample can be obtained through cluster sampling, and hence a higher precision estimation may be obtained For cluster sampling to be efficient, the clusters should be so formed that the variation between cluster means is as small as possible while the variation within clusters is as large as possible [Coch 77, Sukh 84] contain more details

4.2. Sampling and Execution Plan

As in section 3 2, we consider a relation with k tuples as a set of k points in one-dimensional space For simplicity of the following discussion, we assume that points on an axis inherit the same ordering as that of the corresponding tuples in the physical secondary storage Let D_i be the number of physical storage (i e , file) blocks in relation r_i The axis that a relation r_i is mapped to can, therefore, be divided into D_i number of segments with the j_{th} segment corresponding to the j_{th} physical storage block Each segment contains the same number of points as that of tuples in the corresponding block.

Let E be a relational algebra expression that contains only operators σ, $\cap$, and $\bowtie$, and n operand relations Now, consider the n-dimensional point space of E Clearly, this space can be divided into B *space blocks*, where $B=D_1\times \quad \times D_n$ Each space block b(j,k, ,l) contains $M_b=M_j\times M_k\times \quad \times M_l$ points, where $M_j, M_k, \quad , M_l$ denote the number of tuples in the j_{th} block of r_1, in the k_{th} block of r_2, , and in the l_{th} block of r_n, respectively Hereafter, the set of physical storage blocks corresponding to b(j,k, ,l) are called the *operand blocks* of b(j,k, ,l) A block such as b(j,k, ,l) constitutes a sample unit of the following sampling method

To obtain a sample unit from an n-dimensional point space, we need n physical block accesses, one access from each of the n operand relations As a result, for a random sample of b units, approximately $b\times n$ physical block accesses are needed assuming that $b << D_i$ for all i, $1\le i\le n$, and hence the probability that any physical block is referred to more than once is nearly zero Clearly, this random sam-

+ Goodman's estimator is based on the simple random sampling method, not on the cluster sampling Nevertheless, in this section, we use Goodman's estimator for the projection operators Experimental results section lists the performance of the Goodman's estimator for a limited set of RA expressions and a set of database instances

pling method does not make the most use of the sampled data, since there can be b^n sample units constructed from the same set of sampled data (However, in such a case they would be correlated) Therefore, for a given size b of a sample, we can randomly choose d_i blocks from r_i, such that $b=d_1 \times d_2 \times \ldots \times d_n$ This approach needs only $d_1+d_2+\ldots+d_n$, instead of $b \times n$ physical, block accesses Another advantage of using this kind of sample set is that the standard query evaluation techniques can be used only once after the sampled blocks from operand relations are determined In the case of taking a random sample of units, each unit is separately evaluated by substituting its operand blocks for relations in E This process has to be repeated b times for a sample of b units The disadvantage of the new method is that it involves more computation in computing the variance of an estimation since the sample units are correlated We will discuss in section 4 3 how to compute the variance of an estimator and the covariance of two estimators

As stated in section 3 3, COUNT(E) is computed as $\sum_j (\pm)$ COUNT(E'_j) Now, we describe an algorithm CONSTRUCT-SAMPLES for constructing a set of samples for all the E'_j's from samples of operand relations The algorithm is implemented in a prototype database management system, called ERAM [DFHO 86], which uses the relational algebra (RA) as its query language An RA expression is parsed as a binary tree (as shown in Figure 1, for example) with nodes being either a relation or an operator Operators are evaluated when the parse tree is traversed in preorder For simplicity, we assume selection and projection (i e , unary operators) have only left subtrees without considering the selection formula and projected attributes

Please note that the $\cup$'s in the return statements of the algorithm CONSTRUCT-SAMPLES simply denote set union. Below we show one example of the application of the CONSTRUCT-SAMPLES algorithm

4.3 Estimation of COUNT(E)

Let E'_i be a relational algebra expression with operators σ, $\cap$, and $\bowtie$, and n operand relations As before, a point in the n-dimensional point space of E'_i assumes a binary value, 0 or 1, depending on whether the set of corresponding operand tuples can generate a resulting tuple in E'_i or not. The value of a space block b(i_1,i_2, ,i_n),

Algorithm CONSTRUCT-SAMPLES (tree)

* *Input* a *tree* representing a relational algebra expression, and the database instances
* *Output* a set of sample blocks for all the expressions E'_j, which are constructed from samples of the operand relations in the input RA expression
* *Variables* root denotes the root node of the "current" tree
  preceding-π is a global variable, zero initially, and is used to check if a difference operator is preceded by π operators
  left-set and right set are set variables that contain sets of sample blocks returned by calling *CONSTRUCT-SAMPLES* with left and right subtree as input arguments, respectively
  left-subtree (node) and right-subtree (node) denote the left and right subtrees of node, respectively
  S_j denotes either a set of sample blocks of relation r_j or a set of sample blocks of expression E_j

Begin
if *tree* is not empty **then**
begin
if root is π **then** preceding-π = preceding-π + 1,
if root is a relation r_j **then** return ({ S_j }),
left-set = *CONSTRUCT-SAMPLES* (left-subtree(root)),
if root is π **then**
begin
preceding-π = preceding-π − 1,
return({ $\pi(S_j) \mid S_j \in$ left-set }),
end;
if root is σ **then** return({ $\sigma(S_j) \mid S_j \in$ left-set }),
if root is − **then**
if (preceding-π > 0) **then**
begin
evaluate the whole right subtree of root (using the actual relation instances) and denote the result by E_k,
return({ $S_j - E_k \mid S_j \in$ left-set }),
end,
else begin
right-set = *CONSTRUCT-SAMPLES* (right-subtree),
return(left-set $\cup$ {$S_j \cap S_k \mid S_j \in$ left-set and $S_k \in$ right-set}),
end,
right-set = *CONSTRUCT-SAMPLES* (right-subtree(root)),
if root is $\bowtie$ **then**
return({ $S_j \bowtie S_k \mid S_j \in$ left-set and $S_k \in$ right-set }),
if root is $\cap$ **then**
return({ $S_j \cap S_k \mid S_j \in$ left-set and $S_k \in$ right-set }),
if root is $\cup$ **then**
return(left-set $\cup$ right-set $\cup$
{ $S_j \cap S_k \mid S_j \in$ left-set and $S_k \in$ right-set}),
end.
End.

Example 3. Let the query Q be COUNT($r_1-(r_2\cup r_3)$) Then, following the procedure ESTIMATE-COUNT in section 3, we have Q= COUNT(r_1) − COUNT($r_1\cap r_2$) − COUNT($r_1\cap r_3$) + COUNT($r_1\cap r_2\cap r_3$) Figure 1 shows how to construct the set of samples, i e, S_1, $S_1\cap S_2$, $S_1\cap S_3$ and $S_1\cap S_2\cap S_3$, for estimating the COUNT of expressions r_1, $r_1\cap r_2$, $r_1\cap r_3$, and $r_1\cap r_2\cap r_3$, respectively Samples are shown next to each node in the parse tree

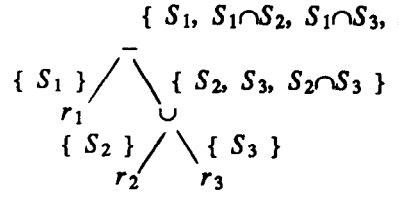


Figure 1 A Parse tree and its samples

denoted by $y_{i_1\ i_n}$, is defined to be the sum of the values of all the points in that space block $\hat{Y}_b(E'_i)$, an estimator of $Y(E'_i)$ based on the cluster sampling plan, is then $\frac{B_i}{b_i}\sum_{i_1}^{d_1}\cdots\sum_{i_n}^{d_n} y_{i_1\ i_n}$, where B_i is the total number of space blocks in the point space of E'_i and b_i is the number of space blocks in a sample $\bar{Y}_b(E'_i)$ (i e, population mean) and $\bar{y}_b(E'_i)$ (i e, sample mean) are defined to be $\frac{Y(E'_i)}{B_i}$ and $\frac{1}{b_i}\sum_{i_1}^{d_1}\cdots\sum_{i_n}^{d_n} y_{i_1\ i_n}$, respectively

Theorem 4 $\hat{Y}_b(E'_i)$ is a consistent and unbiased estimator of COUNT(E'_i)

Note that the number of tuples in each physical block and the ordering of tuples in the files do not effect the consistency and unbiasedness of the estimator, $\hat{Y}_b(E'_i)$ However, they may effect the variance of the estimate

Assume there are n and m operand relations involved in E'_i and E'_k respectively, and among them j relations appear in both E'_i and E'_k Let R be the set of all relations involved in the query, i e , |R|=n+m−j, and J, |J|=j, be the set of common relations Let G = { g | g ⊆ J }, |G|= 2^j In the following theorems, each g, a subset of the common relations, specifies a group of product terms $(y_{i_1\ i_n}-\bar{Y}_b(E'_i))$ $(y_{k_1\ k_m}-\bar{Y}_b(E'_k))$ such that $y_{i_1\ i_n}$ and $y_{k_1\ k_m}$ have the same coordinate values for those coordinates corresponding to relations in g Such a pair of blocks as $y_{i_1\ i_n}$ and $y_{k_1\ k_m}$ are correlated, since they are constructed partly from the same operand blocks

Theorem 5 The covariance of two estimators $\hat{Y}_b(E'_i)$ and $\hat{Y}_b(E'_k)$, denoted by $Cov(\hat{Y}_b(E'_i),\hat{Y}_b(E'_k))$, is computed as

$$\prod_{r_j\in J}\frac{D_j}{d_j}\sum_{g\in G, g\neq\phi}\left\{\left[1-\prod_{r_s\in g}\frac{d_s-1}{D_s-1}\right]\left[\prod_{r_u\in J, r_u\notin g}\frac{d_u-1}{D_u-1}\right]S_g^2\right\}$$

where S_g^2 is

$$\sum_{i_s=k_{s'}}^{D_{i_s}}\cdots\sum_{i_t=k_{t'}}^{D_{i_t}}\left\{\left[\sum_{i_u}^{D_{i_u}}\sum_{i_v}^{D_{i_v}}\sum_{i_a}^{D_{i_a}}\cdots\sum_{i_b}^{D_{i_b}}(y_{i_1\ldots i_n}-\bar{Y}_b(E'_i))\right]\left[\sum_{k_u}^{D_{k_u}}\sum_{k_v}^{D_{k_v}}\sum_{k_a}^{D_{k_a}}\cdots\sum_{k_f}^{D_{k_f}}(y_{k_1\ldots k_m}-\bar{Y}_b(E'_k))\right]\right\}$$

and $\{r_{i_s}(=r_{k_{s'}}),\ \ldots,\ r_{i_t}(=r_{k_{t'}})\}$ = g, $r_{i_u}(=r_{k_{u'}}),\ \ldots,\ r_{i_v}(=r_{k_{v'}})$ are in J, but not in g, $r_{i_a},\ \ldots,r_{i_b}$ are in E'_i, but not in J, $r_{k_a},\ \ldots,r_{k_f}$ are in E'_k, but not in J

Theorem 6 An unbiased estimator for $\mathrm{Cov}(\hat{Y}_b(E'_i),\hat{Y}_b(E'_k))$, denoted by $\hat{Cov}(\hat{Y}_b(E'_i),\hat{Y}_b(E'_k))$, is

$$\prod_{r_j\in R}\frac{D_j}{d_j}\prod_{r_j\in J}\frac{D_j-1}{d_j-1}\sum_{g\in G, g\neq\phi}\left\{\left[1-\prod_{r_s\in g}\frac{d_s-1}{D_s-1}\right]s_g^2\right\}$$

where s_g^2 is

$$\sum_{i_s=k_{s'}}^{d_{i_s}}\cdots\sum_{i_t=k_{t'}}^{d_{i_t}}\left\{\left[\sum_{i_u}^{d_{i_u}}\sum_{i_v}^{d_{i_v}}\sum_{i_a}^{d_{i_a}}\cdots\sum_{i_b}^{d_{i_b}}(y_{i_1\ldots i_n}-\bar{y}_b(E'_i))\right]\left[\sum_{k_u}^{d_{k_u}}\sum_{k_v}^{d_{k_v}}\sum_{k_a}^{d_{k_a}}\cdots\sum_{k_f}^{d_{k_f}}(y_{k_1\ldots k_m}-\bar{y}_b(E'_k))\right]\right\}$$

An unbiased estimate of the variance of $\hat{Y}_b$ (E), computed as $\sum_{j=1}^{j=n}(\pm)\,\hat{Y}_b(E'_j)$, is then given as

$$\hat{Var}(\hat{Y}_b(E))=\sum_{i=1}^{i=n}\sum_{k=1}^{k=n}\hat{Cov}(\hat{Y}_b(E'_i),\hat{Y}_b(E'_k))$$
$$=\sum_{i=1}^{i=n}\hat{Var}(\hat{Y}_b(E_i))+\sum_{i=1}^{i=n}\sum_{k\neq i}^{k=n}\hat{Cov}(\hat{Y}_b(E'_i),\hat{Y}_b(E'_k))$$

6. Preliminary Experimental Results

This section reports preliminary experimental results on the proposed estimators Sampling and estimation of COUNT queries have been incorporated into a prototype database management system, called ERAM [DFHO 86], which uses relational algebra expressions as its query language The system is built on top of Unix 4 3 BSD

operating system on Vax 11/780 In this section, we report experimental results using a single relational algebra operator Extensive experiments on relational algebra expressions with more than one operator are still being performed In what follows, each artificial relation instance has 5000 tuples, with tuple size of 200 bytes That is, each relation instance consists of 1000 disk blocks (1k bytes in each disk block) with 5 tuples in each disk blocks Each disk block is a sampling unit from a relation Tuples in a relation are randomly distributed. Please note that the precision of an estimation in a real life database application may be different (better or worse) from the experimental results reported here, depending on how the tuples are clustered in the disk blocks

In each of the following tables, a table element represents a relative error, i e., *coefficient of variation*, defined as $\sqrt{\hat{Var}\ (COUNT(E))}$ / COUNT(E) × 100%, of the estimator, for different combinations of sample fractions f (in percentage) from each of the operand relations and the actual query result N (represented by the number of resulting tuples), i e , COUNT(E) So, for example, in the table for selection, the upper leftmost element (Error=130, N=50 and f=1) represents the fact that for a query returning N=50 tuples, using a sample of 1% of the original relation size, the estimator produced, on the average, estimations with deviation of 1 3 (i e , 130%) × N Each estimated error is computed using 50 independent experiments.

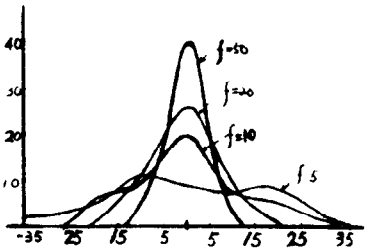
In the figures below, each curve gives the distribution of the relative errors in 50 experiments for a specific N and f value The horizontal axis denotes the relative errors (in percentage) of the estimates, and the vertical axis denotes the number of times (out of 50 experiments) the estimates falling into a certain relative error range There are 7 different relative error ranges (e g., (-35,-25),(-25,-15), ,(25,35)) on the horizontal axis Thus, the curves are actually the traces of the corresponding histograms For example, the curve with f=50 for selection operation is obtained from the fact that out of 50 estimations, 41 fall within the range N±0 05N, 5 are in (N-0 15N, N-0 05N) and the rest are in (N+0 05N, N+0 15N)

(A). Selection operation As observed in the table, estimations with higher precision are obtained using large sample sizes for the same query Also, for queries that return high counts, better relative precision is obtained for the same size of sample It is clear from the following figure that the distribution curves of estimation are more centered on the true value when the sample size gets large Also, when the sample size gets large, the distribution has a bell shaped curve The above results are also true for join, intersection, union and difference operations

f

N	1	5	10	20	50
50	130	68	42	28	14
250	72	29	14	12	7
500	41	22	13	8	4
1000	24	13	7	6	3
2500	15	8	4	3	1

N=500

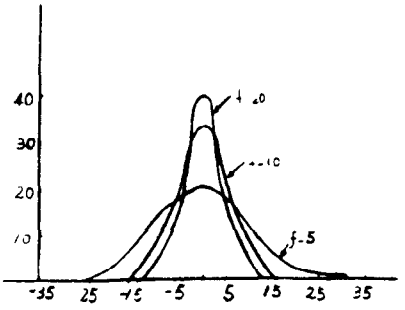


(B). Join operation In the literature, Pearson distributions are claimed to be well suited for describing the distribution of attribute values in real-life database environments [Chri 83] The univariate Pearson Type 2 distributions, by changing a parameter, range from uniform distribution to approximate normal distribution In this experiment, we assume that join is on a single domain and choose the normal distribution for the join attribute values Unfortunately, the estimator proposed in section 4 for estimating the (co)variance may sometimes give negative values Several methods in [Coch 77] are proposed to deal with this problem. In our experiments, we have ignored the cases that returned negative variance values As observed from the table, precision increases faster at low sample fractions (e g , f < 10) than at higher sample fractions When the sample fraction is around 10%, reasonable good estimates are obtained For example, as observed from the curve, there are 32 experiments out of 50 that are in the range of N± 5% × N with f=10

f

N	1	3	5	10	20
7900	92	43	32	14	11
26800	56	25	20	9	5
56000	31	15	10	5	3
70000	34	13	8	5	3

N=56000

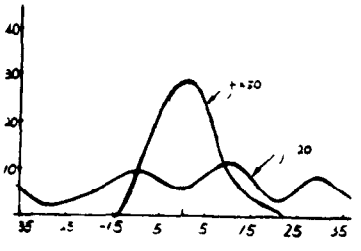


(C). **Intersection Operation** Estimations of the COUNT of union and difference operations use intersection Intersection can be considered as a join operation returning relative low number of resulting tuples For low sample fraction (e g $f < 5$), the estimator gives large deviation Below, we tabulate the performance of the intersection operation

f

N	1	5	10	20	50
250	560	122	69	28	9
500	392	86	40	25	8
1000	360	60	36	15	4
2500	151	37	20	9	3
3500	113	22	8	5	1

N=500

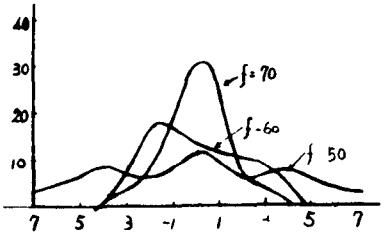


(D). **Projection Operation.** We use Goodman's estimator for the experiment which is based on the cluster sampling plan proposed in section 4 Again, the projected values (on a single domain) assume normal distributions In order to get reasonable estimates (e g, 60%, 70%), larger sample size is needed for those cases which have heavy duplicates In the following table, - denotes those cases in which "unreasonable" estimates (e g , those that return negative values) are found Large sample size may prevent these unreasonable estimates Also, the behavior of Goodman's estimator is different than the other estimators That is, when the sample fraction is lower than some "threshold", Goodman's estimator always gives unreasonable estimates On the other hand, once the sample fraction is higher than the "threshold", the Goodman's estimate converges very fast.

f

N	30	40	50	60	70
37	-	-	-	5	4
290	-	-	5	2	1
518	-	-	5	2	1
1266	-	10	4	2	1

N=518



References

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

[Chri 83] Christodoulakis, S , "Estimating Record Selectivities", Information Systems, Vol 8, 1983

[Devo 84] Devore, J L , "Probability & Statistics for Engineering and Sciences", Brook/Cole, 1984

[DFHO 86] Datta, A , Fournier, B , Hou, W-C , and Ozsoyoglu, G , "The Implementation of SSDB", Proc Third International Workshop on Statistical Database Management", July 1986.

[Good 49] Goodman, L.A., "On the Estimation of the Number of Classes in a Population", Ann Math Sta., 1949

[HoOT 87] Hou, W-C , Ozsoyoglu, G , and Taneja, B k., "Statistical Estimators for Relational Algebra Expressions", Tech Rpt. CES-87-15, CWRU, 1987

[Liu 68] Liu, C L , "Introduction to Combinatorial Mathematics", McGraw-Hill, 1968.

[Morg 81] Morgenstein, J P , "Computer Based Management Information Systems Embodying Answer Accuracy As a User Parameter", Ph D Thesis, Univ of California, Berkeley, 1981

[Olke 86] Olken, F , "Physical Database Support for Scientific and Statistical Databases", Third Int. Scientific and Statistical Databases Workshop, 1986

[OlkR 86] Olken, F and Rotem, D , "Simple Random Sampling from Relational Databases", VLDB Conf 1986

[Ross 80] Ross, S M , "Introduction to Probability Models", 2nd Ed., Academic Press, 1980

[Rowe 85] Rowe, N C , "Antisampling for Estimation An overview", IEE Trans on Software Eng., Oct 1985

[SMO 86] Scheaffer, Mendenhall, and Ott, "Elementary Survey Sampling", 3rd Ed , Duxbury press, 1986

[Sukh 84] Sukhatme, P V , etc , "Sampling Theory of Surveys Application", 3rd Ed , New Delhi, India and Iowa State Univ Press, 1984