

Shantanu Joshi · Christopher Jermaine

Sampling-Based Estimators for Subset-Based Queries

Received: date / Accepted: date

Abstract We consider the problem of using sampling to estimate the result of an aggregation operation over a subset-based SQL query, where a subquery is correlated to an outer query by a `NOT EXISTS`, `NOT IN`, `EXISTS` or `IN` clause. We design an unbiased estimator for our query and prove that it is indeed unbiased. We then provide a second, biased estimator that makes use of the superpopulation concept from statistics to minimize the mean squared error of the resulting estimate. The two estimators are tested over an extensive set of experiments.

Keywords Sampling, Approximate Query Processing, Aggregate Query Processing

1 Introduction

Sampling is well-established as a method for dealing with very large volumes of data, when it is simply not practical or desirable to perform the computation over the entire data set. Sampling has several advantages compared to other widely-studied approximation methodologies from the data management literature such as wavelets [16], histograms [7] and sketches [21]. Not the least of those is generality: it is very easy to efficiently draw a sample from a large data set in a single pass using reservoir techniques [20]. Then, once

Shantanu Joshi
Computer and Information Science and Engineering,
University of Florida, Gainesville, FL 32611.
E-mail: ssjoshi@cise.ufl.edu

Christopher Jermaine
Computer and Information Science and Engineering,
University of Florida, Gainesville, FL 32611.
E-mail: cjermain@cise.ufl.edu

the sample has been drawn it is possible to guess, with greater or lesser accuracy, the answer to virtually any statistical query over those sets. Samples can easily handle many different database queries, including complex functions in relational selection and join predicates. The same cannot be said of the other approximation methods, which generally require more knowledge of the query during synopsis construction, such as the attribute that will appear in the `SELECT` clause of the SQL query corresponding to the desired statistical calculation.

However, one class of aggregate queries that remain difficult or impossible to answer with samples are the so-called “subset” queries, which can generally be written in SQL in the form:

```
SELECT SUM ( $f_1(r)$ )
FROM R as r
WHERE  $f_2(r)$  AND NOT EXISTS
(SELECT * FROM S AS s
WHERE  $f_3(r, s)$ )
```

Note that the function f_2 can be incorporated into f_1 if we have f_1 evaluate to zero if f_2 is not *true*; thus, in the remainder of the paper we will ignore f_2 . An example of such a query is: “Find the total salary of all employees who have not made a sale in the past year”:

```
SELECT SUM (e.SAL)
FROM EMP AS e
WHERE NOT EXISTS
(SELECT * FROM SALE AS s
WHERE s.EID = e.EID)
```

A general solution to this problem would greatly extend the class of database-style queries that are amenable to being answered via random sampling. For example, there is a very close relationship between such queries and those obtained by removing the `NOT` in the subquery. Using the terminology introduced later in the paper, all records from `EMP` with i records in `SALES` are called “class i ” records. The only difference between `NOT EXISTS` and `EXISTS` is that the former query computes a sum over all class 0 records, whereas the latter query computes a sum over all class $i > 0$ records. Since any reasonable estimator for `NOT EXISTS` will likely have to compute an estimated sum over each class, a solution for `NOT EXISTS` should immediately suggest a solution for `EXISTS`. Also, nested queries having an `IN` (or `NOT IN`) clause can be easily rewritten as a nested query having the `EXISTS` (or `NOT EXISTS`) clause. For example, the query “Find the total salary of all employees who have not made a sale in the past year” given above can also be written as:

```
SELECT SUM (e.SAL)
FROM EMP as e
WHERE e.EID NOT IN
(SELECT s.EID FROM SALE AS s)
```

Furthermore, a solution to the problem of sampling for subset queries would allow sampling-based aggregates over SQL `DISTINCT` queries, which can easily be re-written as subset queries. For example:

```
SELECT SUM (DISTINCT e.SAL)
FROM EMP AS e
```

is equivalent to:

```
SELECT SUM (e.SAL)
FROM EMP AS e
WHERE NOT EXISTS
(SELECT * FROM EMP AS e2
 WHERE id(e) < id(e2)
 AND e.SAL = e2.SAL)
```

In this query, *id* is a function that returns the row identifier for the record in question. Some work has considered the problem of sampling for counts of distinct attribute values [14,28], but aggregates over `DISTINCT` queries remains an open problem. Similarly, it is possible to write an aggregate query where records with identical values may appear more than once in the data, but should be considered no more than once by the aggregate function as a subset-based SQL query. For example:

```
SELECT SUM (e.SAL)
FROM EMP AS e
WHERE NOT EXISTS
(SELECT * FROM EMP AS e2
 WHERE id(e) < id(e2)
 AND identical(e, e2))
```

In this query, the function *identical* returns *true* if the two records contain identical values for all of their attributes. This would be very useful in computations where the same data object may be seen at many sites in a distributed environment (packets in an IP network, for example). Previous work has considered how to perform sampling in such a distributed system [4,23], but not how to deal with the duplicate data problem.

Unfortunately, it turns out that handling subset queries using sampling is exceedingly difficult, due to the fact that the subquery in a subset query is not asking for a mean or a sum – tasks for which sampling is particularly well-suited. Rather, the subquery is asking whether we will *ever* see a match for each tuple from the outer relation. By looking at an individual tuple, this is very hard to guess: either we have seen a match already on our sample (in which case we are assured that the inner relation has a match), or we have not, in which case we may have almost no way to guess whether we will ever see a match. For example, imagine that employee Joe does not have a sale

in a 10% sample of the `SALE` relation. How can we guess whether or not he has a sale in the remaining 90%?

There is little relevant work in the statistical literature to suggest how to tackle subset queries, because such queries ask a simultaneous question linking two populations (database tables `EMP` and `SALE` in our example), which is an uncommon question in traditional applications of finite population sampling. Outside of the work on sample from the number of distinct values [28, 14, 18] and one method that requires an index on the inner relation [3], there is also little relevant work in the data management literature; we presume this is due to the difficulty of the problem; researchers have considered the difficulty of the more limited problem of sampling for distinct values in some detail [28].

Our Contributions

In this paper, we consider the problem of developing sampling-based statistical estimators for such queries. In the remainder of the paper, we assume without-replacement sampling, though our methods could easily be extended to other sampling plans. Given the difficulty of the problem, it is perhaps not surprising that significant statistical and mathematical machinery is required for a satisfactory solution.

Our first contribution is to develop an *unbiased* estimator, which is the traditional first step when searching for a good statistical estimator. An unbiased estimator is one that is correct on expectation; that is, if an unbiased estimator is run an infinite number of times, then the average over all of the trials would be exactly the same as the correct answer to the query. The reason that an unbiased estimator is the natural first choice is that if the estimator has low variance¹, then the fact that it is correct on average implies that it will always be very close to the correct answer.

Unfortunately, it turns out that the unbiased estimator we develop often has high variance, which we prove analytically and demonstrate experimentally. Since it is easy to argue that our unbiased estimator is the *only* unbiased estimator for a certain subclass of subset-based queries (see the Related Work section of this paper), it is perhaps doubtful that a better unbiased estimator exists.

Thus, we also propose a novel, biased estimator that makes use of a statistical technique called “superpopulation modeling”. Superpopulation modeling is an example of a so-called Bayesian statistical technique [11]. Bayesian methods generally make use of mild and reasonable distributional assumptions about the data in order to greatly increase estimation accuracy, and have become very popular in statistics in the last few decades. Using this method in the context of answering subset-based queries presents a number of significant technical challenges whose solutions are detailed in this paper, including:

- The definition of an appropriate generative statistical model for the problem of sampling for subset-based queries.

¹ Variance is the statistical measure of the random variability of an estimator.

-
- The derivation of a unique Expectation Maximization algorithm [22] to learn the model from the database samples.
 - The development of algorithms for efficiently generating many new random data sets from the model, without actually having to materialize them.

Through an extensive set of experiments, we show that the resulting biased Bayesian estimator has excellent accuracy on a wide variety of data. The biased estimator also has the desirable property that it provides something closely related to classical confidence bounds, that can be used to give the user an idea of the accuracy of the associated estimate.

Paper Organization

The paper is organized as follows. In Section 2, we present a simple concurrent sampling estimator described in detail in previous work. Section 3 gives details of a sampling-based unbiased estimator that we develop along with a proof of unbiasedness and variance calculations. We present the intuition and framework for our superpopulation-based estimator in Section 4. Section 5 presents complete details of our superpopulation-based estimator. In Section 6 we describe our experimental setup and present results. Related work is covered in Section 7 and Section 8 concludes the paper.

2 The Concurrent Estimator

With a little effort, it is not hard to imagine several possible sampling-based estimators for subset queries. In this section, we discuss one very simple (and sometimes unusable) sample-based estimator. This estimator has previously been studied in detail [3], but we present it here because it forms the basis for the unbiased estimator described in the next section.

We begin our description with an even simpler estimation problem. Given a one-attribute relation $R(A)$ consisting of n_R records, imagine that our goal is to estimate the sum over attribute A of all the records in R . A simple, sample-based estimator would be as follows. We obtain a random sample R' of size $n_{R'}$ of all the records of R , compute $total = \sum_{r \in R'} r.A$, and then scale up $total$ to output $total \times n_R/n_{R'}$ as the estimate for the final sum. Not only is this estimator extremely simple to understand, but it is also unbiased, consistent, and its variance reduces monotonically with increasing sample size.

We can extend this simple idea to define an estimator for the NOT EXISTS query considered in the introduction. We start by obtaining random samples EMP' and $SALE'$ of sizes $n_{EMP'}$ and $n_{SALE'}$, respectively from the relations EMP and $SALE$. We then evaluate the NOT EXISTS query over the samples of the two relations. We compare every record in EMP' with every record in $SALE'$, and if we do not find a matching record (that is, one for which f_3 evaluates to *true*), then we add its f_1 value to the estimated total. Lastly, we scale

up the estimated total by a factor of $n_{\text{EMP}}/n_{\text{EMP}'}$ to obtain the final estimate, which we term M :

$$M = \left(\frac{n_{\text{EMP}}}{n_{\text{EMP}'}} \right) \sum_{e \in \text{EMP}'} f_1(e) \times (1 - \min(1, \text{cnt}(e, \text{SALE}')))$$

In this expression, $\text{cnt}(e, \text{SALE}') = \sum_{s \in \text{SALE}'} I(f_3(e, s))$ where I is the standard indicator function, returning 1 if the boolean argument evaluates to *true*, and 0 otherwise. The algorithm can be slightly modified to accommodate for growing samples of the relations, and has been described in detail in [3], where it is called the “concurrent estimator” since it samples both relations concurrently.

Unfortunately, on expectation, the estimator is often severely biased, meaning that it is, on average, incorrect. The reason for this bias is fairly intuitive. The algorithm compares a record from **EMP** with all records from **SALE'**, and if it does not find a matching record in **SALE'**, it classifies the record as having no match in the entire **SALE** relation. Clearly, this classification may be incorrect for certain records in **EMP**, since although they might have no matching record in **SALE'**, it is possible that they may match with some record from the part of **SALE** that was not included in the sample. As a result, M typically overestimates the answer to the **NOT EXISTS** query. In fact, the bias of M is:

$$\begin{aligned} \text{Bias}(M) = \sum_{e \in \text{EMP}} f_1(e) & (1 - \min(1, \text{cnt}(e, \text{SALE}))) \\ & - \varphi(n_{\text{SALE}}, n_{\text{SALE}'}, \text{cnt}(e, \text{SALE})) \end{aligned}$$

In this expression, φ denotes the hypergeometric probability² that a sample of size $n_{\text{SALE}'}$ will contain none of the $\text{cnt}(e, \text{SALE})$ matching records of e .

The solution that was employed previously to counteract this bias requires an index such as a B+-Tree on the entire **SALE** relation, in order to estimate and correct for $\text{Bias}(M)$. Unfortunately, the requirement for an index severely limits the applicability of the method. If an index on the “join” attribute in the inner relation is not available, the method cannot be used. In a streaming environment where it is not feasible to store **SALE** in its entirety, an index is not practical. The requirement of an index also precludes use of the concurrent estimator for a non-equality predicate in the inner subquery or for non-database environments where sampling might be useful, such as in a distributed system.

In the remainder of the paper, we consider the development of sampling-based estimators for this problem that require nothing but samples from the relations themselves. Our first estimator makes use of a provably unbiased estimator $\widehat{\text{Bias}}(M)$ for $\text{Bias}(M)$. Taken together, $M - \widehat{\text{Bias}}(M)$ is then an unbiased estimator for the final query answer. The second estimator we consider is quite different in character, making use of Bayesian statistical techniques.

² The hypergeometric probability distribution models the distribution of the number of red balls that will be obtained in a sample without replacement of n' balls from an urn containing r red balls and $n - r$ non-red balls.

3 Unbiased Estimator

3.1 High-Level Description

In order to develop an unbiased estimator for $Bias(M)$, it is useful to first re-write the formula for $Bias(M)$ in a slightly different fashion. We subsequently refer to the set of records in **EMP** that have i matches in **SALE** as “class i records”. Denote the sum of the aggregate function over all records of class i by t_i , so $t_i = \sum_{e \in \text{EMP}} f_1(e) \times I(cnt(e, \text{SALE}) = i)$ (note that the final answer to the **NOT EXISTS** query is the quantity t_0). Given that the probability that a record with i matches in **SALE** happens to have no matches in **SALE'** is $\varphi(n_{\text{SALE}}, n_{\text{SALE}'}, i)$, we can re-write the expression for the bias of M as:

$$Bias(M) = \sum_{i=1}^m \varphi(n_{\text{SALE}}, n_{\text{SALE}'}, i) \times t_i \quad (1)$$

The above equation computes the bias of M since it computes the expected sum over the aggregate attribute of all records of **EMP** which are incorrectly classified as class 0 records by M .

Let m be the maximum number of matching records in **SALE** for any record of **EMP**. Equation 1 suggests an unbiased estimator for $Bias(M)$ because it turns out that it is easy to generate an unbiased estimate for t_m : since no records other than those with m matches in **SALE** can have m matches in **SALE'**, we can simply count the sum of the aggregate function f_1 over all such records in our sample, and scale up the total accordingly. The scale-up would also be done to account for the fact that we use **SALE'** and not **SALE** to count matches. Once we have an estimate for t_m , it is possible to estimate t_{m-1} . How? Note that records with $m-1$ matches in **SALE'** must be a member of either class m or class $m-1$. Using our unbiased estimate for t_m , it is possible to guess the total aggregate sum for those records with $m-1$ matches in **SALE'** that in reality have m matches in **SALE**. By subtracting this from the sum for those records with $m-1$ matches in **SALE'** and scaling up accordingly, we can obtain an unbiased estimate for t_{m-1} . In a similar fashion, each unbiased estimate for t_i leads to an unbiased estimate for t_{i-1} . By using this recursive relationship, it is possible to guess in an unbiased fashion the value for each t_i in the expression for $Bias(M)$. This leads to an unbiased estimator for the $Bias(M)$ quantity, which can be subtracted from M to provide an unbiased guess for the query result.

3.2 The Unbiased Estimator In Depth

We now formalize the above ideas to develop an unbiased estimator for each t_k that can be used in conjunction with Equation 1 to develop an unbiased estimator for $Bias(M)$. We use the following additional notation for this section and the remainder of the paper:

- $\Delta_{k,i}$ is a 0/1 (non-random) variable which evaluates to 1 if the i th tuple of **EMP** has k matches in **SALE** and evaluates to 0 otherwise.

-
- s_k is the sum of f_1 over all records of EMP' having k matching records in SALE': $s_k = \sum_{i=1}^{n_{\text{EMP}'}} I(\text{cnt}(e_i, \text{SALE}') = k) \times f_1(e_i)$.
 - α_0 is $n_{\text{EMP}'}/n_{\text{EMP}}$, the sampling fraction of EMP.
 - Y_i is a random variable which governs whether or not the i th record of EMP appears in EMP'.
 - $h(k; n_{\text{SALE}}, n_{\text{SALE}'}, i)$ is the hypergeometric probability that out of the i interesting records in a population of size n_{SALE} , exactly k will appear in a random sample of size $n_{\text{SALE}'}$. For compactness of representation we will refer to this probability as $h(k; i)$ in the remainder of the paper, since our sampling fraction never changes.

We begin by noting that if we consider only those records from EMP which appear in the sample EMP', an unbiased estimator for t_k over EMP' can be expressed as follows:

$$\hat{t}_k = \frac{1}{\alpha_0} \sum_{i=1}^{n_{\text{EMP}'}} Y_i \times \Delta_{k,i} \times f_1(e_i) \quad (2)$$

Unfortunately, this estimator relies on being able to evaluate $\Delta_{k,i}$ for an arbitrary record, which is impossible without scanning the inner relation in its entirety. However, with a little cleverness, it is possible to remove this requirement. We have seen earlier that a record e can have k matches in the sample SALE' provided it has $i \geq k$ matches in SALE. This implies that records from all classes i where $i \geq k$ can contribute to s_k . The contribution of a class i record towards the expected value of s_k is obtained by simply multiplying the probability that it will have k matches in SALE' with its aggregate attribute value. Thus a generic expression to compute the contribution of any arbitrary record from EMP' towards the expected value of s_k can be written as $\sum_{i=k}^m \Delta_{i,j} \times h(k; i) \times f_1(e_j)$. Then, the following random variable has an expected value that is equivalent to the expected value of s_k :

$$\hat{s}_k = \sum_{j=1}^{n_{\text{EMP}'}} \sum_{i=k}^m Y_j \times \Delta_{i,j} \times h(k; i) \times f_1(e_j) \quad (3)$$

The fact that $E[\hat{s}_k] = E[s_k]$ (proven in Section 3.3) is significant, because there is a simple algebraic relationship between the various \hat{s} variables and the various \hat{t} variables. Thus, we can express one set in terms of the other, and then replace each \hat{s}_k with s_k in order to derive an unbiased estimator for each \hat{t} . The benefit of doing this is that since s_k is defined as the sum of f_1 over all records of EMP' having k matching records in SALE', it can be directly evaluated from the samples EMP' and SALE'.

To derive the relationship between \hat{s} and \hat{t} , we start with an expression for \hat{s}_{m-r} using Equation 3:

$$\begin{aligned}
\hat{s}_{m-r} &= \sum_{j=1}^{n_{\text{EMP}}} \sum_{i=m-r}^m Y_j \times \Delta_{i,j} \times h(m-r; i) \times f_1(e_j) \\
&= \sum_{i=m-r}^m h(m-r; i) \sum_{j=1}^{n_{\text{EMP}}} Y_j \times \Delta_{i,j} \times f_1(e_j) \\
&= \sum_{i=0}^r h(m-r; m-r+i) \sum_{j=1}^{n_{\text{EMP}}} Y_j \times \Delta_{m-r+i,j} \times f_1(e_j) \\
&= \sum_{i=0}^r h(m-r; m-r+i) \alpha_0 \hat{t}_{m-r+i}
\end{aligned} \tag{4}$$

By re-arranging the terms we get the following important recursive relationship:

$$\hat{t}_{m-r} = \frac{\hat{s}_{m-r} - \alpha_0 \sum_{i=1}^r h(m-r; m-r+i) \times \hat{t}_{m-r+i}}{\alpha_0 h(m-r; m-r)} \tag{5}$$

For the base case we obtain:

$$\begin{aligned}
\hat{t}_m &= \frac{\hat{s}_m}{\alpha_0 h(m; m)} \\
&= a_m \times \hat{s}_m
\end{aligned} \tag{6}$$

where $a_m = 1/(\alpha_0 h(m; m))$.

By replacing \hat{s}_{m-r} in the above equations with s_{m-r} which is readily observable from the data and has the same expected value, we can obtain a simple recursive algorithm for computing an unbiased estimator for any t_i . Before presenting the recursive algorithm, we note that we can re-write Equation 5 for \hat{t}_i by replacing \hat{s} with s and by changing the summation variable from i to k and actually substituting $m-r$ by i ,

$$\hat{t}_i = \frac{s_i - \alpha_0 \sum_{k=1}^{m-i} h(i; i+k) \hat{t}_{i+k}}{\alpha_0 h(i; i)}$$

The following pseudo-code then gives the algorithm³ for computing an unbiased estimator for any t_i .

³ Note the $h(m; m)$ probability in line 2 of the `GetEstTi` function. If the sample size from `SALE` is not at least as large as m , then $h(m; m) = 0$ and the `GetEstTi` is undefined. This means that our estimator is undefined if the sample is not at least as large as the largest number of matches for any record from `EMP` in `SALE`. The fact that the estimator is undefined in this case is not surprising, since it means that our estimator does not conflict with known results regarding the existence of an unbiased estimator for the distinct value problem. See the Related Work section for more details.

```

Function GetEstTi(int i)
1 if (i == m)
2     return  $s_m / (\alpha_0 h(m; m))$ 
3 else
4     returnval =  $s_i$ 
5     for (int k = 1; k <= m - i; k++)
6         returnval -=  $\alpha_0 h(i; i + k) \times \text{GetEstTi}(i + k)$ 
7     returnval /=  $\alpha_0 h(i; i)$ 
8     return returnval;
9 }

```

Recall from Equation 1 that the bias of M was expressed as a linear combination of various t_i terms. Using `GetEstTi` to estimate each of the t_i terms, we can write an estimator for the bias of M as:

$$\widehat{Bias}(M) = \sum_{i=1}^m \varphi(n_{\text{SALE}}, n_{\text{SALE}'}, i) \times \text{GetEstTi}(i) \quad (7)$$

In the following two subsections, we present a formal analysis of the statistical properties of our estimator.

3.3 Why Is the Estimator Unbiased?

According to Equation 7, the estimator for the bias of M is composed of a sum of m different estimators. Hence by the linearity of expectation, the expected value of the estimator can be written as:

$$E[\widehat{Bias}(M)] = \sum_{i=1}^m \varphi(n_{\text{SALE}}, n_{\text{SALE}'}, i) \times E[\text{GetEstTi}(i)] \quad (8)$$

The above relation suggests that in order to prove that the sample-based estimator of Equation 7 is unbiased, it would suffice to prove that each of the individual `GetEstTi` estimators is unbiased. We use mathematical induction to prove the correctness of the various estimators on expectation.

As a preliminary step for the proof of unbiasedness, we first derive the expected values of the s_i estimator used by `GetEstTi`. To do this, we introduce a zero/one random variable $H_{j,k}$ that evaluates to 1 if e_j has k matches in `SALE'` and 0 otherwise. The expected value of this variable is simply the probability that it evaluates to 1, giving us $E[H_{j,k}] = h(k; cnt(e_j, \text{SALE}'))$. With this:

$$\begin{aligned}
E[s_k] &= E \left[\sum_{j=1}^{n_{\text{EMP}}} \sum_{i=k}^m Y_j \times \Delta_{i,j} \times H_{j,k} \times f_1(e_j) \right] \\
&= \sum_{j=1}^{n_{\text{EMP}}} \sum_{i=k}^m E[Y_j] \times \Delta_{i,j} \times E[H_{j,k}] \times f_1(e_j) \\
&= \alpha_0 \sum_{j=1}^{n_{\text{EMP}}} \sum_{i=k}^m \Delta_{i,j} \times h(k; i) \times f_1(e_j) \tag{9}
\end{aligned}$$

We are now ready to present a formal proof of unbiasedness of the `GetEstTi`.

Theorem 1 *The expected value of `GetEstTi(i)` is $\sum_{j=1}^{n_{\text{EMP}}} \Delta_{i,j} f_1(e_j)$.*

Proof Using Equation 4, the recursive `GetEstTi` estimator can be re-written as:

$$\text{GetEstTi}(i) = \frac{s_i - \alpha_0 \sum_{k=1}^{m-i} h(i; i+k) \text{GetEstTi}(i+k)}{\alpha_0 h(i; i)} \tag{10}$$

We first prove the unbiasedness for the base case: `GetEstTi(m)`. Setting $i = m$ in the above relation and taking the expectation:

$$E[\text{GetEstTi}(m)] = \frac{E[s_m]}{\alpha_0 h(m; m)}$$

Replacing $E[s_m]$ using Equation 9:

$$\begin{aligned}
E[\text{GetEstTi}(m)] &= \frac{\alpha_0}{\alpha_0 h(m; m)} \sum_{j=1}^{n_{\text{EMP}}} \Delta_{m,j} h(m; m) f_1(e_j) \\
&= \sum_{j=1}^{n_{\text{EMP}}} \Delta_{m,j} f_1(e_j)
\end{aligned}$$

which is actually the value of t_m .

By induction, we can now assume that all estimators `GetEstTi(i+k)` for $1 \leq k \leq m-i$ are unbiased and we use this to prove that the estimator `GetEstTi(i)` is unbiased. Taking the expectation on both sides of the above equation:

$$E[\text{GetEstTi}(i)] = E \left[\frac{s_i - \alpha_0 \sum_{k=1}^{m-i} h(i; i+k) \text{GetEstTi}(i+k)}{\alpha_0 h(i; i)} \right]$$

By the linearity of expectation:

$$= \frac{E[s_i] - \alpha_0 \sum_{k=1}^{m-i} h(i; i+k) E[\text{GetEstTi}(i+k)]}{\alpha_0 h(i; i)}$$

Replacing the values of $E[\text{GetEstTi}(i+k)]$ and $E[s_i]$:

$$\begin{aligned}
&= \frac{1}{\alpha_0 h(i;i)} (\alpha_0 \sum_{j=1}^{n_{\text{EMP}}} \sum_{k=i}^m \Delta_{k,j} h(i;k) f_1(e_j) \\
&\quad - \alpha_0 \sum_{k=1}^{m-i} h(i;i+k) \sum_{j=1}^{n_{\text{EMP}}} \Delta_{i+k,j} f_1(e_j)) \\
&= \frac{1}{h(i;i)} (\sum_{j=1}^{n_{\text{EMP}}} \sum_{k=i}^m \Delta_{k,j} h(i;k) f_1(e_j) \\
&\quad - \sum_{j=1}^{n_{\text{EMP}}} \sum_{k=1}^{m-i} \Delta_{i+k,j} h(i;i+k) f_1(e_j))
\end{aligned}$$

For the second term in the parentheses, replacing $i+k$ by p and changing the limits of summation for the inner sum accordingly:

$$\begin{aligned}
&= \frac{1}{h(i;i)} (\sum_{j=1}^{n_{\text{EMP}}} \sum_{k=i}^m \Delta_{k,j} h(i;k) f_1(e_j) \\
&\quad - \sum_{j=1}^{n_{\text{EMP}}} \sum_{p=i+1}^m \Delta_{p,j} h(i;p) f_1(e_j)) \tag{11}
\end{aligned}$$

We notice that the limits of summation of the inner sum of the first term are from i to m . Splitting this term into two terms such that one term has limits of summation from i to i while the other has limits from $i+1$ to m :

$$\begin{aligned}
&= \frac{1}{h(i;i)} (\sum_{j=1}^{n_{\text{EMP}}} \sum_{k=i}^i \Delta_{k,j} h(i;k) f_1(e_j) \\
&\quad + \sum_{j=1}^{n_{\text{EMP}}} \sum_{k=i+1}^m \Delta_{k,j} h(i;k) f_1(e_j) \\
&\quad - \sum_{j=1}^{n_{\text{EMP}}} \sum_{p=i+1}^m \Delta_{p,j} h(i;p) f_1(e_j)) \\
&= \frac{1}{h(i;i)} \left(\sum_{j=1}^{n_{\text{EMP}}} \Delta_{i,j} h(i;i) f_1(e_j) \right) \\
&= \sum_{j=1}^{n_{\text{EMP}}} \Delta_{i,j} f_1(e_j) \tag{12}
\end{aligned}$$

3.4 Computing the Variance of the Estimator

The unbiased property of $\widehat{Bias}(M)$ means that it may be useful. However, the accuracy of any estimator depends on its variance as well as its bias. We now investigate the variance of our unbiased estimator.

We have seen that $\widehat{Bias}(M)$ is a linear combination of various GetEstTi results with $\varphi(n_{\text{SALE}}, n_{\text{SALE}'}, i)$ as the coefficient of $\text{GetEstTi}(i)$. In order to derive an expression for the variance of the estimator and gain insight about the potential values it can take, we first express the estimator as a linear combination of s_i terms:

$$\widehat{Bias}(M) = \sum_{i=1}^m b_i \times s_i \tag{13}$$

The next step in deriving the variance is being able to compute the various b_i values. Intuitively, the b_i terms can be thought of as coming from the linear relationship between the \hat{t}_i and s_i terms. The following algorithm shows how we can actually compute the b_i values.

```

Function ComputeBis(m)
1 // Let table[m][m] be a 2-dimensional array with all elements initialized to zero
2 for(int row = 0; row < m; row++){
3     for(int term = 1; term <= row; term++){
4         factor = -h(m - row; m - row + term)/h(m - row; m - row)
5         pro = row - term
6         for(int pcol = 0; pcol <= pro; pcol++){
7             table[row][pcol] += factor * table[pro][pcol]
8         }
9         table[row][row] = 1/h(m - row; m - row)
10}
11for(int row = 0; row < m; row++)
12     for(int col = 0; col <= row; col++)
13         bm-col +=  $\alpha_0$  * h(0, m - row) * table[row][col]

```

With this, the variance of this estimator can then be written as:

$$\text{Var}(\widehat{\text{Bias}}(M)) = \text{Var}\left(\sum_{i=1}^m b_i s_i\right) \quad (14)$$

Note that the s_i values are not independent random variables since if an EMP' record has i matches in SALE', then it cannot have j matches in SALE'. Hence we have:

$$\text{Var}(\widehat{\text{Bias}}(M)) = \sum_{i=1}^m b_i^2 \text{Var}(s_i) + 2 \sum_{i=1}^m \sum_{i < j \leq m} b_i b_j \text{Cov}(s_i, s_j) \quad (15)$$

The Var and Cov terms can be computed by using the standard formulas:

$$\text{Var}(s_i) = E[s_i^2] - E^2[s_i]; \quad \text{Cov}(s_i s_j) = E[s_i s_j] - E[s_i]E[s_j] \quad (16)$$

To evaluate $\text{Var}(s_i)$ and $\text{Cov}(s_i s_j)$, $E[s_i]$ can be computed using Equation 8. $E[s_i^2]$ and $E[s_i s_j]$ can be computed as follows:

$$\begin{aligned} E[s_i s_j] &= E \left[\left(\sum_{k=1}^{n_{\text{EMP}}} Y_k f_1(e_k) H_{k,i} \right) \left(\sum_{r=1}^{n_{\text{EMP}}} Y_r f_1(e_r) H_{r,j} \right) \right] \\ &= E \left[\sum_{k=1}^{n_{\text{EMP}}} \sum_{r=1}^{n_{\text{EMP}}} Y_k Y_r H_{k,i} H_{r,j} f_1(e_j) f_1(e_r) \right] \\ &= \sum_{k=1}^{n_{\text{EMP}}} \sum_{r=1}^{n_{\text{EMP}}} E[Y_k Y_r] E[H_{k,i} H_{r,j}] f_1(e_j) f_1(e_r) \end{aligned} \quad (17)$$

The above expression can be evaluated using the following rules:

- if $k \neq r$ (that is, e_k and e_r are two different tuples) then $E[H_{k,i} H_{r,j}] \approx h(i, \text{cnt}(e_k, \text{SALE})) h(j, \text{cnt}(e_r, \text{SALE}))$ if we assume that no record s exists in SALE where $f_3(e_k, s) = f_3(e_r, s) = \text{true}$

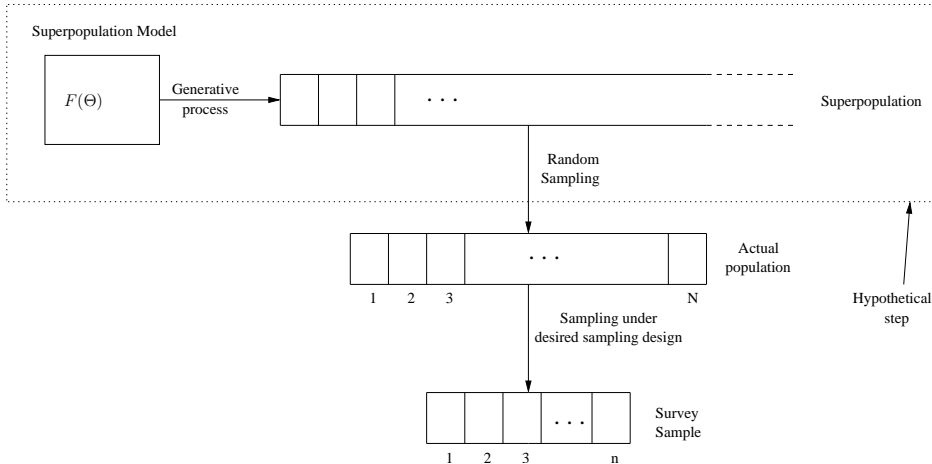


Fig. 1 Sampling from a superpopulation

- if $i = j$ (that is, we are computing $E[s_i^2]$) and $k = r$, then $E[H_{k,i}H_{r,j}] = h(i, cnt(e_k, \text{SALE}))$
- if $i \neq j$ (that is, we are computing $E[s_i s_j]$) and $k = r$, then $E[H_{k,i}H_{r,j}] = 0$ since a record cannot have two different numbers of matches in a sample
- if $k = r$, then $E[Y_k Y_r] = \alpha_0$
- if $k \neq r$, then $E[Y_k Y_r] \approx \alpha_0^2$

3.5 Is This Good?

At this point, we now have a simple, unbiased estimator for the answer to a subset-based query, as well as a formal analysis of the statistical properties of the estimator. However, there are two problems related to the variance that may limit the utility of the estimator.

First, in order to evaluate the hypergeometric probabilities needed to compute or estimate the variance, we need the value of $cnt(e, \text{SALE})$ for an arbitrary record e of EMP. This information is generally unavailable during sampling, and it seems difficult or impossible to obtain a good estimate for the appropriate probability without having this information. This means that in practice, it will be difficult or impossible to tell a user how accurate the resulting estimate is likely to be. We have experimented with general-purpose methods such as the bootstrap [27] to estimate this variance, but have found that these methods often do an extremely poor job in practice.

Second, the variance of the estimator itself may be huge. The b_i coefficients are composed of sums, products and ratios of hypergeometric probabilities which can result in huge values. Particularly worrisome is the $h(i, i)$ value in the denominator used by `GetEstTi`. Such probabilities can be tiny; including such a small value in the denominator of an expression results in a very large value that may “pump up” the variance accordingly.

4 Developing a Biased Estimator

In light of these problems, in this section we describe a biased estimator that is often far more accurate than the unbiased one, and also provides the user with an idea of the estimation accuracy. Just like the unbiased estimator $M - \widehat{Bias}(M)$ from the previous section, our biased estimator will be nothing more than a weighted sum over the observed s_k values. However, the weights will be chosen so as to minimize the expected or mean-squared error of the resulting estimator.

To develop our biased estimator, we make use of the “superpopulation modeling” approach from statistics [26]. One simple way to think of a superpopulation is that it is an infinitely large set of records from which the original data set has been obtained by random sampling. Because the superpopulation is infinite, it is specified using a parametric distribution, which is usually referred to as the *prior distribution*.

Using a superpopulation method, we imagine the following two-step process is used to produce our sample:

1. Draw a large sample of size N from an imaginary infinite superpopulation where N is the data set size.
2. Draw a sample of size $n < N$ without replacement from the large sample of size N obtained in Step 1 where n is the desired sample size.

By characterizing the superpopulation, it is possible to design an estimator that tends to perform well on any data set and sample obtained using the process above.

The following steps outline a road-map of our superpopulation-based approach for obtaining a high-quality biased estimator for a subset-based query. We describe each step in detail in the next section.

1. Postulate a superpopulation model F for our data set (F is the prior distribution; we use the notation p_F to denote the probability density function (PDF) of F). In general, F is parameterized on a parameter set Θ .
2. Infer the most likely values of the parameter set Θ from EMP' and SALE' . Since we do not have the complete data, but rather a random sample of the data, this is a difficult problem. We make use of an Expectation-Maximization (EM) algorithm to learn the model parameters.
3. Use $F(\Theta)$ to generate d different populations P_1, \dots, P_d , where each $P_i = (\text{EMP}_i, \text{SALE}_i)$. Note that if the data set in question is large, this may be very expensive. We show that for our problem it is not necessary to generate the actual populations – it is enough to obtain certain sufficient statistics for each of them, which can be done efficiently.
4. Sample from each P_i to obtain d sample pairs of the form $S_i = (\text{EMP}'_i, \text{SALE}'_i)$. Again, this can be done without actually materializing the samples.
5. Let $q(P_i)$ be the query answer over the i th data set. Construct a weighted estimator W that minimizes $\sum_{i=1}^d (q(P_i) - W(S_i))^2$.
6. Use W on the original samples EMP' and SALE' to obtain the final estimate to the NOT EXISTS query. The MSE of this estimate can gener-

ally be assumed to be the MSE over all of the populations generated:
 $\sum_{i=1}^d (1/d) \times (q(P_i) - W(S_i))^2$.

5 Details of Our Approach

In this section, we discuss in detail each of the steps outlined above of our approach of obtaining an optimal weighted estimator for the NOT EXISTS query.

5.1 Choice of Model and Model Parameters

The first task is to define a generative model and an associated probability density function for the two relations EMP and SALE respectively. While this may seem like a daunting task (and a potentially impossible one given all of the intricacies of modeling real-life data) it is made easy by the fact that we only need to define a model that can realistically reproduce those characteristics of EMP and SALE that may affect the bias or variance of an estimator for a subset-based query. From the material in Section 3 of the paper, for a given record e from EMP, we know that these three characteristics are:

1. $f_1(e)$
2. $\text{cnt}(e, \text{SALE})$, which is the number of SALE records s for which $f_3(e, s)$ is *true*
3. $\text{cnt}(e, e', \text{SALE})$ where $e' \neq e$, which is the number of SALE records s for which $f_3(e, s) \wedge f_3(e', s)$ is *true*

To simplify our task, we will actually ignore the third characteristic and define a model such that this count is always zero for any given record pair. While this may introduce some inaccuracy into our method, it still captures a large number of real-life situations. For example, if f_3 consists of an equality check on a foreign key from SALE into EMP (which is arguably the most common example of such a subset-based query) then two records from EMP can never match with the same record from SALE and this count is always zero.

Given that our model needs to be able to generate instances of EMP and SALE that realistically model the first two aspects given above, we choose the parameter set $\Theta = \{\mathbf{p}, \boldsymbol{\mu}, \sigma^2\}$ where:

- \mathbf{p} is a vector of probabilities, where p_i represents the probability that any arbitrary record of EMP belongs to class i
- $\boldsymbol{\mu}$ is a vector of means, where μ_i represents the mean aggregate value of all records belonging to class i .
- σ^2 is the variance of $f_1(e)$ over all records $e \in \text{EMP}$.

Then given these parameters, EMP and SALE are generated using our model as follows:

Procedure GenData

- 1 For $rec = 1$ to n_{EMP} do
 - 2 Randomly generate k between 0 and m such that for any
 $i; 0 \leq i \leq m, Pr[i] = p_i$
 - 3 Generate a value for $f_1(e)$ by sampling from $N(\mu_k, \sigma)$
 - 4 Add the resulting e to EMP
 - 5 For $j = 1$ to k do
 - 6 Generate a record s where $f_3(e, s)$ is true
 - 7 Add s to SALE
-

In step (3), N is a normally distributed random variable with the specified mean and variance. We use a normal random variable because we are interested in sums over classes in EMP; due to the central limit theorem (CLT), these sums will be normally distributed for a large database. Thus, using a normal random variable does not result in loss of generality. Also, note that in step (6), according to our earlier assumption $f_3(e', s) = false, \forall e' \neq e$.

In our actual model, the various μ_i values are not assumed to be independent but we rather assume a linear relationship between them to limit the degrees of freedom of the model and thus avoid the problem of overfitting the model (see Sec. 5.2). In our model the various μ_i values are related as $\mu_i = s \times i + \mu_0$, where s and μ_0 are the only two parameters that need to be learned to determine all the μ_i . Also in order to avoid overfitting, we assume that σ^2 is the variance of $f_1(e)$ over all records, rather than modeling and learning variance values of all the individual classes separately.

We now define the density function for the superpopulation model corresponding to the *GenData* algorithm. For a given EMP record e , if $f_1(e) = v$ and $cnt(e, SALE) = k$ the probability density for e given a parameter set Θ is given by:

$$p(e|\Theta) = p(v, k|\Theta) = p_k p_N(\mu_k, \sigma, v) \quad (18)$$

Where it is convenient, we will use the notation $p(v, k|\Theta)$ for values v and k and $p(e|\Theta)$ for record e interchangeably. In this expression, p_N is the PDF for the normal distribution evaluated at v and is given by:

$$p_N(\mu_k, \sigma, v) = \frac{e^{-(v-\mu_k)^2/2\sigma^2}}{\sigma\sqrt{2\pi}} \quad (19)$$

Then if we consider a given data set $\{EMP, SALE\}$, the probability density of the data set is simply the product of the densities of all the individual records:

$$p_F(EMP, SALE) = \prod_{e \in EMP} p(e|\Theta) \quad (20)$$

A Note on The Generality of the Model. As described, our model is extremely general, making almost no assumptions about the data other than the fact that $f_1(e)$ values are normally distributed. This is actually an inconsequential assumption anyway, since we are interested in sums over

$f_1(e)$ values which will be normally distributed whatever the distribution of $f_1(e)$ due to the CLT.

On one hand, this generality can be seen as a benefit of the approach: it makes use of very few assumptions about the data. Most significant is the lack of any sort of restriction on the probability vector \mathbf{p} . The result is that the number of records from SALE matching a certain record from EMP is multinomially distributed. On the other hand, a Bayesian argument [11] can be made that such extreme freedom is actually a poor choice, and that in "real-life", an analyst will have some sort of idea what the various p_i values look like, and a more restrictive distribution providing fewer degrees of freedom should be used. For example, a negative binomial distribution has been assumed for the distinct value estimation problem [32]. Such background knowledge could certainly improve the accuracy of the method.

Though we eschew any such restrictions in the remainder of the paper (except for an assumption of a linear relationship among the μ_i values; see "Dealing with Over-fitting" in the next section), we note that it *would* be very easy to incorporate such knowledge into our method. The only change needed is that the EM algorithm described in the next section would need to be modified to incorporate any constraints induced on the various parameters by additional distributional assumptions.

5.2 Estimation of Model Parameters

Now that we have defined our superpopulation model, we need access to the parameter set Θ that was used to create our particular instances of EMP and SALE in order to develop an estimator that performs well for the resulting superpopulation. However, we have several difficulties. First, we do not know Θ ; since EMP and SALE are in reality not sampled from any parametric distribution, Θ does not even exist. We could compute a maximum-likelihood estimate (MLE)⁴ to choose a Θ that optimally fits EMP and SALE, but then we have an even bigger problem: we do not even have access to EMP and SALE; we only have access to samples from them. Thus, we need a way to infer Θ by looking only at the samples EMP' and SALE'.

It turns out that we can still make use of an MLE. Since EMP' may be treated as a set of independent, identically distributed samples from F , if we simply replace EMP with EMP' as an argument to p_F , then by choosing Θ so as to maximize p_F , we will still produce exactly the same estimate for Θ on expectation that we would have if EMP were used instead. Thus, we can essentially ignore the distinction between EMP and EMP'. However, the same argument does not hold for SALE because without access to all of SALE, we cannot compute $k = \text{cnt}(e, \text{SALE})$ for arbitrary e in order to apply an MLE.

To handle this, we will modify our PDF slightly to also take into account the sampling from SALE. This can easily be done by modifying the function $p(v, k|\Theta)$. To simplify the modification, we ignore the fact that the number of

⁴ An MLE is a standard statistical estimator for unknown model parameters when a sample is available; the MLE simply chooses Θ so as to maximize the value of the PDF of the sample.

such records s from SALE' where $f_3(e_1, s)$ is *true* may be correlated with the number of records from SALE' where $f_3(e_2, s)$ is *true* for arbitrary records e_1 and e_2 from EMP ; that is, we assume that we are looking for matches of a record e in its own “private” sample from SALE and that all of these samplings are independent. With this, if $f_1(e) = v$ and $\text{cnt}(e, \text{SALE}) = k$ and $\text{cnt}(e, \text{SALE}') = k'$ then:

$$p(v, k, k'|\Theta) = p(v, k|\Theta)h(k'; k) \quad (21)$$

In this expression, h is the hypergeometric probability of seeing k' matches for e in SALE' , given that there were k matches in SALE .

Since the portion of SALE that is not in SALE' is hidden to us due to the sampling, we do not know k and we have a classic example of an MLE problem with hidden or missing data. There are several methods from the literature for solving such a problem; the one that we employ is the Expectation Maximization (EM) algorithm.

The EM algorithm [22] is a general method of finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data set when the data is incomplete or has missing values. EM starts out with an initial assignment of values for the unknown parameters and at each step, recomputes new values for each of the parameters via a set of update rules. EM continues this process until the likelihood stops increasing any further. Since $\text{cnt}(e, \text{SALE})$ is unknown, the likelihood function:

$$L(\Theta|\{\text{EMP}', \text{SALE}'\}) = \prod_{e \in \text{EMP}'} \sum_{k=1}^m p(f_1(e), k, \text{cnt}(e, \text{SALE}')|\Theta)$$

We present the derivation of our EM implementation in Appendix A, while here we give only the algorithm. In this algorithm, $\tilde{p}(i|\Theta, e)$ denotes the posterior probability for record e belonging to class i . This is the probability that given the current set of values for Θ , record e belongs to class i .

Procedure EM(Θ)

```

1 Initialize all parameters of  $\Theta$ ;  $L_{prev} = -9999$ 
2 while (true){
3   Compute  $L(\Theta)$  from the sample and assign it to  $L_{curr}$ 
4   if  $((L_{curr} - L_{prev})/L_{prev} < 0.01)$  break
5   Compute posterior probabilities for each  $e \in \text{EMP}'$  and each  $k$ 
6   Recompute all parameters of  $\Theta$  by using the following
   update rules:
7    $\mu_i = \frac{\sum_{e \in \text{EMP}'} \tilde{p}(i|\Theta', e) \times f_1(e)}{\sum_{e \in \text{EMP}'} \tilde{p}(i|\Theta', e)}$ 
8    $\sigma^2 = \frac{\sum_{e \in \text{EMP}'} \sum_{j=0}^m \tilde{p}(j|\Theta', e) (f_1(e) - \mu_j)^2}{\sum_{e \in \text{EMP}'} \sum_{j=0}^m \tilde{p}(j|\Theta', e)}$ 
9    $p_i = \frac{\sum_{e \in \text{EMP}'} \tilde{p}(i|\Theta', e)}{\sum_{e \in \text{EMP}'} \sum_{j=1}^m \tilde{p}(j|\Theta', e)}$ 
10   $L_{prev} = L_{curr}$ 
11 }
12 Return values in  $\Theta$  as the final parameters of the model
```

Every iteration of the EM algorithm performs an expectation (E) step and a maximization (M) step. In our algorithm, the E-step is contained in step (6) where for each record e of EMP', a set of probability values $\tilde{p}(i|\Theta, e), 0 \leq i \leq m$, is computed under the current model parameters, Θ . The posterior probability $\tilde{p}(i|\Theta, e)$ is computed as described in the Appendix. Intuitively, the posterior probability for record e and class i is a ratio of two quantities: (1) the probability that e belongs to class i according to the density function of the model, and (2) the sum of probabilities that it belongs to each of the classes 0 through m , also according to the model density function.

The M-step (which corresponds to steps (7) - (10) of our algorithm) updates the parameters of our model in such a way that the expected value of the likelihood function associated with the model is maximized with respect to the posterior probabilities. Details of how we obtain the various update rules are explained in Appendix A.

The observant reader may note that the EM algorithm assumes that the parameter m is known before the process has begun. This is potentially a problem, since m will typically be unknown. Fortunately, knowing the exact value for m is not vital, particularly if m is overestimated (in which case the class probabilities associated with the class i records for large i will end up being zero, if the EM algorithm functions correctly). As a rough estimate for m , we take the record from EMP' with the largest number of matches in SALE' and scale up the number of matches by $n_{\text{SALE}}/n_{\text{SALE}'}$. Particularly if several records with m matches in SALE are expected to appear in EMP', this estimate for m will be quite conservative.

Dealing with Over-fitting. The superpopulation model has a total of $2(m+1)+1$ parameters within Θ . Since the number of degrees of freedom of the model is so large, the model has a tremendous leeway when choosing parameter values. This potentially leads to a well-known drawback of learned models – over-fitting the training data, where the model is tailored to be excessively well-suited to the training data at the cost of generality.

Several techniques have been proposed to address the over-fitting problem [30]. We use the following two methods in our approach:

- Limiting the number of degrees of freedom of the model.
- Using multiple models and combining them to develop our final estimator.

To use the first technique, we restrict our generative model so that the mean aggregate value of all records of any class i is not independent of the mean value of other classes. Rather, we use a simple linear regression model $\mu_i = s \times i + \mu_0$. s and μ_0 are the two parameters of the linear regression model and can be learned easily. This means that once we have learned the two parameters s and μ_0 , the μ_i values for all other classes can be determined directly by the above relation and will not be learned separately. As mentioned previously, it would also be possible to place distributional constraints upon the vector \mathbf{p} in order to reduce the degrees of freedom even more, though we choose not to do this in our implementation.

Our second strategy to tackle the over-fitting problem is to learn multiple models rather than working with a single model. These models differ from each other only in that they are learned using our EM algorithm with different

initial random settings for their parameters. When generating populations from the models learned via EM (as described in the next subsection), we then rotate through the various models in round-robin fashion.

Are we not done yet? Once the model has been learned, a simple estimator is immediately available to us: we could return $p_0 \times \mu_0 \times n_{\text{EMP}}$, since this will be the expected query result over an arbitrary database sampled from the model. This is equivalent to first determining a class of databases that the database in question has been randomly selected from, and then returning the average query result over all of those databases. If multiple models are learned in order to alleviate the over-fitting problem, then we can use the average of this expression over all of those models.

While this estimator is certainly reasonable, the concern is twofold. First, if there is high variability in the possible populations that could be produced by the model or models (corresponding to uncertainty in the correctness of the model), then simply taking the average of all of these populations will expectedly result in an answer with high variance. A related concern is that this is not very robust to errors in the model-learning process – an error in the model will lead directly to an error in the estimate.

Thus, in the next few subsections we detail a process that attempts to simultaneously perform well on *any* and *all* of the databases that could be sampled from the model, rather than simply returning the mean answer over all potential databases. The method samples a large number of $((\text{EMP}_i, \text{SALES}_i), (\text{EMP}'_i, \text{SALES}'_i))$ combinations from the model, and then attempts to construct an estimator that can accurately infer the query answer over precisely the $(\text{EMP}_i, \text{SALES}_i)$ that has been sampled by looking at $(\text{EMP}'_i, \text{SALES}'_i)$.

5.3 Generating Populations From the Model

Once we know the parameter set Θ , the next task is to generate many instances of $P_i = (\text{EMP}_i, \text{SALE}_i)$ and $S_i = (\text{EMP}'_i, \text{SALE}'_i)$ in order to optimize our biased estimator over these population-sample pairs. The difficulty is that in practice, EMP and SALE can have billions of records in them. Hence, it would not be feasible to actually materialize each (P_i, S_i) pair. The good news is that for our problem it is not necessary to actually generate the populations if we can generate statistics associated with the pair that are sufficient to optimize our biased estimator.

Computing sufficient statistics for EMP and SALE. For each P_i , we must generate the following statistics:

- The number of records of EMP belonging to each class (we use n_i to denote this).
- The mean over f_1 for all records belonging to each class.

The first set of statistics are easy to generate if we notice that the number of records belonging to each class is simply a multinomial distribution with n_{EMP} trials and each multinomial bucket probability is given by the vector \mathbf{p} . A

single, vector-valued sample from an appropriately-distributed multinomial distribution can then give us each n_i .

The next set of statistics can be computed by relying on the CLT. According to the generative model, the aggregate attribute value of records of the superpopulation belonging to class i is given by μ_i with variance of σ^2 . Since the population is an i.i.d. random sample from the superpopulation, the mean aggregate value of records belonging to class i follows a normal distribution with mean given by μ_i and variance of σ^2/n_i . Thus, t_i which is the sum over the aggregate attribute of all records of class i can then be obtained by drawing a trial from the normal distribution $N(\mu_i, \sigma^2/n_i)$ and multiplying it by n_i .

Computing sufficient statistics for EMP' and SALE'. For each S_i , we must generate the following statistics:

- The number of sampled records from each class of EMP; this is denoted by n'_i .
- The number of sampled records from the i th class of EMP that have j matches in SALE' for each i and j . We denote this by $n'_{i,j}$.
- The mean over f_1 corresponding to each $n'_{i,j}$.

The first set of statistics can be produced by repeatedly sampling from a hypergeometric distribution. To compute n'_0 , we sample from a hypergeometric distribution with parameters n_{EMP} , $n_{\text{EMP}'}$, and n_0 (these parameters are the population size, the sample size, and the size of the subpopulation of interest, respectively). To compute n'_1 , we sample from a hypergeometric distribution with parameters $n_{\text{EMP}} - n_0$, $n'_{\text{EMP}} - n'_0$, and n_1 . n'_2 is a sample from a hypergeometric distribution with parameters $n_{\text{EMP}} - (n_0 + n_1)$, $n'_{\text{EMP}} - (n'_0 + n'_1)$, and n_2 . This process is repeated for each n'_i .

Once each n'_i is generated, each $n'_{i,j}$ is generated. In order to speed the process of generating each $n'_{i,j}$, we can assume that the expected value of each $n'_{i,j}$ is small compared to n_{SALE} , so that there is little difference between sampling with and without replacement. Thus, we can assume that each $n'_{i,j}; j \leq i$ is binomially distributed which in turn means that all $n'_{i,j}$ are multinomially distributed, where the probability that any class i record will have j matches in the sample SALE' is a hypergeometric probability denoted by $h(j; i)$. A single trial over a multinomial random variable having probabilities of $h(j; i)$ for j from 0 to i will then give us each $n'_{i,j}$ for a given i .

Finally, again using a CLT-based argument, the mean over f_1 for all of the records corresponding to each $n'_{i,j}$ is generated by a single trial over a normal random variable $N(\mu_i, \sigma^2/n'_{i,j})$.

5.4 Constructing the Estimator

We have seen in the previous subsection that once a model has been learned, it can be used to generate statistics for any number of population(s)/sample(s). Recall from Section 4 that the j th population generated and the sample from

that population are $P_j = (\text{EMP}_j, \text{SALE}_j)$ and $S_j = (\text{EMP}'_j, \text{SALE}'_j)$, respectively. Let s_{ij} be the value of s_i computed over S_j ; that is, it is the sum for f_1 over all tuples in EMP'_j that have i matches in SALE'_j . Our goal in all of this is to construct a weighted estimator:

$$W(S_j) = \sum_{i=0}^m w_i s_{ij} \quad (22)$$

that minimizes:

$$SSE = \sum_j (W(S_j) - q(P_j))^2 = \sum_j \left(\sum_{i=0}^m w_i s_{ij} - q(P_j) \right)^2 \quad (23)$$

where $q(P_j)$ is the answer to the NOT EXISTS query over the j th population.

W should be optimized by choosing each w_i so as to minimize the SSE (sum-squared-error) given above. In order to compute these weights we evaluate the partial derivative of the SSE w.r.t each of the unknown weights. For example, by taking the partial derivative of the SSE w.r.t w_0 , we obtain:

$$\frac{\partial SSE}{\partial w_0} = \sum_j 2 \left(\sum_{i=0}^m w_i s_{ij} - q(P_j) \right) (s_{0j})$$

If we differentiate with respect to each w_i and set the resulting $m + 1$ expressions to zero, we obtain $m + 1$ linear equations in the $m + 1$ unknown weights. These equations can be represented in the following matrix form:

$$\begin{bmatrix} \sum_j s_{0j}^2 & \sum_j s_{0j} s_{1j} & \cdots \\ \sum_j s_{0j} s_{1j} & \sum_j s_{1j}^2 & \cdots \\ \vdots & \vdots & \ddots \\ \sum_j s_{0j} s_{mj} & \sum_j s_{1j} s_{mj} & \cdots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} \sum_j s_{0j} q(P_j) \\ \sum_j s_{1j} q(P_j) \\ \vdots \\ \sum_j s_{mj} q(P_j) \end{bmatrix}$$

The optimal weights can then be easily obtained by using a linear equation solver to solve the above system of equations.

Once W has been derived, it is then applied to the original samples EMP' and SALES' in order to estimate the answer to the query. By dividing the SSE obtained via the minimization problem described above by the number of data sets generated, we can also obtain a reasonable estimate of the mean-squared error of W .

6 Experiments

In this section we describe results of the experiments we performed to test our estimators. Our experiments are designed to test the accuracy of our estimators and the running time of the biased estimator, over a wide variety of data sets.

6.1 Experimental Setup

In this subsection, we describe the properties of the various data sets we use to test our estimators. We generate 66 synthetic data sets and use three real-life data sets for conducting our experiments. All our experiments were performed on a Linux workstation having 1 GB of RAM and a 2.4 GHz clock speed and all software was implemented using the C++ programming language.

6.1.1 Synthetic Data Sets

In each data set, we have two relations, `EMP(EID, AGE, SAL)` and `SALE(SALEID, EID, AMOUNT)` of size 10 million and 50 million records, respectively. We evaluate the following SQL query over each data set:

```
SELECT SUM (e.SAL)
FROM EMP as e
WHERE NOT EXISTS
(SELECT * FROM SALE AS s
 WHERE s.EID = e.EID)
```

Two important data set properties that affect the query result are:

1. The distribution of the number of matching records in `SALE` for each record of `EMP`
2. The distribution of `e.SAL` values of all records of `EMP`

Based on these two important properties, we synthetically generated data sets so that the distribution of the number of matching records for all `EMP` records follows a discretized Gamma distribution. The Gamma distribution was chosen because it produces positive numbers and is very flexible, allowing a long tail to the right. This means that it is possible to create data sets for which most records in `EMP` have very few matches, but some have a large number. We chose values of 1, 2 and 5 for the Gamma distribution's shift parameter and values of 0.5 and 1 for the scale parameter. Based on these different values for the shift and scale parameters, we obtained six possible data sets: 1: (shift = 1, scale = 0.5); 2: (shift = 2, scale = 0.5); 3: (shift = 5, scale = 0.5); 4: (shift = 1, scale = 1); 5: (shift = 2, scale = 1); and 6: (shift = 5, scale = 1). For these six data sets, the fraction of `EMP` records having no matches in `SALE` (and thus contributing to the query answer) were .86, .59, .052, .63, .27, and .0037, respectively. A plot of the probability that an arbitrary tuple from `EMP` has m matches in `SALE` for each of the six data sets is given as Figure 2. This shows the wide variety of data set characteristics we tested.

We also varied the distribution of the `e.SAL` values such that the distribution can be one of the following:

- a. Normally distributed with a mean of 100 and standard deviation of 10
- b. Normally distributed with a mean of 100 and standard deviation of 200, with only the absolute values considered

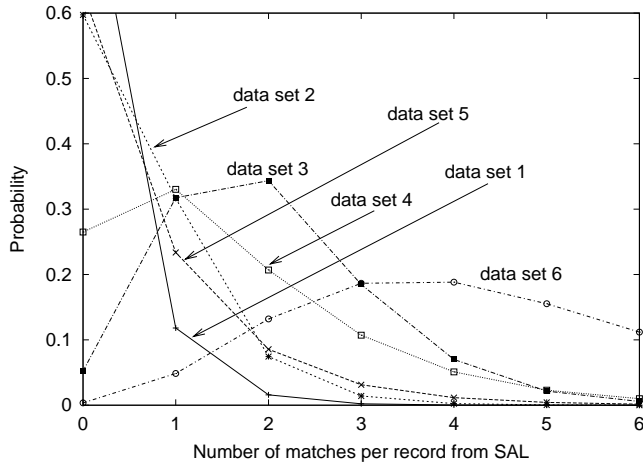


Fig. 2 Six distributions used to generate for each e in EMP the number of records s in SALE for which $f_3(e, s)$ evaluates to *true*.

- c. Zipfian distributed with a skew parameter of 0.5
- d. Zipfian distributed with a skew parameter of 1.0

We doubled the number of data sets by further providing a linear positive correlation or no correlation between the e .SAL value of a record and the number of matching records it has in SALE. We thus obtained 48 different data sets considering all possible combinations of the distribution of matching records and the distribution of e .SAL values.

We also tested our estimator on 18 additional synthetic data sets that were deliberately designed to have properties that violate the assumptions of the superpopulation model of our biased estimator, so as to see how robust this estimator is to inaccuracies in the parametric model. From Section 5.1, the three specific assumptions we made for our superpopulation model were:

1. $\text{cnt}(e, e', \text{SALE}) = 0$ when $e' \neq e$. Thus, the number of SALE records s for which $f_3(e, s) \wedge f_3(e', s)$ is *true* is zero. In other words, different records from EMP do not “share” matching records in SALE.
2. There exists a linear relationship between the mean aggregate values of the different classes of EMP records given by $\mu_i = s \times i + \mu_0$ where s is the slope of the straight line connecting the various μ_i values.
3. The variance of the aggregate attribute values of records of any class is approximately equal to the single model parameter σ^2 .

For each of these three cases, we generate six different data sets using the six different sets of gamma parameters described earlier. Thus we obtain 18 more data sets where the first six sets violate assumption 1, the next six sets violate assumption 2 and the last six sets violate assumption 3. For each of these 18 data sets, the aggregate attribute value is normally distributed with a mean of 100 and standard deviation of 200 except for the last six sets where different values of standard deviation are chosen for records from different classes.

In order to violate assumption 1, we no longer assume a primary key-foreign key relationship between EMP and SALE. To generate a data set violating this assumption, a set s_1 of records of size 100 from EMP is selected. Let max be the largest number of matches in SALE for any record from s_1 . Then an associated set s_2 of max records is added to SALE such that all records in s_1 have their matching records in s_2 . Assumption 2 was violated using $\mu_i = s \times j + \mu_0$, where $j \neq i$ (in fact, the j value for a given i is randomly selected from $1\dots m$). Assumption 3 was violated by assuming different values for the variance of records from different classes. We randomly chose these values from the range (100, 15000).

6.1.2 Real-life Data Sets

The three real-life data sets we use in our experiments are from the Internet Movie Database (IMDB) [1], the Synoptic Cloud Reports [2] obtained from the Oak Ridge National Laboratory, and the network connections data sets from the 1999 KDDCup event.

The IMDB database contains several relations with information about movies, actors and production studios. For our experiments, we use the two relations `MovieBusiness` and `MovieGoofs`. `MovieBusiness` contains information about box-office revenues of movies while `MovieGoofs` contains records that describe unintended mistakes or “goofs” in various movies. The following schema shows the relevant attributes of the two relations for the queries we tested in our experiments.

```
MovieBusiness (MovieName, NumAdmissions)
MovieGoofs (GoofId, MovieName)
```

`MovieName` is the primary key of `MovieBusiness` and a foreign key of `MovieGoofs`.

We tested the following three SQL queries on the two relations of the IMDB dataset.

```
Q1: SELECT SUM (b.NumAdmissions)
      FROM MovieBusiness as b
      WHERE NOT EXISTS
      (SELECT * FROM MovieGoofs AS g
       WHERE g.MovieName = b.MovieName)

Q2: SELECT SUM (b.NumAdmissions)
      FROM MovieBusiness as b
      WHERE NOT EXISTS
      (SELECT * FROM MovieBusiness AS b2
       WHERE id(b) < id(b2)
       AND b.NumAdmissions = b2.NumAdmissions)

Q3: SELECT COUNT (*)
      FROM MovieBusiness as b
      WHERE NOT EXISTS
```

```
(SELECT * FROM MovieGoofs AS g
WHERE g.MovieName = b.MovieName)
```

The second real-life data set we use is the Synoptic Cloud Report (SCR) data set. It contains weather reports for a 10-year period obtained from measuring stations on land as well as water. We use weather reports for the months of December 1981 and November 1991 from measuring stations on land. Specifically, the two relations and their relevant schema used in our experiments are:

```
DEC81 (Id, Latitude, CloudAmount)
NOV91 (Id, Latitude, CloudAmount)
```

Here, *Id* is the key in both the relations. We tested the following two SQL queries on the relations DEC81 and NOV91.

```
Q4: SELECT SUM (D81.CloudAmount)
FROM DEC81 as D81
WHERE NOT EXISTS
(SELECT * FROM NOV91 AS N91
WHERE N91.Latitude = D81.Latitude)
```

```
Q5: SELECT COUNT (*)
FROM DEC81 as D81
WHERE NOT EXISTS
(SELECT * FROM NOV91 AS N91
WHERE N91.Latitude = D81.Latitude)
```

The KDDCup data set contains information about various network connections that can potentially be used for intrusion detection. This data set has 42 integer, real-valued, and categorical attributes. We tested our estimator on this data set by estimating the total number of source bytes of connections that were “significantly different” from the rest of the network connections. That is, we summed the total number of source bytes created by outlier connections. Our definition of “significantly different” records is those records whose distance from all other records in the data set is greater than some predefined threshold. For our experiments, we use a simple distance function that uses Euclidean distance for numerical attributes and a 0/1 distance for categorical attributes.

We execute the following query on the KDDCup data set for our experiments.

```
SELECT SUM (kc1.SourceBytes)
FROM KDDCup as kc1
WHERE NOT EXISTS
(SELECT * FROM KDDCup AS kc2
WHERE  $d(kc1, kc2) < d_{threshold}$ )
```

By choosing different values for $d_{threshold}$, we can control the selectivity of

the above query. For our experiments, we define Q_6 , Q_7 and Q_8 as three variants of the above query with different values of $d_{threshold}$ so that Q_6 has a selectivity of around 24%, Q_7 has a selectivity of 1.75% while Q_8 has a selectivity of 0.4%.

6.2 Results

We ran our experiments on 1%, 5% and 10% random samples of the data sets (both relations in each data set were sampled independently without replacement at the same rate). Both the biased estimator and the unbiased estimator were run ten times on each of the test cases. For comparison we also analytically compute the standard error for the concurrent estimator described in Section 2. Results from the first 48 synthetic data sets are given in tabular form in Figure 3 while results from the next 18 synthetic data sets (which specifically violate the model assumptions) are presented in Figure 4. Real-life data set results are shown in Figure 5. For each of the test cases, we give the square root of the observed mean-squared error (that is, the *standard error*) for the biased, unbiased as well as concurrent estimator. Because having an absolute value for the standard error lacks any sort of scale and thus would not be informative, we give the standard error as a percentage of the total aggregate value of all records in the database. For example, for the synthetic data sets, we give the standard error as a percentage of the answer to the query:

```
SELECT SUM (e.SAL)
FROM EMP as e
```

Thus, if the estimation method simply returned zero every time, its error would vary between 0% and 100%, depending on the selectivity of the sub-query. If the method is also able to estimate with high accuracy which of the constituent records should not be counted in the aggregate total, then the error can be reduced to an arbitrarily small level.

Although our error metric is different from the relative error (which takes the ratio of the absolute error with the true query answer), the value of the relative error could be readily computed from the error value given by dividing by the ratio of the query answer and the total aggregate value of all records in the outer relation. For all the eight cases of data set 1, the query answer is approximately 86% of the total answer. Hence, the relative error is about 1.1 times the error reported in Figure 3. Similarly for the rest of the data sets, the factors are: data set 2: 1.7; data set 3: 19; data set 4: 1.5; data set 5: 3.7 and data set 6: 270. For the IMDB and SCR data sets, the factors are between 1 and 5.5 while for the KDDCup the factors range from 2 (for the high selectivity query) to 40 (for the very low selectivity query).

When we tested the queries, we also recorded the number of times (out of ten) that the answer given by the biased estimator was within ± 2 estimated standard errors of the real answer to the query and found that for almost all the test cases this number was ten while only for a couple of test cases this number was found to be nine out of ten.

Finally, we measured the computation time required by the biased estimator to initially learn the generative model, then compute weights for the various components of the estimator, and to finally provide an estimate of the query result. We observed that for the synthetic data sets (which consists of 10 million and 50 million records in the two relations) the maximum observed running time of biased estimator was between 3 and 4 seconds for a 10% sample from each. The vast majority of this time is spent in the EM learning algorithm, which requires $O(m \times |\text{EMP}'| \times i)$ time, where m is the maximum possible number of matches for a record in EMP with records in SALES, and i is the number of iterations required for EM convergence. We speed our implementation by sub-sampling EMP' and using the subsample in the EM algorithm rather than using EMP' directly. The justification for this is that the EM can be quite expensive with a large EMP', and the accuracy of the modeling step is much more closely related to the size of SALE'. We use a subsample of size 500 in our experiments.

In comparison, computation for the unbiased estimator is almost instantaneous, requiring a small fraction of a second. In our test data, the most costly operation for the unbiased estimator is running the “join” between EMP' and SALE'; that is, searching for matches for each record from EMP' in SALE'. Given summary statistics describing this matching, the core GetEstTi routine itself can be implemented as a dynamic programming algorithm that takes time $O(m'^2)$, where m' is the maximum number of matches for any record from EMP' in SALE'.

6.3 Discussion

One of the most obvious results from Figure 3 is that the unbiased estimator has uniformly small error only on those eight tests performed using synthetic data set 1, where the number of matches for each record $e \in \text{EMP}$ is generated using a Gamma distribution with parameters (shift = 1, scale = 0.5). In this particular data set, only a very small number of the records are excluded by the NOT EXISTS clause since 86% of the records in EMP do not have a match in SALE. Furthermore, only a very small number of the records have a large number of matches. Both of these characteristics tend to stabilize the variance of the unbiased estimator, making it a fine choice.

For all the other data sets, the unbiased estimator does very poorly for most of the cases. For synthetic data, the estimator’s worst performance is for data set 6, in which less than one percent of the records are accepted by the NOT EXISTS clause and several records from EMP have more than 15 matching records in SALE. In this case, the unbiased estimator is unusable, and the results were particularly poor with correlation between the number of matches and the aggregate value that is summed. For example, in the correlated case with a 1% sample, most of the relative standard errors were more than 40000%. Such very poor results are found sporadically throughout most of the data sets, though the results were somewhat erratic. The reason that the observed errors associated with the unbiased estimator are highly variable is the very long tail of the error distribution. Under many circumstances, most of the answers computed using the unbiased estimator

Data Set Type			1% Sample Error			5% Sample Error			10% Sample Error		
Gamma	Cor-red ?	Val Dist.	U (%)	C (%)	B (%)	U (%)	C (%)	B (%)	U (%)	C (%)	B (%)
1	No	a.	7.39	13.32	38.30	2.39	12.62	3.88	1.09	11.89	1.46
1	No	b.	6.69	13.45	37.87	3.04	12.63	5.92	1.08	11.93	1.38
1	No	c.	6.89	12.92	22.59	5.23	12.04	8.18	3.79	11.23	7.09
1	No	d.	16.65	6.32	68.37	15.94	6.19	29.34	9.56	5.94	19.72
1	Yes	a.	11.90	20.90	34.50	4.59	19.94	2.26	3.15	18.68	1.42
1	Yes	b.	13.50	17.80	36.30	4.07	16.37	5.12	1.75	15.50	2.18
1	Yes	c.	7.70	15.06	21.14	5.69	14.06	7.84	3.98	13.13	6.21
1	Yes	d.	18.05	1.04	66.94	16.26	0.52	25.35	12.98	0.41	15.33
2	No	a.	11.79	40.12	6.09	8.10	37.98	3.55	2.43	35.44	3.37
2	No	b.	13.65	39.48	5.00	6.82	37.86	4.83	2.54	35.51	4.03
2	No	c.	179.87	39.20	14.75	6.35	37.00	8.34	4.54	34.44	7.12
2	No	d.	31.60	20.45	43.43	10.24	19.26	12.88	9.99	17.08	6.25
2	Yes	a.	24.70	65.60	21.39	19.83	62.00	18.45	4.78	57.51	13.70
2	Yes	b.	19.34	54.27	12.99	12.61	51.19	12.28	3.46	47.72	7.48
2	Yes	c.	220.14	46.60	23.01	12.19	44.01	12.01	5.10	40.88	5.10
2	Yes	d.	52.61	39.08	39.45	19.62	36.75	5.32	9.20	33.19	2.25
3	No	a.	234.60	92.75	18.61	59.67	84.91	12.22	33.00	76.00	6.28
3	No	b.	315.97	93.29	19.42	70.32	84.68	11.68	34.78	76.05	5.84
3	No	c.	188.17	91.50	20.53	46.14	84.01	18.50	24.92	75.07	15.80
3	No	d.	139.27	72.67	14.24	63.56	67.36	12.18	6.79	59.83	5.33
3	Yes	a.	753.73	189.70	42.19	220.00	172.10	28.99	115.25	151.85	17.02
3	Yes	b.	421.00	146.70	30.93	151.00	133.50	21.05	74.50	118.40	11.99
3	Yes	c.	240.20	119.80	28.28	74.66	109.50	25.99	42.57	97.22	21.86
3	Yes	d.	47.95	144.61	33.85	18.52	130.93	28.69	3.63	114.00	18.63
4	No	a.	153.70	36.20	14.52	37.17	33.90	4.73	24.47	31.20	0.89
4	No	b.	226.00	37.00	18.56	50.32	33.95	5.27	42.87	31.11	1.33
4	No	c.	242.70	35.20	11.10	19.40	32.85	3.62	17.03	30.04	3.59
4	No	d.	146.37	16.56	45.16	23.60	14.85	21.26	8.85	12.62	16.61
4	Yes	a.	418.70	64.50	10.85	116.55	59.94	2.71	27.55	54.52	1.64
4	Yes	b.	327.02	52.06	8.62	75.95	48.42	3.92	45.62	44.12	2.83
4	Yes	c.	359.60	43.40	13.90	30.19	40.39	7.17	27.21	36.80	5.16
4	Yes	d.	1144.63	37.53	40.29	54.33	33.99	10.66	18.94	29.32	5.68
5	No	a.	236.00	72.04	13.19	46.18	66.08	12.07	38.30	59.60	6.15
5	No	b.	395.00	72.30	11.78	55.78	66.09	11.73	42.73	59.55	5.37
5	No	c.	167.70	71.10	7.70	120.81	65.20	1.99	62.70	58.50	1.15
5	No	d.	135.65	51.87	13.58	77.12	48.29	4.30	24.14	42.21	4.16
5	Yes	a.	862.00	71.79	31.25	203.81	64.90	7.21	57.22	57.00	2.93
5	Yes	b.	650.80	56.60	28.64	129.75	51.46	6.75	74.16	43.90	1.86
5	Yes	c.	298.70	92.30	11.47	189.70	84.22	4.06	69.63	74.80	2.53
5	Yes	d.	283.26	105.24	10.84	178.61	95.07	9.38	145.78	81.86	3.04
6	No	a.	7129.00	95.13	19.30	6287.80	79.49	9.82	4113.00	63.33	6.09
6	No	b.	19771.80	95.20	18.40	2149.40	79.58	9.47	662.00	63.40	5.74
6	No	c.	19330.00	94.32	13.03	1195.30	78.60	5.96	964.00	62.74	1.71
6	No	d.	46919.00	76.71	7.54	204.57	66.87	8.42	68.87	54.96	3.97
6	Yes	a.	54188.00	307.00	62.00	30197.10	249.30	30.90	5754.00	119.00	18.78
6	Yes	b.	42371.00	214.00	42.70	19265.00	174.25	21.12	7025.00	135.00	12.88
6	Yes	c.	32151.00	156.30	22.70	2010.30	128.10	10.87	869.00	100.12	3.05
6	Yes	d.	135883.00	234.39	29.78	2991.00	192.46	28.25	2427.30	148.28	12.79

Fig. 3 Observed standard error as a percentage of SUM (e .SAL) over all $e \in \text{EMP}$ for 48 synthetically generated data sets. The table shows errors for three different sampling fractions: 1%, 5% and 10% and for each of these fractions, it shows the error for the three estimators: U - Unbiased estimator, C - Concurrent sampling estimator and B - Model-based biased estimator.

Data Set Type		1% Sample Error			5% Sample Error			10% Sample Error		
Gamma	Violates	U (%)	C (%)	B (%)	U (%)	C (%)	B (%)	U (%)	C (%)	B (%)
1	(1)	8.83	13.37	62.60	3.12	12.47	15.24	1.19	11.75	4.62
2	(1)	24.66	39.33	34.39	8.14	37.89	2.74	3.41	35.60	2.48
3	(1)	94.11	92.31	21.14	72.94	84.82	16.76	20.27	75.78	13.05
4	(1)	22.30	36.67	37.99	12.72	34.07	7.96	6.34	31.12	2.95
5	(1)	231.50	72.60	6.76	123.30	66.14	6.37	85.68	59.48	4.35
6	(1)	1366.80	95.96	9.99	1254.40	78.64	5.85	700.00	62.62	1.88
1	(2)	14.18	21.70	100.70	4.42	21.09	26.34	2.69	20.20	12.44
2	(2)	21.62	72.24	59.94	14.25	67.50	7.56	6.25	62.90	4.47
3	(2)	886.20	220.20	45.73	136.00	201.90	31.73	79.75	180.10	25.76
4	(2)	462.00	95.80	106.80	269.19	88.74	22.18	81.03	82.43	11.52
5	(2)	247.60	205.00	18.84	233.00	187.00	17.69	88.55	168.30	9.78
6	(2)	6891.00	369.00	42.30	5988.00	310.00	40.90	1924.00	246.57	19.77
1	(3)	14.70	21.14	61.86	6.24	20.20	10.15	1.13	19.13	2.67
2	(3)	26.15	66.73	29.10	22.49	62.25	20.25	5.38	57.69	17.35
3	(3)	920.10	185.30	41.86	147.60	167.20	30.12	65.63	146.88	27.20
4	(3)	23806.00	64.42	35.96	714.00	60.54	16.87	150.80	54.77	9.24
5	(3)	1350.30	143.00	33.59	856.00	127.76	29.58	306.70	113.14	10.08
6	(3)	22335.00	264.02	38.37	4519.10	212.80	34.92	2530.00	162.70	21.96

Fig. 4 Observed standard error as a percentage of SUM ($e.SAL$) over all $e \in \mathbf{EMP}$ for 18 synthetically generated data sets. The table shows errors for three different sampling fractions: 1%, 5% and 10% and for each of these fractions, it shows the error for the three estimators: U - Unbiased estimator, C - Concurrent sampling estimator and B - Model-based biased estimator.

		1% Sample Error			5% Sample Error			10% Sample Error		
Data-Set	Query	U (%)	C (%)	B (%)	U (%)	C (%)	B (%)	U (%)	C (%)	B (%)
IMDB	Q_1	9683.00	27.67	70.88	3371.00	17.51	33.44	413.5	13.71	14.14
IMDB	Q_2	124.92	75.12	65.10	91.26	62.86	31.97	49.82	52.69	9.31
IMDB	Q_3	10420.00	25.21	18.47	3548.00	16.58	14.38	472.60	12.71	1.92
SCR	Q_4	14783.00	65.22	10.31	5074.00	44.97	6.84	826.20	23.27	4.41
SCR	Q_5	12639.00	59.06	9.42	4663.00	41.62	7.51	783.90	24.07	3.95
KDDCup	Q_6	1.1e+10	60.47	12.39	74865.00	54.92	10.96	7640.00	42.08	2.10
KDDCup	Q_7	6.5e+147	41.30	11.24	5.85e+83	26.54	4.32	9.28e+36	17.04	3.28
KDDCup	Q_8	7.3e+210	15.24	8.46	3.6e+172	10.80	1.56	2.28e+120	6.35	0.98

Fig. 5 Observed standard error as a percentage of the total aggregate value of all records in the database for 8 queries over 3 real-life data sets. The table shows errors for three different sampling fractions: 1%, 5% and 10% and for each of these fractions, it shows the error for the three estimators: U - Unbiased estimator, C - Concurrent sampling estimator and B - Model-based biased estimator.

are very good, but there is still a small (though non-negligible) probability of getting a ridiculous estimate whose error is hundreds of times the sum over the aggregate value over the entire EMP relation. Unfortunately, it is interesting to note that the unbiased estimator’s worst performance overall was observed on Q_8 over the KDDCup data, where the error was astronomically high: larger than 10^{100} .

In comparison, the biased estimator generally did a very good job predicting the final query result, and in most cases with a 5% or 10% sampling fraction the observed standard error was less than 10% of the total aggregate value found in EMP. In other words, if the total value of SUM (e.SAL) with no NOT EXISTS clause is x , then for just about any query tested, the standard error was less than $x/10$, and it was frequently much smaller. This is actually quite impressive when one considers the difficulty of the problem. The primary drawback associated with the biased estimator is its complexity (requiring non-trivial and substantial statistically-oriented computations) and the fact that a significant amount of computation is required, most of it associated with running the EM algorithm to completion. By comparison, the unbiased estimate can be calculated via an almost trivial recursive routine that relies on the calculation of simple hypergeometric probabilities.

One case where the biased estimator had questionable qualitative performance was with the 16 tests associated with data sets 3 and 6. The problem in this case was that the EM algorithm tended to overestimate p_0 in Θ , which is actually very small in these two data sets (.052 and .0037, respectively). This results in an error that hovers at 10%+ of the total aggregate value of e.SAL (even for a 5%+ sample) when the real answer is only 5% of this total for data set 3 or less than 1% of this total for data set 6. We stress that guessing that only a few percent of the tuples in EMP have no matches in SALE from a small sample with limited information is an extremely difficult estimation problem, and we conjecture that without additional information (such as prior knowledge that the distribution represented by \mathbf{p} is a discretized gamma distribution) it will be very difficult to achieve better results.

Results from the synthetic data sets which specifically violate the assumptions of the superpopulation model are shown in Figure 4. The first six rows in the table show results for data sets in which more than one EMP record can match with a given record from SALE. The results show that violating this assumption of the model in the actual data set did not affect the accuracy of the biased estimator significantly. The next set of six rows in the table show results for data sets in which there is no linear relationship between the mean aggregate values of the different classes of EMP records. The results show that the biased estimator is about twice as inaccurate over these data sets as compared to corresponding data sets which do not have a strict violation of the assumption. The last six rows in the table show results over data sets in which the variances of the aggregate values of records from different classes are significantly different. Results show that these data sets affect the accuracy of the biased estimator as much as the data sets which violate the “linear relationship of mean values” assumption. However, the results are certainly not poor when these assumptions are violated, and the method

still seems to have qualitative performance that may be acceptable for many applications, particularly with a larger sample size.

The results from the eight queries over the three real-life data sets are depicted in Figure 5. The key difference in the characteristics of the real-life data sets compared to the synthetically-generated data sets is the number of matching records in the inner relation for a given record from the outer relation of the NOT EXISTS query. For the KDDCup data set, the maximum number of matching records in the inner relation is as high as 2500, while for the IMDB and SCR data sets this number is about 200 and 90 respectively. Due to this, none of the cases which are favorable for the use of the unbiased estimator (as described above) are observed in the real-life data sets. On the other hand, it can be seen from Figure 5 that the accuracy of the biased estimator is generally quite good over the real data.

We also note that the standard error of the biased estimator over the learned superpopulation seems to be a reasonable surrogate for the standard error of the biased estimator in practice. For most biased estimators, it is reasonable to use the standard error of the biased estimator in the same way that one would use the standard deviation of an unbiased estimator when constructing confidence bounds (see Sarndal et al. [5], Section 5.2). According to the Vysochanskii-Petunin inequality [29], any unbiased uni-modal estimator will be within three standard deviations of the correct answer 95% of the time, and according to the more aggressive central limit theorem, an estimator will be within two standard deviations of the correct answer 95% of the time. We observed that almost all of the tests, ten out of ten of the errors for the biased estimator were actually within *two* predicted standard errors of zero. This seems to be strong evidence for the utility of the bounds computed using the predicted standard error of the biased estimator.

We finally remark on the time required for the execution of the biased estimator. The biased estimator performs several computations including learning the model parameters, generating sufficient statistics for several population-sample pairs and then solving a system of equations to compute weights for the various components of the estimator. As discussed previously, this took no longer than four seconds for the largest samples tested. If this is not fast enough, we point out that it may be able to speed this time even more, though this is beyond the scope of the paper. While we used the traditional EM algorithm in our implementation, we note that EM can be made faster by using incremental variants [8,6,17] of the EM algorithm. These variants of the EM algorithm typically achieve faster convergence time by implementing the Expectation and/or the Minimization step of the EM algorithm partially.

7 Related Work

Estimation via sampling has a long history in databases. One of the oldest and best known works is Frank Olken’s PhD thesis [33]. Other classic efforts at sampling-based estimation over database data are the adaptive sampling of Lipton and Naughton [12,13] for join query selectivity estimation, and the sampling techniques of Hou et al. [9,10] for aggregate queries. More

recent well-known work on sampling is that on online aggregation by Haas, Hellerstein, and their colleagues [24, 15, 25].

The sampling-based database estimation problem that is closest to the one studied in this paper is that of sampling for the number of distinct values in a database. As discussed in the introduction to this paper, a solution to the problem of estimation over subset-based queries is a solution to the problem of estimating the number of distinct values in a database since the latter problem can be written as a `NOT EXISTS` query. The classic paper in distinct value estimation is due to Haas et al. [14]. For a survey of the state-of-the-art work on this problem in databases through the year 2000, we refer the reader to the Introduction of the paper by Charikar et al. on the topic [28]. The paper of Bunge and Fitzpatrick [19] provides a survey of work in the statistics area, current through the early 1990's. Work in statistics continues on this problem to this day. In fact, a recent paper from statistics by Mingoti [32] on the distinct value problem provided inspiration for our use of superpopulation techniques.

Though the problems of distinct value estimation and subset-based aggregate estimation are related, we note that the problem of estimating the number of distinct values is a very restricted version of the problem we study in this paper, and it is not immediately clear how arbitrary solutions to the distinct value problem can be generalized to handle subset-based queries. The most obvious difficulty in extending such methods to subset-based queries is the fact that a `NOT EXISTS` or related clause results in a complicated statistic summarizing two populations (the two tables that are queried over). Nonetheless, links between the problems do exist. For example, though our own unbiased estimator was not directly inspired by Goodman's estimator [31]⁵ and it takes a very different form, it is easy to argue that our unbiased estimator *must* be a generalization of Goodman's estimator. The reasoning is straightforward: Goodman's estimator is proven to be the only unbiased estimator for distinct value queries, and our own unbiased estimator is unbiased for distinct value queries. Therefore, they must be equivalent when used on this particular problem.

8 Conclusion

This paper has presented two sampling-based estimators for the answer to a *subset-based* query, where the answer to a `SUM` aggregate query (and by trivial extension, `AVERAGE` and `COUNT`) is restricted to consider only those tuples that satisfy a `NOT EXISTS` or related clause. The first estimator is provably unbiased, while the second makes use of superpopulation methods and was found to be much more accurate.

As discussed in Section 5.1 of the paper, one of the most controversial decisions made in the development of the latter estimator was our choice of a very general prior distribution. To a statistician from the so-called "Bayesian" school [11], this may be seen as a poor choice and Bayesian statistician may

⁵ Goodman's estimator is one of the earliest statistical estimators for distinct value queries.

argue that a more descriptive prior distribution, if appropriate, would increase the accuracy of the method. This is certainly true, if the selected distribution were a good match for the actual data distribution. In this paper, however, we have consciously chosen generality and its associated drawbacks in place of specificity. Our experimental results seem to argue that for a variety of different data distributions, the resulting estimator still has high accuracy. Still, this represents an intriguing question for future work: can a different prior distribution be chosen that is appropriate for use in real-world data sets, and which results in a more accurate estimator?

Finally, we note that the model-based method outlined in the latter half of the paper was designed specifically to address the problem of estimating the answer to a nested SQL query with a single table in the inner query and a single table in the outer query linked by a `NOT EXISTS` predicate. As is, our model is not directly applicable to arbitrarily complex nested queries. For example, nested queries may include multiple relations in the outer as well as the inner query. One could imagine sampling all of the input relations, and then using any result tuples that are discovered as part of the inner or outer subqueries as input into an estimator such as the one studied in this paper. However, this may be dangerous, and our superpopulation model is not directly applicable. The problem is that if there is a join in the inner (or outer) query, then the tuples produced via joining samples from the input relations are not i.i.d. samples from the join [15]. This means that the join itself must be modeled, which is a problem for future work. Another problem for future work is arbitrary levels of nesting. An inner query may itself be linked with another inner query via a `NOT EXISTS` or similar clause.

References

1. <http://www.imdb.com>.
2. <http://cdiac.ornl.gov/epubs/ndp/ndp026b/ndp026b.htm>.
3. C. Jermaine and A. Dobra and A. Pol and S. Joshi. Online estimation for subset-based SQL queries. In *31st International conference on Very large data bases*, pages 745–756, 2005.
4. D. Kempe and A. Dobra and J. Gehrke. Gossip-based computation of aggregate information. In *FOCS*, pages 482–491, 2003.
5. C.E. Sarndal and B. Swensson and J. Wretman. *Model Assisted Survey Sampling*. Springer, New York, 1992.
6. B. Thiesson and C. Meek and D. Heckerman. Accelerating em for large databases. *Mach. Learn.*, 45(3):279–299, 2001.
7. M. Muralikrishna and D.J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *SIGMOD Conference*, pages 28–36, 1988.
8. R. Neal and G. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, 1998.
9. W.-C. Hou and G. Özsoyoglu. Statistical estimators for aggregate relational algebra queries. *ACM Trans. Database Syst.*, 16(4):600–654, 1991.
10. W.-C. Hou and G. Özsoyoglu. Processing time-constrained aggregate queries in case-db. *ACM Trans. Database Syst.*, 18(2):224–261, 1993.
11. A. Gelman and J.B. Carlin and H.S. Stern and D.B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman & Hall/CRC, 2003.
12. R.J. Lipton and J.F. Naughton. Query size estimation by adaptive sampling. In *PODS*, pages 40–46, 1990.

-
13. R.J. Lipton and J.F. Naughton and D.A. Schneider. Practical selectivity estimation through adaptive sampling. In *SIGMOD Conference*, pages 1–11, 1990.
 14. P.J. Haas and J.F. Naughton and S. Seshadri and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB*, pages 311–322, 1995.
 15. P.J. Haas and J.M. Hellerstein. Ripple joins for online aggregation. In *SIGMOD Conference*, pages 287–298, 1999.
 16. Y. Matias and J.S. Vitter and M. Wang. Wavelet-based histograms for selectivity estimation. In *SIGMOD Conference*, pages 448–459, 1998.
 17. H. Huang and L. Bi and H. Song and Y. Lu. A variational em algorithm for large databases. In *International Conference on Machine Learning and Cybernetics*, pages 3048–3052, 2005.
 18. P. Haas and L. Stokes. Estimating the number of classes in a finite population. *Journal of the American Statistical Association*, 93:1475–1487, 1998.
 19. J. Bunge and M. Fitzpatrick. Estimating the number of species: A review. *Journal of the American Statistical Association*, 88:364–373, 1993.
 20. C.T. Fan and M.E. Muller and I. Rezucha. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *Journal of the American Statistical Association*, 57:387–402, 1962.
 21. A. Dobra and M.N. Garofalakis and J. Gehrke and R. Rastogi. Processing complex aggregate queries over data streams. In *SIGMOD Conference*, pages 61–72, 2002.
 22. A.P. Dempster and N.M. Laird and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B.*, 39, 1977.
 23. P.G. Brown and P.J. Haas. Techniques for warehousing of sample data. In *ICDE*, page 6, 2006.
 24. J.M. Hellerstein and P.J. Haas and H.J. Wang. Online aggregation. In *SIGMOD Conference*, pages 171–182, 1997.
 25. J.M. Hellerstein and R. Avnur and A. Chou and C. Hidber and C. Olston and V. Raman and T. Roth and P.J. Haas. Interactive data analysis: The cONTROL project. *IEEE Computer*, 32(8):51–59, 1999.
 26. D. Krewski and R. Platek and J.N.K. Rao. *Current Topics in Survey Sampling*. Academic Press, 1981.
 27. B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1998.
 28. M. Charikar and S. Chaudhuri and R. Motwani and V.R. Narasayya. Towards estimation error guarantees for distinct values. In *PODS*, pages 268–279, 2000.
 29. D.F. Vysochanskii and Y.I. Petunin. Justification of the 3-sigma rule for unimodal distributions. *Theory of Probability and Mathematical Statistics*, 21:25–36, 1980.
 30. P. Domingos. Bayesian averaging of classifiers and the overfitting problem. In *17th International Conf. on Machine Learning*, 2000.
 31. L.A. Goodman. On the estimation of the number of classes in a population. *Annals of Mathematical Statistics*, 20:272–579, 1949.
 32. S.A. Mingoti. Bayesian estimator for the total number of distinct species when quadrat sampling is used. *Journal of Applied Statistics*, 26(4):469–483, 1999.
 33. F. Olken. Random sampling from databases. Technical Report LBL-32883, Mailstop 50B-3238, 1 Cyclotron Road, Berkeley, California 94720, U.S.A., 1993.

A EM Algorithm Derivation

Let Y_e be the information about record $e \in \text{EMP}$ that can be observed i.e. $v = f_1(e)$ and $k' = \text{cnt}(e, \text{SALE}')$. Let X_e be the information about record e that includes Y_e as well as the relevant data that cannot be observed, i.e. $k = \text{cnt}(e, \text{SALE})$. Then let:

$$f(X_e = \langle Y_e, k \rangle | \Theta) = p_k \times \frac{e^{-(v - \mu_k)^2 / 2\sigma^2}}{\sigma\sqrt{2\pi}} \times h(k'; k)$$

Also, let:

$$g(Y_e|\Theta) = \sum_{i=0}^m p_i \times \frac{e^{-(v-\mu_i)^2/2\sigma^2}}{\sigma\sqrt{2\pi}} \times h(k'; i)$$

We then compute the posterior probability that e belongs to class i as:

$$\tilde{p}(i|\Theta, e) = \frac{f(X_e = \langle Y_e, i \rangle | \Theta)}{g(Y_e|\Theta)} \quad (24)$$

Then the logarithm of the expected probability that we would observe EMP' and SALE' is:

$$\begin{aligned} E &= \sum_{e \in \text{EMP}} \log(f(X_e = \langle Y_e, i \rangle | \Theta)) \tilde{p}(i|\Theta', e) \\ &= \sum_{e \in \text{EMP}} \sum_{i=0}^m \tilde{p}(i|\Theta', e) \\ &\quad \times \log \left(p_i \times \frac{e^{-(f_1(e) - \mu_i)^2/2\sigma^2}}{\sigma\sqrt{2\pi}} \times h(k'; i) \right) \\ &= \sum_{e \in \text{EMP}} \sum_{i=0}^m \tilde{p}(i|\Theta', e) \times (\log(p_i) - \log(\sigma) - \log(\sqrt{2\pi}) \\ &\quad - \frac{(f_1(e) - \mu_i)^2}{2\sigma^2} + \log(h(k'; i))) \end{aligned}$$

To find the unknown parameters, μ_i, σ and p_i , we maximize E for the given set of posterior probabilities at that step. We do this by taking partial derivatives of E w.r.t each of these parameters and setting the result to zero:

$$\frac{\partial E}{\partial \mu_1} = \sum_{e \in \text{EMP}} \tilde{p}(1|\Theta', e) \times \frac{f_1(e) - \mu_1}{\sigma^2}$$

Setting this expression to zero gives:

$$\mu_1 = \frac{\sum_{e \in \text{EMP}} \tilde{p}(1|\Theta', e) \times f_1(e)}{\sum_{e \in \text{EMP}} \tilde{p}(1|\Theta', e)} \quad (25)$$

We can obtain μ_2, \dots, μ_m in a similar manner.

By taking the partial derivative of E w.r.t σ^2 and setting to zero we get:

$$\sigma^2 = \frac{\sum_{e \in \text{EMP}} \sum_{j=0}^m \tilde{p}(j|\Theta', e) (f_1(e) - \mu_j)^2}{\sum_{e \in \text{EMP}} \sum_{j=0}^m \tilde{p}(j|\Theta', e)} \quad (26)$$

Finally, to evaluate the p_i 's, we also consider the additional constraint that $\sum_{i=0}^m p_i = 1$. We can find the values of p_i 's that maximize E subject to this constraint by using the method of Lagrangian multipliers to obtain:

$$p_j = \frac{\sum_{e \in \text{EMP}} \tilde{p}(j|\Theta', e)}{\sum_{e \in \text{EMP}} \sum_{i=1}^m \tilde{p}(i|\Theta', e)} \quad (27)$$

This completes the derivation of the update rules given in Section 5.2.