PSALM: Accurate Sampling for Cardinality Estimation in a Multi-user Environment

Huaxin Zhang, Ihab F. Ilyas, Kenneth Salem

Technical Report CS2007-21

David R. Cheriton School of Computer Science

Abstract

In database systems that support fine-grained access controls, each user has access rights that determine which tuples are accessible and which are inaccessible. Queries are answered as if the inaccessible tuples are not present in the database. Thus, users with different access rights may get different answers to a given query. To process queries efficiently in the presence of fine-grained access controls, the database system needs accurate estimates of the number of tuples that are both accessible according to the access rights of the submitting user and relevant according to the selection predicates in the query. In this paper we present sampling-based cardinality estimation techniques for use in the presence of fine-grained access controls. These techniques exploit the fact that access rights are relatively static and are common to all queries that are evaluated on behalf of a particular user. We show that the proposed techniques provide more accurate estimates than simpler techniques that do not exploit knowledge of access rights. We quantify these improvements analytically and through simulations.

1 Introduction and Motivation

Access controls determine which data are accessible to each database user. Fine-grained access controls allow access rights to be specified on a per-tuple basis. Under the so-called *Truman model* [17], a user's queries are answered as if the database includes only those tuples that are accessible to the user. Variations of the Truman model are commonly used by systems that support fine-grained access controls, including Oracle [14, 15], DB2 [10], and SQL Server [12]. For example, Oracle customizes each user's query by extracting the user's *access control predicates* from the user's profile and appending the predicates to the *where* clause of the query [15]. The access control predicates filter tuples that are inaccessible to the user.

To create efficient query plans in the presence of fine-grained access controls, a query optimizer needs accurate cardinality estimates that account for the effect of the access controls. That is, instead of estimating the number of tuples that satisfy a particular query predicate, the optimizer must estimate the number of *accessible* tuples that satisfy the predicate. Since different users have different access rights, this number depends on which user submitted the query. This is the problem that we address in this paper.

Suppose that a query includes a predicate P_Q on one of its input relations, R. P_Q may be a simple or complex predicate. Suppose further that the predicate P_{ACi} represents the access rights of the i^{th} user, for relation R. That is, those tuples of R that satisfy P_{ACi} are accessible to user i, and those that do not satisfy P_{ACi} are not accessible. If the query is being executed on behalf of the i^{th} user, the effect of the fine-grained access controls is to replace P_Q in the query with $P_Q \wedge P_{ACi}$. To obtain an accurate estimate of the number of R tuples that will be used by the query, the query optimizer needs to estimate the selectivity of $P_Q \wedge P_{ACi}$.

If P_{ACi} is expressed in a form that is understandable to the query optimizer, e.g., as a SQL predicate, it can simply estimate the selectivity of $P_Q \wedge P_{ACi}$ as it would for any other complex predicate. However, there are two problems with this approach. First, it ignores the fact that P_{ACi} is relatively static and is common to all queries involving relation R that are executed on behalf of the i^{th} user. A user's access rights change only in response to changes in the system's access control policies, and we expect that such changes will occur much less frequently than queries. The second problem is the difficulty of estimating the selectivity of complex predicates. The typical approach is to obtain individual selectivity estimates for P_Q and P_{ACi} and then multiply those estimates to obtain an estimate for $P_Q \wedge P_{ACi}$. This assumes that P_Q and P_{ACi} are independent, which will not, in general, be the case. If P_Q and P_{ACi} are not independent, the resulting estimate may be inaccurate. Of course, this problem is not restricted to selectivity estimation in the presence of fine-grained access controls. The query optimizer faces similar problems when it is presented with any complex predicate, whether it involves access controls or not. However, fine-grained access controls exacerbate the problem. In the presence of fine-grained controls, *all* query predicates become more complex, because they must incorporate the effects of the access controls.

In this paper, we consider estimation techniques based on sampling. Sampling allows for direct estimation of the joint selectivity of P_Q and P_{ACi}, without resorting to independence assumptions. Furthermore, the sampling techniques that we propose are able to exploit the fact that the access controls are query-independent and relatively static. This paper makes the following contributions:

1. We propose a biased sampling technique called basic PSALM (Partitioned SAmpLing for Multiple users) for estimating cardinality in the presence of fine-grained access controls. We compare basic PSALM to estimation based on simple uniform random sampling and show that basic PSALM provides better estimation accuracy for users with few access rights, i.e., users with highly selective access control predicates.
2. We show how to improve on the basic PSALM algorithm by using biased sampling for low-privilege users and uniform random sampling for high-privilege users. We show that this hybrid approach provides better estimation accuracy than either uniform sampling or the basic PSALM algorithm alone.
3. We show how to exploit correlation of access rights among different users

Table 1: Symbols used in this paper

U	number of users in the system
P_Q	query selection predicate
P_{ACi}	access control predicate for the i^{th} user
N	cardinality of the target relation
N_i	number of tuples matching P_{ACi}
C_i	number of tuples matching $P_Q \wedge P_{ACi}$
$\tilde{C}_i$	estimate of C_i
n_i	number of tuples in i^{th} user's private sample
c_i	number of tuples in i^{th} user's private sample that satisfy $P_Q \wedge P_{ACi}$
ϵ_i	estimation error for i^{th} user
ϵ	mean estimation error over all users
ϵ	maximum estimation error over all users
$\hat{\epsilon_i}$	estimation error bound for i^{th} user
$\hat{\epsilon}_{mean}$	mean estimation error bound over all users
$\hat{\epsilon}_{max}$	maximum estimation error bound over all users
Δ	confidence level for estimation error bounds

to obtain higher estimation accuracy.

The remainder of the paper is organized as follows. Section 2 defines the problem of selectivity estimation in the presence of fine-grained access controls and introduces terminology and notation. Section 3 presents the basic PSALM algorithm and compares it to simple random sampling. Section 4 presents the hybrid PSALM technique, and Section 5 describes a refinement of hybrid PSALM that is able to exploit access rights correlations to reduce estimation error. Section 7 describes related work, and Section 8 concludes and summarizes our results.

2 Definitions and Notation

As discussed in Section 1, our problem is to estimate the selectivity of query predicates in the presence of fine-grained user access controls. We will focus on the problem of estimating the cardinality of the result of applying an arbitrary query predicate, P_Q, to a single access-controlled relation. When there are multiple relations, our estimation techniques can be applied independently to each relation.

We assume that each user's access rights for the target relation are defined by a user-specific access control predicate. We use P_{ACi} to denote the access control predicate for the i^{th} user on the target relation. Tuples are accessible to user i if and only if they satisfy P_{ACi}. The form of P_{ACi} is not important. It may be an actual SQL predicate, as in Oracle VPD [15], or it may be implemented as an externally-defined function, as is the case in some fine-grained label-based access control systems [10]. For the purposes of our work, it is only necessary to be able to evaluate the predicate given a tuple from the target relation.

We will use N to denote the cardinality of the target relation, and N_i to denote the number of tuples from the target relation that are accessible to the i^{th} user. That is, N_i is number of target relation tuples for which P_{ACi} is true. Finally, we will use C_i to denote the number of tuples from the target relation that satisfy $P_Q \wedge P_{ACi}$. C_i is the cardinality we wish to estimate, for any given user, target relation, and query predicate P_Q. Table 1 summarizes the notation that we will use in this paper.

Given a fixed space budget for cardinality estimation, our goal is to design estimation techniques with low estimation error. Suppose that $\tilde{C}_i$ is a cardinality estimate produced by one of the estimation techniques that we will present in this paper. Following earlier work in this area [2, 3], we define ϵ_i, the estimation error for the i^{th} user, to be

$$\epsilon_i = \frac{|\tilde{C}_i - C_i|}{C_i}$$

This metric characterizes the estimation error relative to the actual cardinality of the query result. Thus, for example, $\epsilon_i = 0.3$ indicates that the estimate is $\pm 30\%$ of the actual cardinality. Unlike absolute error metrics such as $|\tilde{C}_i - C_i|$ or $|\tilde{C}_i - C_i|/N$, our relative error metric reflects the fact a large cardinality estimation error may be much more significant to the query optimizer when the true cardinality is small than when the true cardinality is large. For example, if $|\tilde{C}_i - C_i| = 10000$ and $\tilde{C}_i = 100000$, the estimation error may have little effect on the optimizer. However, if $|\tilde{C}_i - C_i| = 10000$ and $\tilde{C}_i = 100$, the optimizer may significantly underestimate the cost of a candidate query plan. One disadvantage of our metric is that estimation error explodes as $C_i \rightarrow 0$. Fortunately, this is not a serious issue. Our cardinality estimation techniques do not require calculating estimation error. Estimation error is used only for comparing the estimation techniques. To avoid the blowup, we simply avoid scenarios in which C_i is extremely small.

3 Two Basic Estimation Techniques

We begin by presenting two simple sampling-based estimation techniques. The first uses a single uniform random sample to generate a cardinality estimate for any user. That is, estimates for all users are based on the same tuple sample from the target relation. The second technique is *basic Partitioned SAmpling for MuLtiple users*, or *basic PSALM.* It partitions the available space and draws a separate, smaller sample for each user. Each user's cardinality estimations are based on that user's private sample. We determine bounds on the estimation error resulting from each technique, and characterize situations in which basic PSALM provides more accurate estimates than the single-sample approach. In subsequent sections, we will present techniques that improve on both the single-sample approach and basic PSALM.

3.1 Single Uniform Sample

We can estimate the cardinality of $P_Q \wedge P_{ACi}$, for any user and any query predicate, using a random sample of n tuples from the target relation. Such a sample can be built by a single pass over the target relation, using a technique such as reservoir sampling [18]. If desired, such a sample can be incrementally maintained in the face of tuple insertions and deletions in the target relation [7]. Alternatively, the target relation can be periodically resampled as necessary to account for changes.

To obtain a cardinality estimate for P_Q for the i^{th} user, we evaluate $P_Q \wedge P_{ACi}$ for each sample tuple, and count the number of tuples for which the predicate is true. An unbiased cardinality estimate can then be obtained by

$$\tilde{C}_i = c_i \frac{N}{n}$$

where c_i is the number of sample tuples matching $P_Q \wedge P_{ACi}$.

Because $\tilde{C}_i$ is based on a random sample of tuples rather than the entire relation, the estimate may not be accurate. Since each sample tuple satisfies $P_Q \wedge P_{ACi}$ with probability $\frac{C_i}{N}$, c_i can be modeled as binomial random variable with parameters n and $\frac{C_i}{N}$. Using the Chernoff inequality, we can bound the estimation error as follows:

$$\begin{aligned}
& Prob[\tfrac{|\tilde{C}_i - C_i|}{C_i} \leq \epsilon] = Prob[\tfrac{|\frac{c_i N}{n} - C_i|}{C_i} | \leq \epsilon] \\
= \; & Prob[(1-\epsilon)n\tfrac{C_i}{N} \leq c_i \leq (1+\epsilon)n\tfrac{C_i}{N}] \\
\geq \; & 1 - 2e^{(-nC_i\epsilon^2)/(4N)}
\end{aligned}$$

Thus, with probability $(1 - \Delta)$, the cardinality estimation error ϵ_i for the i^{th} user under the single random sample method is bounded by

$$\epsilon_i \leq \hat{\epsilon}_i = \sqrt{\frac{4N}{nC_i} \log \frac{2}{\Delta}} \qquad (1)$$

We will refer to Δ as the confidence level of the error bound $\hat{\epsilon}_i$. Note that the estimation error bound is inversely related to C_i which is the number of tuples that satisfy $P_Q \wedge P_{ACi}$. Thus, as either the query predicate P_Q or the user's access controls P_{ACi} become more highly selective, the estimation error bound increases.

3.2 Basic PSALM

Another way to estimate the cardinality of $P_Q \wedge P_{ACi}$ is to create and maintain a separate sample for each user. Let n_i denote the size of the sample for the i^{th} user, and let U represent the number of users. We can choose the n_i such that $\sum_{i=1}^{U} n_i = n$ to ensure that the basic PSALM technique uses the same amount of space as the single-sample approach.

Under the basic PSALM approach, the sample for the i^{th} user is a simple uniform random sample of the tuples *that are accessible to the i^{th} user.* As was the case for the single-sample approach, we can draw all U such random samples using a single pass over the target relation by maintaining Useparate reservoir samples in parallel during the scan. Each tuple encountered in the scan is considered separately and independently for inclusion in the sample for each user. For the i^{th} user, the tuple is first tested against P_{ACi}. If the tuple satisfies the predicate, then it is considered for inclusion in the reservoir sample for user i as usual. Otherwise, it is not included in the i^{th} sample.

To estimation cardinality for P_Q for the i^{th} user, we evaluate P_Q for each tuple in the i^{th} user's sample, and count the number of tuples for which the predicate is true. All other users' samples are ignored. An unbiased cardinality estimate can then be obtained by

$$\tilde{C_i} = c_i \frac{N_i}{n_i}$$

where c_i is the number of sample tuples matching P_Q in the i^{th} user's sample. Notice that this estimator makes use of N_i, the total number of target relation tuples that are accessible to the i^{th} user. This value can be determined exactly for every user during the same scan that is used to draw the tuple samples from the target relation.

Using an analysis similar to the one in Section 3.1, we can bound, with confidence level Δ, the estimation error that results from the basic PSALM technique:

$$\epsilon_i \leq \hat{\epsilon_i} = \sqrt{\frac{4N_i}{n_i C_i} \log \frac{2}{\Delta}} \tag{2}$$

Recall that under the single sample approach, the estimation error bound is inversely related to the selectivity of the joint predicate $P_Q \wedge P_{ACi}$. For the basic PSALM technique, the estimation error is inversely related to C_i/N_i, which is the *conditional selectivity* of the query predicate P_Q, given that a tuple is accessible to the i^{th} user. Unlike the single sample approach, basic PSALM's estimation error is independent of the selectivity of the users' access control predicates. This makes sense, because the basic PSALM technique ensures that it has a sample of size n_i of tuples accessible to the i^{th} user, regardless of how many such tuples exist.

One question remains for the basic PSALM technique. How should we determine the number of tuples, n_i, to include in the sample for the i^{th} user? To answer this question, we use two metrics for evaluating the accuracy of a multi-user sampling scheme.

The ***mean estimation error bound*** ($\hat{\epsilon}_{mean}$) over all users:

$$\hat{\epsilon}_{mean} = \frac{\sum_{i=1}^{U} \hat{\epsilon_i}}{U}$$

The ***maximum estimation error bound*** ($\hat{\epsilon}_{max}$) over all users

$$\hat{\epsilon}_{max} = \frac{\max_{i=1}^{U} \hat{\epsilon}_i}{U}$$

For example, under uniform sampling of size n, each user u_i has estimation error $\sqrt{\frac{4N}{n \cdot C_i} \log \frac{2}{\Delta}}$ according to Formula 1. Therefore, the mean estimation error bound is

$$\frac{1}{U} \sum_{i=1}^{U} \sqrt{\frac{4N}{n \cdot C_i} \log \frac{2}{\Delta}} \tag{3}$$

and the maximum estimation error bound for uniform sampling is:

$$\max_{i=1}^{U} \sqrt{\frac{4N}{n \cdot C_i} \log \frac{2}{\Delta}} \tag{4}$$

Ideally, we would like to distribute the available space among the users' samples in the basic PSALM in such a way as to minimize either the mean estimation error bound or the maximum estimation error bound. In general, the best such distribution will be workload dependent, determined by the conditional selectivities of the query predicates with respect to each user's accessible tuples. However, for the special case in which the query predicates P_Q in the workload are independent of the users' access controls, there is a simple workload-independent way to determine optimal per-user sample sizes for either metric, as shown in Section 3.3 and Section 3.4 respectively. Note that we are assuming independence of the query predicate and the access controls *only* for the purpose of choosing the sizes of the per-user samples. The actual estimation of cardinalities using those samples does not rely on the independence of query predicates and access controls.

3.3 Minimizing the Mean Estimation Error Bound

We first have the following theorem:

Theorem 3.1. *If the query predicate P_Q is independent of the access control predicates P_{ACi} of all users, then the expected $\hat{\epsilon}_{mean}$ is minimized by choosing $n_i = n/U$ for every user.*

Proof. Basic PSALM has a mean estimation error bound of

$$\frac{\sum_{i=1}^{U} \hat{\epsilon}_i}{U} = \frac{\sum_{i=1}^{U} \sqrt{\frac{4N_i}{n_i C_i} \log \frac{2}{\Delta}}}{U}$$

Because we assume P_Q is independent of the access control predicates, (N_i/C_i) is in expectation the same for all users, and we can rewrite the mean estimation error as

$$\alpha \sum_{i=1}^{U} \sqrt{\frac{1}{n_i}}$$

where α is a constant term independent of i. Our goal is to minimize this quantity by adjusting $\{n_i, i \in (1, \ldots, U)\}$ under the constraint that $\sum_{i=1}^{U} n_i = n$. We use the method of Lagrange multipliers. Adding the constraint to our formula, we have:

$$\Phi(X, \lambda) = \alpha \sum_{i=1}^{U} \sqrt{\frac{1}{n_i}} + \lambda \left(\sum_{i=1}^{U} n_i - n \right)$$

The critical value of Φ occurs when the gradients on $\{n_i, i \in (1, \ldots, U)\}$ and λ are all zero:

$$\frac{\partial \Phi}{\partial n_i} = \frac{\partial \frac{\alpha}{\sqrt{n_i}}}{\partial n_i} + \lambda = -\alpha n_i^{-\frac{3}{2}} + \lambda = 0, i \in \{1, \ldots, U\}$$

$$\frac{\partial \Phi}{\partial \lambda} = \sum_{i=1}^{U} n_i - n = 0$$

Solving the above, we have $n_i = \frac{n}{U}, i \in (1, \ldots, U)$.

□

Comparing Equations 1 and 2, we can see that the i^{th} user will have a tighter error bound under the basic PSALM technique under the condition:

$$\frac{n_i}{N_i} > \frac{n}{N}$$

This condition states that basic PSALM will provide more accurate estimates than the single-sample approach for the i^{th} user if the sampling rate from among the i^{th} user's accessible tuples is greater than the single sample's sampling rate from the whole target relation. Assuming that PSALM sample sizes are chosen according to Theorem 3.1 ($n_i = n/U$), then PSALM will have a tighter estimation error bound for the i^{th} user if

$$N_i < \frac{N}{U} \tag{5}$$

Thus, users with very limited access rights, i.e., highly selective access control predicates, will benefit from the basic PSALM technique. However, users with broad access rights may experience larger estimation errors. Over all of the users in the system, we have the following theorem concerning the mean estimation error bound:

Theorem 3.2. *Under the optimal sample quota assignment from Theorem 3.1, basic PSALM has a lower mean estimation error bound $\hat{\epsilon}_{mean}$ than the single sample approach if*

$$\sum_{i=1}^{U}\sqrt{\frac{N}{N_i}} > U^{\frac{3}{2}}$$

Proof. The mean estimation error bound reaches its minimum ($\alpha U\sqrt{\frac{U}{n}}$) when we assign equal sample quota to each user. On the other side, we have total estimation error of uniform sampling as ($\alpha \sum_{i=1}^{U}\sqrt{\frac{N}{nN_i}}$). Comparing these two values, we get the theorem.

□

3.4 Minimizing the Maximum Estimation Error

To minimize the maximum estimation error bound over all users, we have the following theorem:

Theorem 3.3. *If the query predicate P_Q is independent of the access control predicates P_{ACi} of all users, then the expected $\hat{\epsilon}_{max}$ is minimized for the basic PSALM by choosing $n_i = n/U$ for every user. Under this sample size assignment, the basic PSALM has lower maximum estimation error bound than uniform sampling whenever there is a user u_i such that $\frac{N}{N_i} > U$.*

Proof. The maximum estimation error bound over all users is:

$$\max_{i=1}^{U}\sqrt{\frac{4N_i}{n_i \cdot C_i}\log\frac{2}{\Delta}}$$

Similar to the previous analysis on minimizing the mean estimation error bound, we can rewrite the above into the following, where β is a constant independent of i:

$$\max_{i=1}^{U}\beta\sqrt{\frac{1}{n_i}}$$

We already know that the mean error bound is minimized when every user has the same error estimation bound. Assume there is a scheme that gives a lower maximum error estimation bound. Then this new scheme will give a lower mean error estimation bound, which is a contradiction. Therefore, the maximum estimation error bound can not be lower than the mean estimation error bound when each user has the same size sample. This shows that the maximum estimation error bound is achieved when each user has the same sample size as well. □

4 Exploiting Access Privilege Skew

As shown by Equation 5, the basic PSALM approach works well when users have few access rights. The single-sample approach works best if users are able to access most of the target relation's tuples. In this section, we develop a hybrid estimation technique that combines advantages of both approaches, and we show that it always provides estimation accuracy that is at least as good as that provided by either the single-sample approach or the basic PSALM approach.

Under the hybrid approach, which we call *hybrid PSALM*, users are divided into two groups; one group consisting of high-privilege users and the other consisting of low privilege users. We will denote the high-privilege and low-privilege groups by $\mathcal{U}_H$ and $\mathcal{U}_L$, respectively. Later, we will show how to decide which group each user should belong to.

Hybrid PSALM maintains a single shared sample for all users in $\mathcal{U}_H$, and a separate private sample for each individual user in $\mathcal{U}_L$. We will use n_0 to represent the size of the shared sample for the users in $\mathcal{U}_H$. If the i^{th} user is in $\mathcal{U}_L$, we will use n_i to represent the size of that user's private sample.

To obtain a cardinality estimate for P_Q for the i^{th} user, hybrid PSALM proceeds as follows:

- If the i^{th} user is in $\mathcal{U}_H$, we evaluate $P_Q \wedge P_{ACi}$ against the tuples in the shared sample. If c_i is the number of matching tuples from the shared sample, then the cardinality estimate for P_Q is

$$\tilde{C}_i = c_i \frac{N}{n_0} \tag{6}$$

  As was the case for the single-sample approach, we can show that the estimation error for this estimate is bounded, with confidence level Δ, by

$$\epsilon_i \leq \sqrt{\frac{4N}{n_0 C_i} \log \frac{2}{\Delta}} \tag{7}$$

- If the i^{th} user is in $\mathcal{U}_L$, evaluate P_Q against the tuples in that user's private sample. If c_i is the number of matching tuples from the private sample, the cardinality estimate is

$$\tilde{C}_i = c_i \frac{N_i}{n_i} \tag{8}$$

  and the estimation error for this estimates is bounded, with confidence level Δ, by

$$\epsilon_i \leq \sqrt{\frac{4N_i}{n_i C_i} \log \frac{2}{\Delta}} \tag{9}$$

To implement this hybrid approach, we must determine which users belong in $\mathcal{U}_L$ and which belong in $\mathcal{U}_H$. In addition, we must determine the sizes n_i of the private samples for users in $\mathcal{U}_L$ and the size n_0 of the shared sample for those in $\mathcal{U}_H$. As was the case for the basic PSALM, our goal is to do this so as to minimize the mean estimation error bound $\hat{\epsilon}_{mean}$ or the maximum estimation error bound $\hat{\epsilon}_{max}$ over all of the users.

4.1 Minimizing the Mean Estimation Error Bound

The following theorem tells us how to choose sample sizes given a specific group partition of the users.

Theorem 4.1. *Suppose that the users are partitioned arbitrarily into two groups, $\mathcal{U}_L$ and $\mathcal{U}_H$, such that each user in $\mathcal{U}_L$ is given a private sample for cardinality estimation while all users in $\mathcal{U}_H$ share a single common sample, and that the sum of the sample sizes is n. If query predicates P_Q are independent of the access control predicates of all users, then the mean estimation error bound $\hat{\epsilon}_{mean}$ for the hybrid PSALM algorithm is minimized when the sample sizes are chosen as*

$$n_0 = n\frac{Y^{\frac{2}{3}}}{Y^{\frac{2}{3}}+|\mathcal{U}_L|} \quad \text{for } \mathcal{U}_H$$

$$n_i = n_L = n\frac{1}{Y^{\frac{2}{3}}+|\mathcal{U}_L|} \quad \text{for each user } u_i \text{ in } \mathcal{U}_L$$

where $Y = \sum_{u_i \in \mathcal{U}_H} \sqrt{\frac{N}{N_i}}$.

Proof. Hybrid PSALM has mean estimation error bound of

$$\frac{1}{U}\left(\sum_{i\in(1,\ldots,j)} \sqrt{\frac{4N_i}{n_iC_i}\log\frac{2}{\Delta}} + \sum_{i\in(j+1,\ldots,U)} \sqrt{\frac{4N}{n_0C_i}\log\frac{2}{\Delta}}\right)$$

Because the query predicate P_Q is independent from the access control predicates P_{AC}, this can be rewritten as

$$\beta\left(\sum_{u_i\in\mathcal{U}_L}\sqrt{\frac{1}{n_i}} + \sum_{u_i\in\mathcal{U}_H}\sqrt{\frac{N}{n_0N_i}}\right) \tag{10}$$

where β is a constant independent of i. Our goal is to find n_0 and n_i to minimize this expression under the constraint that $n_0 + \sum_{u_i\in\mathcal{U}_L} n_i = n$.

We use Lagrange multipliers to find the minimum, by adding the constraint $n = n_0 + \sum_{u_i\in\mathcal{U}_L} n_i$ as multiplier:

$$\begin{aligned}\Phi(X,\lambda) \quad = \quad & \beta\left(\sum_{u_i\in\mathcal{U}_L}\sqrt{\frac{1}{n_i}} + \sum_{u_i\in\mathcal{U}_H}\sqrt{\frac{N}{n_0N_i}}\right) + \\ & \lambda\left(\sum_{u_i\in\mathcal{U}_L} n_i + n_0 - n\right)\end{aligned}$$

The critical value of Φ occurs when the gradients on n_0, n_i, λ are all zero:

$$\begin{aligned}
\frac{\partial\Phi}{\partial n_i} &= \beta\frac{\partial\frac{1}{\sqrt{n_i}}}{\partial n_i} + \lambda = -\frac{\beta}{2}n_i^{-\frac{3}{2}} + \lambda = 0, u_i \in \mathcal{U}_L \\
\frac{\partial\Phi}{\partial n_0} &= -\frac{\beta}{2}\sum_{u_i\in\mathcal{U}_L}\sqrt{\frac{N}{N_i}}n_0^{-\frac{3}{2}} + \lambda = 0 \\
\frac{\partial\Phi}{\partial\lambda} &= \sum_{u_i\in\mathcal{U}_L} n_i + n_0 - n = 0
\end{aligned}$$

We use Y to represent $\left(\sum_{u_i\in\mathcal{U}_H}\sqrt{\frac{N}{N_i}}\right)$. The mean estimation bound reaches its minimum when the following holds:

$$\begin{aligned}
\lambda &= \frac{\beta}{2}[\frac{Y^{\frac{2}{3}} + |\mathcal{U}_L|}{n}]^{\frac{3}{2}} \\
n_0 &= \frac{Y^{\frac{2}{3}}}{Y^{\frac{2}{3}} + |\mathcal{U}_L|}n \\
n_i &= \frac{1}{Y^{\frac{2}{3}} + |\mathcal{U}_L|}n, \;\; u_i \in \mathcal{U}_L
\end{aligned}$$

□

Algorithm 1

input: the number of accessible tuples (N_i) for each user
output: optimal user grouping $\{\mathcal{U}_L, \mathcal{U}_H\}$

```
1:  Sort the users by their number of accessible tuples
        in ascending order into list $\{u_i \; i \in (1,\dots,U)\}$;
2:  $min = \infty$, $G_{opt} = \emptyset$;
3:  for $j = 0$ to $U$
4:          $\mathcal{U}_L = \{u_i, i \in (1,\dots,j-1)\}$;
            $\mathcal{U}_H = \{u_i, i \in (j,\dots,U)\}$;
5:          $val = j + \left(\sum_{u_i\in\mathcal{U}_H}\sqrt{\frac{N}{N_i}}\right)^{\frac{2}{3}}$
6:          if $(val < min)$
7:                  $min = val$;
                    $G_{opt} = \{\mathcal{U}_L, \mathcal{U}_H\}$;
8:  return $G_{opt}$;
```

With Theorem 4.1, we next consider how to partition the users so as to minimize estimation error. To do this, we use Algorithm 1, which checks all possible partitions having the property that no user in $\mathcal{U}_L$ has access to more tuples than any user in $\mathcal{U}_H$. There are $(U+1)$ such partitions to be checked.

Theorem 4.2. *The user groups returned by Algorithm 1 minimize the mean estimation error bound $\hat{\epsilon}_{mean}$ over all possible user-groupings.*

Proof. Using Theorem 4.1 and Equation 10, we can write the mean total estimation error of a given user grouping as:

$$\begin{aligned}
& \beta\left(\sum_{u_i \in \mathcal{U}_L} \sqrt{\frac{Y^{\frac{2}{3}} + |\mathcal{U}_L|}{n}} + \sum_{u_i \in \mathcal{U}_H} \sqrt{\frac{N}{N_i}} \sqrt{\frac{Y^{\frac{2}{3}} + |\mathcal{U}_L|}{nY^{\frac{2}{3}}}}\right) \\
= \quad & \beta\sqrt{\frac{Y^{\frac{2}{3}} + |\mathcal{U}_L|}{n}} \left(|\mathcal{U}_L| + \frac{\sum_{u_i \in \mathcal{U}_H} \sqrt{\frac{N}{N_i}}}{\sqrt{Y^{\frac{2}{3}}}}\right) \\
= \quad & \beta\sqrt{\frac{Y^{\frac{2}{3}} + |\mathcal{U}_L|}{n}} \quad |\mathcal{U}_L| + \frac{Y}{Y^{\frac{1}{3}}} \\
= \quad & \beta\frac{|\mathcal{U}_L| + Y^{\frac{2}{3}} \quad ^{\frac{3}{2}}}{\sqrt{n}}
\end{aligned}$$

Because both n and β are constants and the function $F(x) = x^{\frac{3}{2}}$ is monotone, minimizing the above expression is equivalent to minimizing

$$|\mathcal{U}_L| + \left(\sum_{u_i \in \mathcal{U}_H} \sqrt{\frac{N}{N_i}}\right)^{\frac{2}{3}} \tag{11}$$

Suppose the user-grouping returned by Algorithm 1 is:

$$\mathcal{U}_L = \{u_1, u_2, \ldots, u_j\},\ \mathcal{U}_H = \{u_{j+1}, u_{j+2}, \ldots, u_U\}$$

We use E to denote the value of Formula 11 for this user-grouping. Now suppose there is another user-grouping as follows:

$$\mathcal{U}'_L = \{u_{p_1}, u_{p_2}, \ldots, u_{p_l}\},\ \mathcal{U}'_H = \{u_{p_{l+1}}, u_{p_{l+2}}, \ldots, u_{p_U}\}$$

We use E' to denote the value of Formula 11 from this user-grouping. Assume $E' < E$. We will show that this assumption leads to a contradiction.

From Equation 11, we have:

$$\begin{aligned}
E &= j + \left(\sum_{i \in (j+1, \ldots, U)} \sqrt{\frac{N}{N_i}}\right)^{\frac{2}{3}} \\
E' &= l + \left(\sum_{i \in (p_{l+1}, \ldots, p_U)} \sqrt{\frac{N}{N_i}}\right)^{\frac{2}{3}}
\end{aligned}$$

Now suppose that we sort all of the users in descending order of the number of tuples to which they have access, and we consider a configuration with

$\mathcal{U}_H$consisting of the first l users, i.e., the users that can access the most tuples. Let E'' represent the value of Equation 11 under this configuration:

$$E'' = l + \left(\sum_{i \in (l+1,\ldots,U)} \sqrt{\frac{N}{N_i}} \right)^{\frac{2}{3}}$$

Because $\mathcal{U}_H$ consists of the l users with access to the most tuples, we have $E'' \leq E'$. We also have $E \leq E''$ because both of those configurations are considered by Algorithm 1, which returns the configuration with the lowest error bound among those that it considers. Therefore $E \leq E'$, a contradiction. □

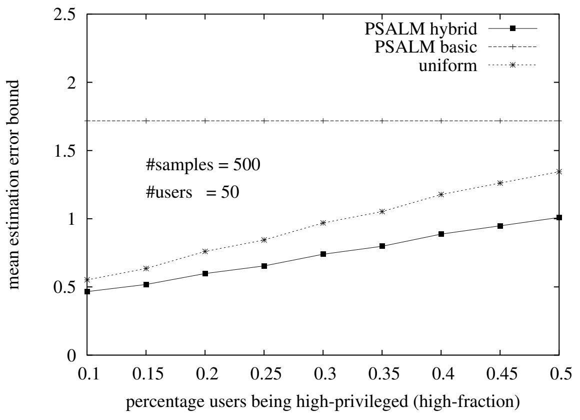
By partitioning the users into groups according to Algorithm 1 and choosing sample sizes according to Theorem 4.1, we can minimize the estimation error from hybrid PSALM. Because both the single-sample estimation technique and the basic PSALM technique from Section 3.2 are special cases of hybrid PSALM, we have the following corollary:

Corollary 1. *Hybrid PSALM under the optimal user-grouping from Algorithm 1 and the sample size assignment from Theorem 4.1 has a mean estimation error bound no greater than the mean estimation error bound of single-sample estimation and no greater than the mean estimation error bound of basic PSALM estimation.*

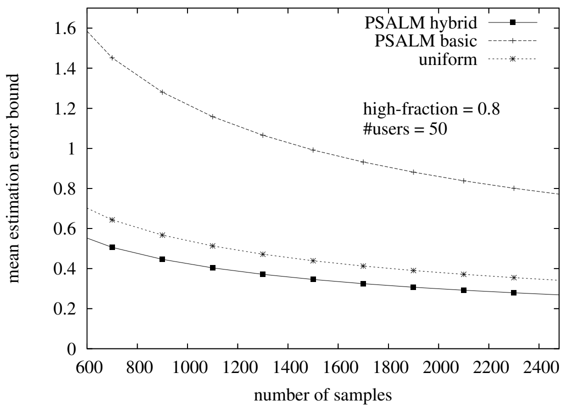
4.2 Evaluation of Hybrid PSALM Under Mean Estimation Error Bound

Figure 1 illustrates the mean 95%-confidence estimation error bounds for the single-sample, basic PSALM and hybrid PSALM approaches under a variety of conditions. These curves are determined by Equations 1 and 2 for the single-sample and basic PSALM approaches, and by Equations 7 and 9 for hybrid PSALM. To obtain these curves, we varied the total number of users (U) and the total sample size (n) as shown in the Figure. We chose the number of tuples accessible to each user (N_i) by choosing some percentage of users as high-privileged users ($\mathcal{U}_H$), and the remaining users as low-privileged users ($\mathcal{U}_L$). The percentage of high-privileged users is denoted as *high-fraction* in the experiments. Each high-privileged user has 50% of the data accessible to him while each low-privileged user has 1% of the data accessible to him. Finally, we fixed the selectivity of the query predicate P_Q at 50%, and assumed independence between P_Q and the users' access control predicates. Varying the selectivity of P_Q rescales the reported mean estimation error bounds, but does not affect the shape of the curves.

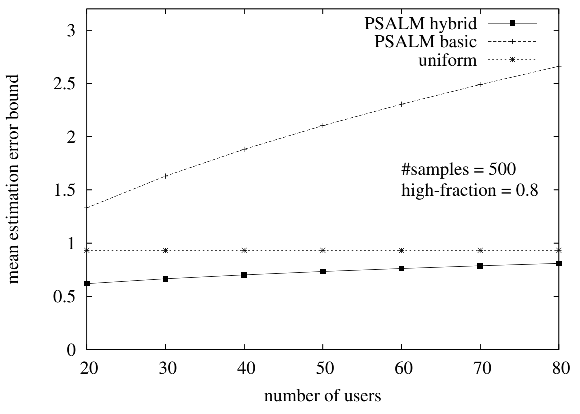
Figure 1(a) shows the behavior of the sampling techniques as the percentage of high-privileged users varies. The basic PSALM technique is insensitive to this parameter, while the performance of the single-sample approach gets better as we tend towards more high-privilege users and fewer low-privilege users. This is because the single-sample technique is more effective for high-privilege users



(a)



(b)



(c)

Figure 1: Mean estimation error bound between different sample techniques

than for low-privilege users. The hybrid PSALM approach outperforms both of the simpler alternatives. Figure 1(b) shows that all of the techniques are able to exploit an increasing sample space budget to improve accuracy. Finally, Figure 1(c) shows estimation error as a function of the number of users, with the sample budget fixed at 500 tuples. The single-sample technique is relatively insensitive to the number of users, while both PSALM techniques do best when the number of users is small. The basic PSALM and single-sample techniques have a crossover point, as shown in Theorem 3.2. However, the hybrid PSALM technique performs at least as well as either simpler technique regardless of the number of users.

We also used simulation analysis to study the actual mean total estimation error (ϵ) achieved by the hybrid PSALM and single-sample approaches. In our simulations, we assumed $N = 100,000$ tuples in the target relation, $U = 100$ users, and a total sample cardinality of $n = 500$ tuples. We first assigned a target access control cardinality (N_i) for each user according to a Zipfian distribution with parameter $z = 1.2$. We then randomly and independently determined the accessibility of each tuple to each user such that the expected number of tuples accessible to each user would be N_i. Finally, we drew samples of these tuples for cardinality estimation as prescribed by the single-sample and hybrid PSALM techniques.

We then simulated a series of queries. For each query, we assumed a query predicate selectivity of 50%, plus or minus a small, normally-distributed perturbation. Based on this selectivity, we randomly and independently determined which tuples satisfied the query predicate. Once the qualifying tuples were determined, we calculated the actual query result cardinality (C_i) as well as the sample-based estimated cardinality ($\tilde{C_i}$) under each of the two estimation techniques for each user. We then calculated the mean estimation error (ϵ) for each technique using

$$\epsilon = \frac{\sum_{i=1}^{U} \epsilon_i}{U}$$

Each point in Figure 2 shows the mean estimation error of a single query under hybrid PSALM as a function of the mean estimation error under the single-sample approach. In almost all cases, hybrid PSALM resulted in a lower mean estimation error than the single-sample approach.

4.3 Minimize the Maximum Estimation Error Bound

Now we are trying to find the user-grouping and sample quota assignment to minimize the maximum estimation error bound among all users. We do this using a different approach. Instead of minimizing the error bound, we try to *minimize the size of all samples required such that the maximum estimation error is bounded by a given* $\hat{\epsilon}$. For example, a single uniform sampling requires at least the following sample size to bound its maximum estimation error under $\hat{\epsilon}$:

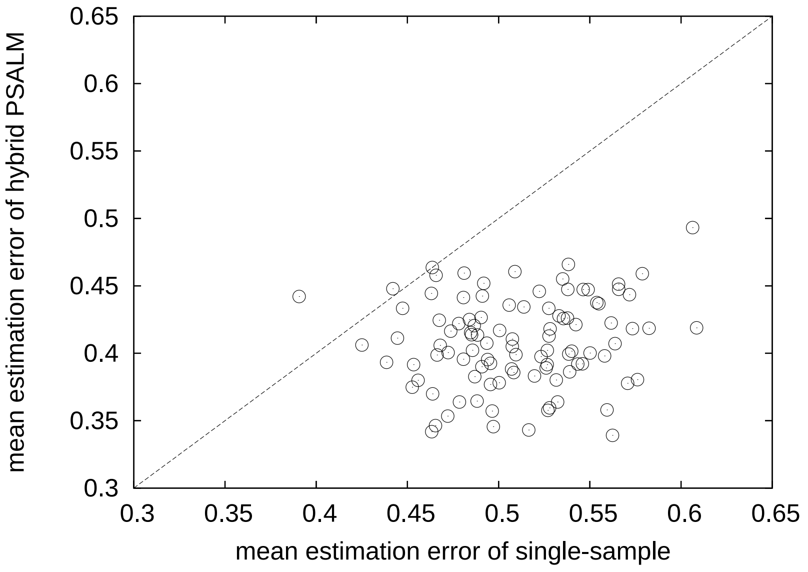


Figure 2: Mean estimation error between hybrid PSALM and single uniform sampling

$$\max_{i=1}^{U} \frac{4N}{\hat{\epsilon}^2 \cdot C_i} \log \frac{2}{\Delta}$$

We first have the following on the sample size requirement for users in $\mathcal{U}_H$ and users in $\mathcal{U}_L$ such that their maximum estimation errors are all bounded by $\hat{\epsilon}$.

$$\begin{aligned} n_0 &\geq \max_{u_i \in \mathcal{U}_H} \left(\frac{4N}{\hat{\epsilon}^2 \cdot C_i} \cdot \log \frac{2}{\Delta}\right), \quad u_i \in \mathcal{U}_H \\ n_i &\geq \frac{4N_i}{\hat{\epsilon}^2 \cdot C_i} \cdot \log \frac{2}{\Delta}, \quad u_i \in \mathcal{U}_L \end{aligned}$$

Similar to the analysis in minimizing the mean estimation error bound, assuming P_Q is independent of the access control predicates P_{AC}, the minimal sample size to bound the maximum estimation error from all users under $\hat{\epsilon}$ is:

$$n_0 + \sum_{u_i \in \mathcal{U}_L} n_i = \frac{\gamma}{\hat{\epsilon}^2} \cdot \left(\max_{u_i \in \mathcal{U}_H} \left(\frac{N}{N_i}\right) + |\mathcal{U}_L|\right) \tag{12}$$

Here γ is a constant. To minimize the total sample space as Formula 12, we sort all users by their number of accessible tuples in ascending order into a list $\{u_1, u_2, \ldots, u_U\}$. Then we segment the sorted list at different position ranging from 0 to U. The segmentation $\{u_1, \ldots, u_j\}$ and $\{u_{j+1}, \ldots, u_U\}$ corresponds to a user-grouping into $\mathcal{U}_L$ and $\mathcal{U}_H$. At each user-grouping there is a corresponding total sample space computed from Formula 12. We pick the segmentation that minimizes Formula 12, and this segmentation corresponds to the grouping of users that has the smallest total sample size to have the maximum estimation error bound under $\hat{\epsilon}$. The proof is very similar to the one in Section 4.1, and is omitted.

Under the optimal user grouping, the optimal sample quota assignment among all users is the following (we use Z to represent $\max_{u_i \in \mathcal{U}_H} \frac{N}{N_i}$ for brevity):

$$\begin{aligned} n_0 &= \frac{Z}{|\mathcal{U}_L| + Z} \cdot n, \\ n_i &= \frac{1}{|\mathcal{U}_L| + Z} \cdot n, \ u_i \in \mathcal{U}_L \end{aligned}$$

Under this optimal user-grouping and sample quota assignment, the maximum estimation error bound will be:

$$\sqrt{4 \cdot \log \frac{2}{\Delta}} \cdot (\max\, (\max_{u_i \in \mathcal{U}_L} \sqrt{\frac{N_i \cdot (|\mathcal{U}_L| + Z)}{n \cdot C_i}}, \ \max_{u_i \in \mathcal{U}_H} \sqrt{\frac{N \cdot (|\mathcal{U}_L| + Z)}{n \cdot C_i \cdot Z}} \ ) \ )$$

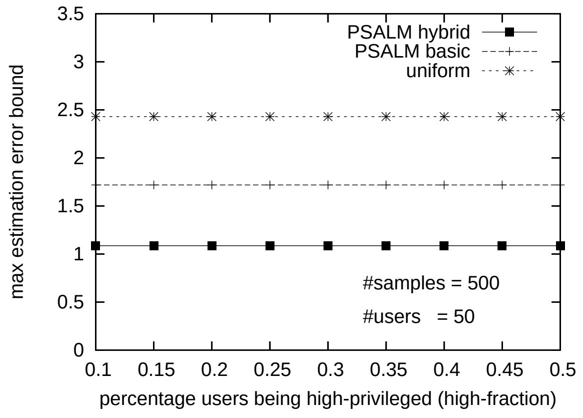
The 95%-confidence maximum estimation error bounds for the single-sample, the optimal basic PSALM and the optimal hybrid PSALM are depicted in Figure 3 under the same settings as in Section 4.2. The optimal hybrid PSALM always performs at least as well as either simpler technique regardless of the parameters. Please note that in Figure 3(c) that hybrid PSALM degrade to basic PSALM once the number of users exceeds 150, and thus the two schemes are in fact identical at this point.

5 Exploiting Access Privilege Correlation

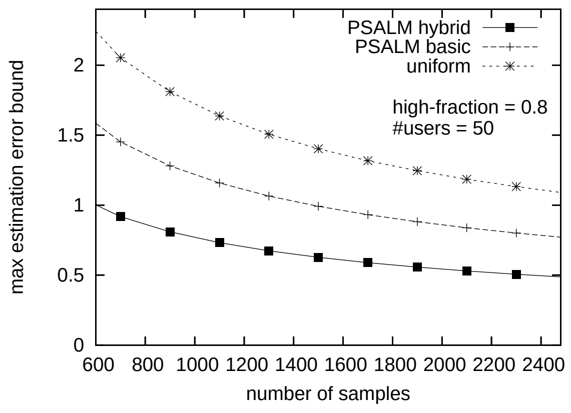
In previous work [9] involving access control data, we observed strong correlations in access privileges among different users. The existence of such correlations is not surprising. For example, in the common situation in which there are "private" tuples that are accessible to only a single user, or to a few users, all of the remaining users will agree that such tuples are inaccessible. In this section, we explore an enhancement to the hybrid PSALM technique described in Section 4 that exploits such correlations.

Suppose that we use the hybrid PSALM algorithm to partition users into $\mathcal{U}_L$ and $\mathcal{U}_H$ and to assign sample sizes n_L for users in $\mathcal{U}_L$ and n_0 for users in $\mathcal{U}_H$. The enhancement improves the accuracy of cardinality estimates by exploiting access privilege correlations among the users in $\mathcal{U}_L$. The approach is to identify groups of low-privilege users that have correlated access privileges, using a technique that will be described shortly. Suppose that $G \subseteq \mathcal{U}_L$ is such a group. Instead of creating a separate, private sample for each user in G, we create a single, combined sample of size $|G|n_L$ that is shared by all users in G. Here n_L is the private sample size for the users in $\mathcal{U}_L$, and is a consistent value for all such users in G according to Theorem 4.1.

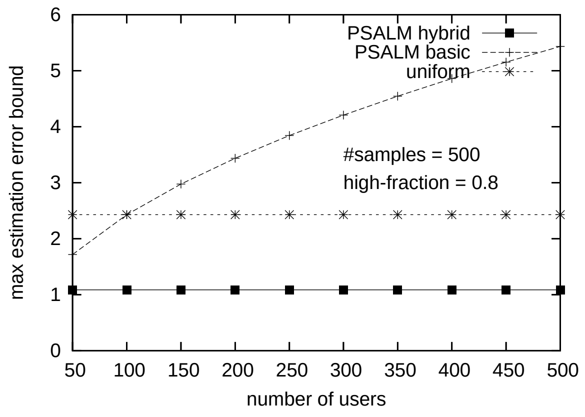
This shared sample is a simple random sample drawn from among all tuples that satisfy P_{ACG}, defined as



(a)



(b)



(c)

Figure 3: Maximum estimation error bound between the different techniques

$$\bigvee_{i \in G} P_{ACi}$$

That is, we sample from among those tuples that are accessible to at least one user in the group. To compute an unbiased cardinality estimate for predicate P_Q for user $u_i \in G$, we use

$$\tilde{C}_i = c_i \frac{N_G}{|G| n_L} \tag{13}$$

where c_i is the number of sample tuples matching $P_Q \wedge P_{ACi}$ and N_G is the number of tuples in the target relation that satisfy P_{ACG}. This results in a Δ-confidence estimation error bound of

$$\epsilon_i \leq \sqrt{\frac{4N_G}{|G| n_L C_i} \log \frac{2}{\Delta}} \tag{14}$$

for all users in G.

Grouping users in this way does not change the cardinality estimates of users outside the group, nor does it affect the accuracy of those estimates. Thus, any change in the total estimation error bound from all users is from the change of the estimation error bound of the users in G. By combining Equations 2 and 14, we can see that user grouping will result in lower total estimation error bound among the users in the group if

$$\sum_{i \in G} \sqrt{\frac{4N_G}{|G| n_L C_i} \log \frac{2}{\Delta}} < \sum_{i \in G} \sqrt{\frac{4N_i}{n_i C_i} \log \frac{2}{\Delta}}$$

Simplifying, we find that this inequality holds provided that

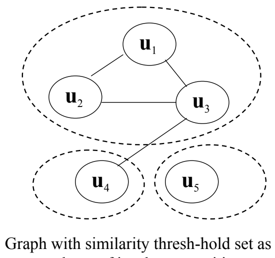
$$\sum_{i \in G} \sqrt{\frac{N_G}{|G| C_i}} < \sum_{i \in G} \sqrt{\frac{N_i}{C_i}}$$

Similarly, user grouping will reduce the maximum estimation error bound from all users in the group if

$$\max_{i \in G} \sqrt{\frac{N_G}{|G| C_i}} < \max_{i \in G} \sqrt{\frac{N_i}{C_i}}$$

When there is little overlap among the accessible tuples of the grouped users, then N_G approaches $\sum_{i \in G} N_i$ and there is no benefit to grouping users to reduce the total estimation error bound or the maximum estimation error bound. However, if the grouped users have most of their accessible tuples in common, then N_G is much smaller and grouping will result in improved estimates for all users in the group because of the larger size of the group's sample. Thus, to form groups of users we should identify users that have correlated access rights and group them together. In the following, we describe how we group the users together.

	u_1	u_2	u_3	u_4	u_5
t_1	1	1	1	0	0
t_2	0	0	0	1	1
t_3	1	1	1	1	0
t_4	1	1	1	1	0
t_5	0	0	0	0	1
t_6	0	1	0	0	1
t_7	1	1	1	0	0
t_8	0	1	1	1	0



Graph with similarity thresh-hold set as 3 and one of its cluster partition

CLUSTER	SAMPLE	
u_1 u_2 u_3	t_1 t_4 t_7	
u_4	t_2 t_3	
u_5	t_2 t_5	

Figure 4: An example of user grouping

5.1 Grouping Correlated Users

To determine which users to group together, we first define a pairwise similarity function, SIM, over the users in $\mathcal{U}_L$. The similarity between the i^{th} and j^{th} users is defined as follows

$$SIM(i,j) = \min\left(\frac{N_{i,j}}{N_i}, \frac{N_{i,j}}{N_j}\right) \tag{15}$$

Here $N_{i,j}$ is defined as the number of tuples from the target relation that are accessible to both user i and user j. Using this function SIM, we can define a *similarity graph* as an undirected graph with one node for each user in $\mathcal{U}_L$. There is an edge between user i and user j if and only if $SIM(i,j) \geq \theta$, where θ is a tunable parameter of the grouping algorithm.

To place the users into groups, we attempt to find a *minimum clique partition* [16] of the user similarity graph. This identifies a set of non-overlapping cliques that, together, cover the entire user similarity graph. Each such clique becomes one of the user groups for which we will create a sample. By using cliques as sampling groups, we ensure that all users in a group have pairwise-similar access rights. By minimizing the number of cliques we minimize the number of separate samples that are required, thus allowing us to use larger samples while remaining within the space budget.

The problem of finding a minimum clique partition is NP-complete [16]. Thus, we use a fast greedy algorithm to partition the graph. The algorithm produces a set of non-overlapping cliques that cover the user similarity graph, but the set is not guaranteed to have minimum cardinality. The greedy algorithm first finds a maximal clique in the graph by starting from a random node, finding the clique around the node, and removing all nodes in the clique and all edges induced by these nodes. This process continues until there are no more nodes. We repeat this greedy algorithm several times from different initial nodes and record the smallest clique partition at all iterations of this algorithm.

The example in Figure 4 illustrates the process of identifying user groups.

The example illustrates a scenario in which there are five users $(u_1, u_2, \ldots, u_5)$ and eight tuples $(t_1, t_2, \ldots, t_8)$. The large matrix indicates which tuples are accessible to each user, with a 1 indicating accessibility. Using a similarity threshold θ of 0.5, we obtain the user similarity graph with pairwise connections among u_1, u_2 and u_3, as shown in Figure 4. A minimum clique partition for this graph is illustrated using dashed lines.

5.2 Cost of Exploiting Correlation

Once user groups have been defined and the corresponding samples have been drawn, the cost of estimating query cardinalities is essentially the same whether low-privilege users are grouped or not. In either case, a predicate is evaluated against each tuple in the appropriate sample and an estimate is computed using either Equation 8 or Equation 13. Similarly, all of the necessary samples can be drawn in a single pass over the target relation, regardless of whether grouping is used.

There is an additional cost associated with determining how users should be grouped together. Since grouping depends only the access rights of the various users, and not on the query predicates, user groups will not need to change unless access privileges are redefined. We assume that this will occur rarely with respect to query evaluation.

To group users, it is necessary to obtain the value of $N_{i,j}$ for all pairs of users in $\mathcal{U}_L$, for use in determining $SIM(i, j)$ (Equation 15). This can be accomplished in one scan of the target relation. In addition, once the SIM function is known and user groups have been selected, it is necessary to obtain the value of N_G for each selected user group G. This is necessary for obtaining cardinality estimates for users in G using Equation 14. However, N_G can be computed during the same scan used by the PSALM algorithm to determine N_i for users that are not part of any group. Thus, exploiting access privileges increases the cost of PSALM from one scan of each target relation to two scans, one for determining $N_{i,j}$ of all user pairs, and one for drawing samples and determining N_G or N_i for each individual user or user group for which a sample is drawn.

5.3 Effectiveness of User-Grouping

As a test of the procedure for forming user groups, we applied the procedure to a snapshot of the access control data from a production instance of a multi-user content management system. The system had approximately 370000 access-controlled objects and a total of 1584 users.

We applied our procedure to these data to determine user groups, using a similarity threshold θ of 0.7. We randomly selected sets of users from among all 1584 users, and we applied our user grouping algorithm to the selected users. We experiment with different number of users in $\mathcal{U}_L$. Figure 5 shows the number of groups (or individual samples) required after applying the user-grouping on similar access controls. These data show that the user grouping algorithm reduces the number of samples significantly.

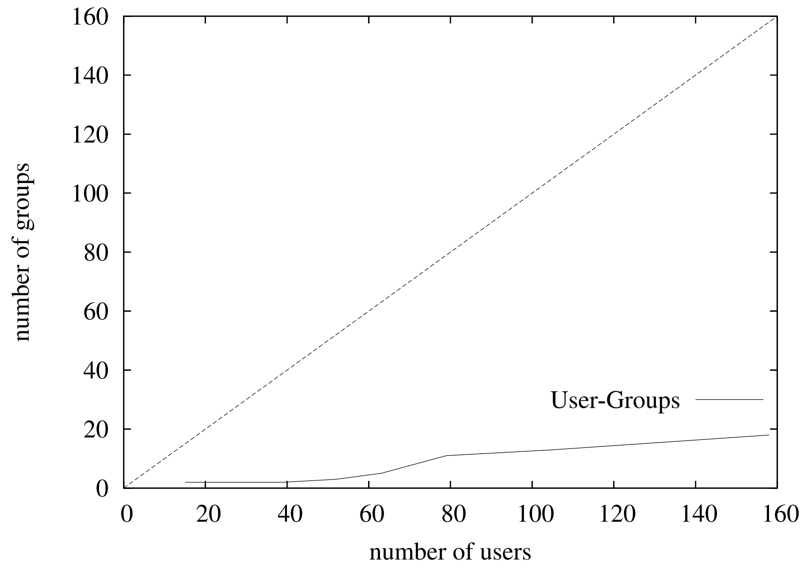


Figure 5: Effect of grouping on the number of samples required for low-privilege users

6 Cardinality Estimation Under Role-based Access Control

Fine-grained access controls are often *not* explicitly specified for each user in the system. Instead, an administrator defines a number of *roles*, and assigns access rights to these roles.[1] Roles, in turn, are assigned to users, in which case all access rights of the role are granted to the user. Roles can also be assigned to other roles. This defines a hierarchy of users and roles, as illustrated in Figure 6, where each user obtains access rights from all the roles to which he is directly or indirectly assigned. In other words, the set of accessible tuples for a user is the *union* of the tuples that are accessible to that user's ancestors in the role hierarchy. For example, the user in Figure 6 obtains access rights from three roles R_0, R_1, R_2.

Suppose a user is authorized with k roles directly or indirectly from the role hierarchy, and each role R_i has access control predicate P_{ACi} on a target relation I. The user's access control predicates on the target relation is thus $(\bigvee_{i\in\{1,\ldots,k\}} P_{ACi})$. Given query selection predicate P_Q, we are trying to estimate the cardinality of tuples matching $P_Q \wedge (\bigvee_{i\in\{1,\ldots,k\}} AC_i)$ from the target relation.

A straightforward approaches to compute the estimated cardinality is to compute the *effective access controls* for each user from all his roles, and apply PSALM sampling scheme for all the users. However, if there are many users with distinct access controls, the individual samples may become extremely small. On the other hand, the users have different effective access controls

[1]User-groups are similar to roles and we only use the term *role* here to denote similar implementation using user groups.

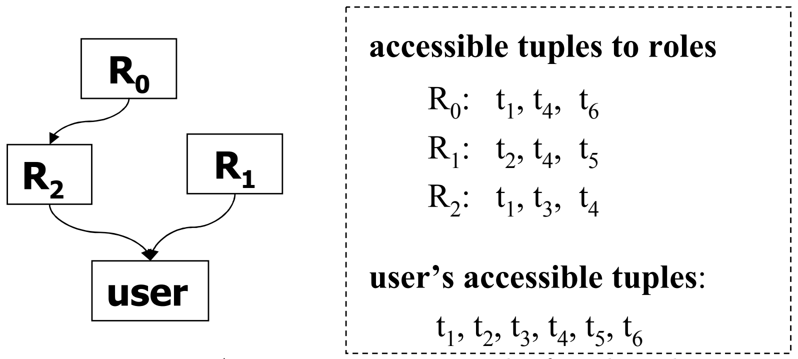


Figure 6: A user gets access to tuples from his roles

simply because they are granted with *different combinations* of roles. If we build PSALM samples based on roles, we will have fewer, larger individual samples. Therefore, our goal is to accurately estimate the number of accessible tuples satisfying a given query predicate by applying the PSALM sampling scheme to the roles.

We first apply the distributive law to rewrite the cardinality of matching tuples:

$$P_Q \wedge (\bigvee_{i \in \{1,\ldots,k\}} P_{ACi}) \equiv \bigvee_{i \in \{1,\ldots,k\}} (P_Q \wedge P_{ACi}) \tag{16}$$

We are able to apply PSALM sampling on the roles to estimate the cardinality of each $P_Q \wedge P_{ACi}(I)$ accurately. To the best of our knowledge, there is no effective approach to directly estimate the cardinality of disjunctive predicates on a relation. One approach used in PostgreSQL is to estimate cardinality of disjuncts by rewriting them into a conjunctive form. For example, the cardinality of the disjunctive predicate $(P_1 \vee P_2 \vee P_3)$ on relation I can be estimated as follows:

$$\begin{aligned} & |(P_1 \vee P_2 \vee P_3)(I)| \\ = \; & |P_1(I)| + |P_2(I)| + |P_3(I)| - \\ & \quad |(P_1 \wedge P_2)(I)| - |(P_1 \wedge P_3)(I)| - |(P_2 \wedge P_3)(I)| + \\ & \quad |(P_1 \wedge P_2 \wedge P_3)(I)| \end{aligned}$$

where the cardinality of each conjunct is estimating separately. However, one significant disadvantage of this approach is that the number of terms in the formula is exponential in the size of the disjunctions. In the next section, we present an alternative approach that solves this problem.

6.1 Coverage Algorithm for Union of Sets

We use the Coverage Algorithm [13] to estimate cardinality of disjunctive predicates. This approach takes time polynomial in the number of conjunctions in a disjunctive formulae, has more accurate estimates than the approach used in PostgreSQL, and can be built on top of PSALM. Coverage Algorithm is able to accurately estimate the size of union $\bigcup_{i\in(1,\ldots,k)} C_i$, provided that [13]:

1. we can accurately estimate the cardinality of each set C_i, and
2. we can sample uniformly at random from each C_i, and
3. we can determine in polynomial time whether a given tuple belongs to C_i.

The Coverage Algorithm takes as input the sets $\mathcal{C} = \{C_i, i \in \{1,\ldots,k\}\}$ and proceeds as in Algorithm 2. The algorithm starts with a counter W with value zero, and a total sample size n. It then samples from each of the sets C_i, where the number of samples taken from C_i is proportional to $|C_i|$. For each sample from C_i, we examine if it belongs to any C_j such that $j \in \{1,\ldots,i-1\}$. If it does *not* belong to any of these sets, we increment the counter W by one.

Algorithm 2 Coverage Algorithm [13]

```
COVERAGE($\mathcal{C}, n$)
1: set counter $W = 0$;
2: sort sets in $\mathcal{C}$ by their sizes in descending order
3: for each $C_i \in \mathcal{C}$
4:         sample $\frac{n|C_i|}{\sum_{C\in\mathcal{C}}|C|}$ tuples uniformly from $C_i$
5:         for each tuple $t$ sampled
6:                 if $t \notin C_j, j \in \{1,\ldots,i-1\}$
7:                 $W++$;
8: return $W$;
```

After checking all the samples from $C_1, C_2 \ldots, C_k$, we take the counter W from this routine and compute estimated cardinality $|C| = |\bigcup_{i\in(1,\ldots,k)} C_i|$ as follows:

$$|C| = \sum_{i\in\{1,\ldots,k\}} |C_i| \frac{W}{N} \tag{17}$$

Theorem 6.1. *[13] The sampling procedure in Algorithm 2 and Formula 17 yields an ϵ-approximation to $|C|$ with $(1-\Delta)$ probability, provided sample size $n \geq \frac{4k}{\epsilon^2}\ln\frac{2}{\Delta}$.*

6.2 Applying The Coverage Algorithm

We now show how to apply the Coverage Algorithm for estimating cardinality of disjunctive predicates in the form shown in Equation 16. For a given target relation I, we need to show that we are able to:

1. estimate the size of each $P_Q \wedge P_{ACi}(I)$, and

2. sample uniformly from each $P_Q \wedge P_{ACi}(I)$, and
3. decide whether a given tuple satisfies $P_Q \wedge P_{ACi}$ in polynomial time.

The first requirement is already fulfilled by $PSALM$. The third requirement can be satisfied if the predicate does not involve exponential computation. For access control predicates, this requirement is easily satisfied. For the second requirement, we only need to sample uniformly from the sample tuples of role R_i that satisfy P_Q. One way to accomplish this by a single scan through the sample tuples of role R_i, and reservoir-sampling those tuples that satisfy P_Q. We have the following Lemma:

Lemma 6.1. *Reservoir-sampling on the sample tuples of role R_i that satisfy $P_Q$generates a uniform sampling of $P_Q \wedge P_{ACi}$.*

Proof. We use D_i to denote the reservoir from role R_i. We first have that each tuple in $P_Q(D_i)$ is also in $P_Q \wedge P_{ACi}$. We also know that a tuple in $P_Q \wedge P_{ACi}$ has probability $\frac{|P_Q \wedge P_{ACi}|}{N_i}$ to appear in D_i, and this probability is independent of other tuples in $P_Q \wedge P_{ACi}$. Hence the proof. □

6.3 Evaluating the Coverage Algorithm

To evaluate the Coverage Algorithm, we used a data set that describes expenditures of American families [2]. The data include 127931 tuples, with each tuple representing a family' expenses on insurance, property tax, electricity, gas, water, and fuel (six attributes). We choose this data set since there is some correlation between the different attributes, and we can model access controls and queries as being predicates on different attributes.

To model access controls, we first create a number of roles. The access control of each role is a single predicate P_{AC} on one of the six attributes with selectivity of 0.25, i.e., each role corresponds to a predicate that selects a quarter of the total tuples. For each database user, we assign him with four roles, thus his accessible tuples are the disjunctions of the corresponding four predicates from the roles. The query is another single attribute predicate P_Q with the same selectivity (0.25) on one of the six attributes.

We then compare the Coverage Algorithm and a single uniform sampling for estimating the cardinality of tuples matching both the query and the access controls for each user, i.e., $(P_Q \wedge \bigvee_{i \in \{1,\ldots,4\}} P_{ACi})$. The Coverage Algorithm is applied on PSALM basic scheme which has the same total size as the single uniform sampling.

Figure 7 shows the estimation error of the Coverage Algorithm and the uniform sampling, where both techniques use a sample size of 60 tuple. In another word, the Coverage Algorithm is applied on four individual samples with 15 tuples each. Each dot in the figure represents a query. The x-coordinate denotes the estimation error of the uniform sampling and the y-coordinate denotes the estimation error of the Coverage Algorithm. We see that the Coverage Algorithm out-performs the uniform approach significantly.

[2]available from http://www.ipums.org

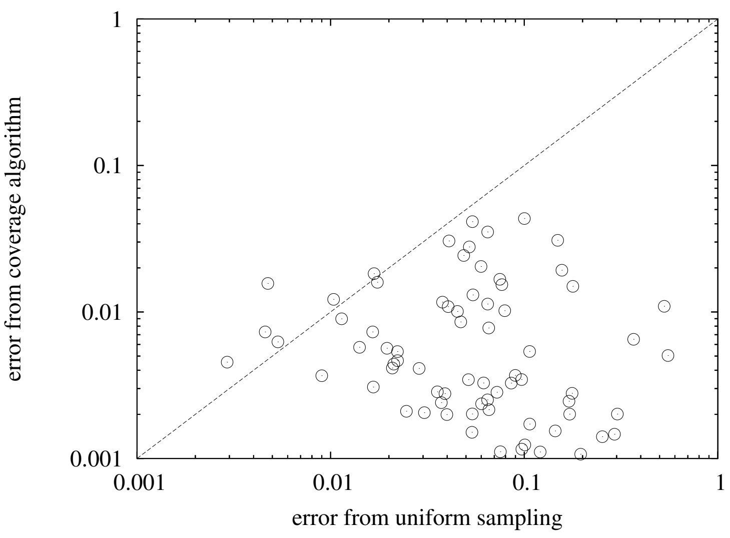


Figure 7: Performance of Coverage Algorithm versus uniform sampling

7 Related Work

While there exists a broad literature on cardinality estimation for query optimization, we focus our attention on sampling-based techniques. Sampling is widely used for estimating the cardinality of intermediate query results from joins and selection queries. Because we view access controls as predicates, we are primarily interested in the latter.

There are two main approaches to sampling data for cardinality estimation. One is to sample for each query at runtime, either proactively before executing the query [5], or reactively after executing the query [1]. The alternative is to sample off-line without prior knowledge of the queries. Although PSALM takes advantage of prior knowledge access controls, it is closer to the off-line sampling category since it does not make any assumptions about the users' queries.

Kolmogorov's statistics show that a moderately-sized sample gives accurate selectivity estimation for queries, and that the sample size does not depend on the size of the underlying dataset [6]. However, that evaluation is based on the relative error of selectivity, rather than the relative error of cardinality. It is well-known that uniform random sampling does not provide accurate cardinality estimation when data distribution is highly skewed or the queries results are very small. Therefore, instead of specifying a fixed sample size for all queries, adaptive sampling [11](known as sequential sampling [8]) iteratively samples from data until the accuracy of the estimation satisfies a stopping rule. This approach falls into the runtime sampling category.

Another approach to skewed data or highly selective queries is to sample without uniformity. This includes biased sampling [4] or stratified sampling [3], whose idea is to partition data into non-overlapping clusters of different density, and then assign different weights to the sample tuples from clusters of different density. The density distributions are either estimated through a pilot sampling phase, or from prior knowledge of the queries together with some statistics on the data.

The work by Acharya, Gibbons and Poosala [2] is the most similar to PSALM. Their sampling mechanism is intended for efficient approximate answering of aggregation queries, and their approach is to partition data according to the prior knowledge of grouping attributes, and judiciously assign sample quota among all the partitions to minimize estimation variance. However, their approach, like those of other biased or stratified sampling techniques, assumes that the data are partitioned into non-overlapping sets for sampling. In our setting, the set of tuples accessible to different users may overlap.

8 Conclusion

In this paper we propose a several novel multi-user sampling schemes for estimating query result cardinalities in the presence of fine-grained access controls. These schemes leverage prior knowledge of users' access controls, which a relatively static. Compared with simple uniform sampling, our hybrid PSALM approach provides more accurate cardinality estimates.

References

[1] Ashraf Aboulnaga, Peter J. Haas, Sam Lightstone, Guy M. Lohman, Volker Markl, Ivan Popivanov, and Vijashankar Raman. Automated Statistics Collection in DB2 Stinger. In *Proc. 33th Int. Conf. on Very Large Data Bases*, 2007.

[2] Swarup Acharya, Phillip B. Gibbons, and Viswanath Poosala. Congressional Samples for Approximate Answering of Group-by Queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 487–498, 2000.

[3] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2001.

[4] Christopher R. Palmer and Christos Faloutsos. Density Biased Sampling: An Improved Method for Data Mining and Clustering. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 82–92, 2000.

[5] Amr El-Helw, Ihab F. Ilyas, Wing Lau, Volker Markl, and Calisto Zuzarte. Collecting and Maintaining Just-in-Time Statistics. In *Proc. 23st Int. Conf. on Data Engineering*, pages 516–525, 2007.

[6] Benjamin Epstein. Introduction to Statistical Analysis. *SIAM Review*, 1(1):75–77, 1959.

[7] Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. A Dip in the Reservoir: Maintaining Sample Synopses of Evolving Datasets. In *Proc. 32th Int. Conf. on Very Large Data Bases*, pages 595–606, 2006.

[8] Peter J. Haas and Arun N. Swami. Sequential Sampling Procedures for Query Size Estimation. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 341–350, 1992.

[9] Huaxin Zhang and Ning Zhang and Ken. Salem and Donghui Zhuo. Compact Access Control Labeling for Efficient Secure XML Query Evaluation. *Journal of Data and Knowledge Engineering*, 60(2):326–344, 2007.

[10] IBM. *DB2 Administration Guide: Implementation*, 2006.

[11] Richard J. Lipton and Jeffrey F. Naughton. Query Size Estimation by Adaptive Sampling. In *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems*, pages 40–46, 1990.

[12] Microsoft. *Implementing Row- and Cell-Level Security in Classified Databases Using SQL Server 2005*, 2005.

[13] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms.* Cambridge University Press, New York, NY, USA, 1995.

[14] Oracle. *Data Classification with Oracle Label Security: A White Paper.*

[15] Oracle. *The Virtual Private Database in Oracle9ir2.*

[16] P. Crescenzi and V. Kann. *A Compendium of NP Optimization Problems*, page 13. Springer Verlag, 1999.

[17] Shariq Rizvi, Alberto Mendelzon, S. Sudarshan, and Prasan Roy. Extending Query Rewriting Techniques Fine-Grained Access Control. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2004.

[18] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.