Privacy Preserving Data Generation for Database Application Performance Testing[⋆]

Yongge Wang, Xintao Wu, and Yuliang Zheng

UNC Charlotte, USA ({yonwang, xwu, yzheng}@uncc.edu)

Abstract. Synthetic data plays an important role in software testing. In this paper, we initiate the study of synthetic data generation models for the purpose of application software performance testing. In particular, we will discuss models for protecting privacy in synthetic data generations. Within this model, we investigate the feasibility and techniques for privacy preserving synthetic database generation that can be used for database application performance testing. The methodologies that we will present will be useful for general privacy preserving software performance testing.

1 Introduction

Functionality and performance testing is essential for software application development. Currently, two approaches dominate software application testing. With the first approach, application developers carry out their tests on their own local *development* synthetic synthetic data sets. Obviously this approach can not fulfill the requirements of all the testing phases if the synthetic data sets are of small size or do not reflect the real data sets. In particular, the performance could be significantly different if the synthetic data is not similar to the production databases. With the second approach, new applications are tested over *live production* databases. This approach cannot be applied in most situations due to the high risks of disclosure and incorrect updating of confidential information.

Recently, Wu, Wang, and Zheng [15] have proposed a general framework for privacy preserving database application testing by generating *synthetic* data sets based on some a-priori knowledge about the current *production* data sets. The generated data sets will be used to help software vendors to arrive at a close estimate of the performance of a software application. In this paper, we investigate the tradeoff of privacy preserving and performance preserving in detail. We also present an approach on privacy preserving data generation based on the generation location model. Specifically, our contributions include: (1) A model for quantifying the privacy leakage and application performance metric difference in the synthetic data set; (2) Some infeasibility results for privacy preserving synthetic data generation; (3) A heuristic approach for privacy preserving synthetic data generation within our model.

The current state of the art is that there has been little work dedicated to application software testing that achieves both goals: privacy preserving and performance

[⋆] This research was supported by USA National Science Foundation NSF CCR-0310974.

preserving. One related research area is the privacy preserving statistical databases [1, 5] which has developed methods to prevent the disclosure of confidential individual data while satisfying requests for aggregate information. Though the experience in statistical database research are useful for the study of privacy preserving application software testing, it does not consider the performance preserving issues. Furthermore, most statistical database literatures considered two types of indirect confidential information leakage [14]: re-identification disclosure and prediction disclosure. Re-identification disclosure occurs if an attacker is able to deduce the values of a sensitive attribute for a target individual after this individual has been re-identified. Prediction disclosure occurs if the data enable the attacker to predict the value of a sensitive attribute for some target individual with some degree of confidence. For example, Dinur and Nissim [5] discussed a model for the re-identification of some specific entries in the database. In a live production database, the confidential information is not limited to the re-identification of some specific entries. The statistical information or some rules/patterns about the production database are also considered as the confidential information.

Also related to our research is private information retrieval and privacy preserving data mining. The theoretical work of private information retrieval [3, 7] enables users to obtain information from databases while keeping their queries secret from the database managers. The objective of privacy-preserving data mining (e.g., distortion based approach [2, 6, 9]) is to prevent the disclosure of confidential individual values while preserving general patterns and rules. There have been some prior investigations into data generation [12, 11], however, the current data generation tools are built either for testing data mining algorithms or for assessing the performance of database management systems, rather than testing database applications. In addition, they lack the required flexibility to produce more realistic data needed for software testing and do not take into consideration privacy issues.

The organization of the paper is as follows. In Section 2, we present the general location model and two test statistics. In Section 3, we give a formal definition for the model of privacy preserving synthetic data generator. Section 4 describes a heuristic approach to privacy preserving synthetic data generators. Section 5 briefly addresses the applications of some statistical database results to privacy preserving synthetic data generation. Section 6 concludes the paper.

2 Preliminaries

2.1 The General Location Model

Let $A_1, A_2, \cdots, A_p$ denote a set of categorical attributes and $Z_1, Z_2, \cdots, Z_q$ a set of numerical ones in a table with n entries. Suppose A_j takes possible domain values $1, 2, \cdots, d_j$, the categorical data $\mathbf{W}$ can be summarized by a contingency table with total number of cells equal to $D = \prod_{j=1}^{p} d_j$. let $\mathbf{x} = \{x_d : d = 1, 2, \cdots, D\}$ denote the number of entries in each cell. Clearly $\sum_{d=1}^{D} x_d = n$.

The general location model [13] is defined in terms of the marginal distribution of $\mathbf{A}$ and the conditional distribution of $\mathbf{Z}$ given $\mathbf{A}$. The former is described by a multinomial distribution on the cell counts $\mathbf{x}$,

$$\mathbf{x} \mid \pi \sim M(n,\pi) = \frac{n!}{x_1! \cdots x_D!} {\pi_1}^{x_1} \cdots {\pi_D}^{x_D}$$

where $\pi = \{\pi_d : d = 1, 2, \cdots, D\}$ is an array of cell probabilities corresponding to $\mathbf{x}$. Given $\mathbf{A}$, the rows $z_1^T, z_2^T, \cdots, z_n^T$ of $\mathbf{Z}$ are then modeled as conditionally multivariate normal. We assume that

$$z_i \mid \mu_i = E_d, \mu_d, \Sigma \sim N(\mu_d, \Sigma)$$

is independent for $i = 1, 2, \cdots, n$, where μ_d is a q-vector means corresponding to cell d, and Σ is a $q \times q$ covariance matrix. The parameters of the generation location model can be written as $\theta = (\pi, \mu, \Sigma)$, where $\mu = (\mu_1, \mu_2, \cdots, \mu_D)^T$ is a $D \times q$ matrix of means. The maximum likelihood estimates of θ is as follows:

$$\hat{\pi}_d = \frac{x_d}{n}, \quad \hat{\mu}_d^T = x_d^{-1} \sum_{i \in B_d} z_i^T, \quad \text{and} \quad \hat{\Sigma} = \frac{1}{n} \sum_{d=1}^{D} \sum_{i \in B_d} (z_i - \hat{\mu}_d)(z_i - \hat{\mu}_d)^T \quad (1)$$

where $B_d = \{i : \mu_i = E_d\}$ is the set of all tuples belonging to cell d.

2.2 Test Statistics of Two Distributions

Test Staistics 1 *χ^2-test for two binned data sets, let x and x' be two data sets with the same number of bins D, the $chi-square$ probability function $Q(\chi^2 \mid \nu)$ is an incomplete gamma function with the degrees of freedom ν (equal to D), where*

$$\chi^2 = \sum_d \frac{\left(\sqrt{n'/n}x_d - \sqrt{n/n'}x'_d\right)^2}{x_d + x'_d}, \quad n = \sum_d x_d, \quad \textit{and} \quad n' = \sum_d x'_d$$

Test Staistics 2 *Kolmogorov-Smirnov test [4] for unbinned data sets, let z and z' be two continuous data sets with size n and n' respectively. The Kolmogorov-Smirnov test statistic G is the largest absolute deviation between two cumulative distribution functions $F(z)$ and $F(z')$:*

$$G = max_{z,z'}\{\mid F(z') - F(z) \mid\}$$

The significance level of an observed value of G (as a disproof of the null hypothesis that the distributions are the same) is given by

$$\text{Prob}(G > obs) = Q_{KS}([\sqrt{n_e} + 0.12 + 0.11/\sqrt{n_e}]G)$$

where $Q_{KS}(\lambda) = 2\sum_{j=1}^{\infty} (-1)^{j-1} e^{-2j^2\lambda^2}$ and $n_e = \frac{nn'}{n+n'}$

3 (γ, τ)-privacy preserving synthetic data generator

Notation. N is the set of natural numbers and R is the set of real numbers. $\Sigma = \{0, 1\}$ is the binary alphabet, Σ^* is the set of (finite) binary strings and Σ^n is the set of binary strings of length n. The length of a string x is denoted by $|x|$. For strings $x, y \in \Sigma^*$, xy is the concatenation of x and y. For a string $x \in \Sigma^*$ and an integer number $n \geq 0$, $x[0..n]$ denotes the initial segment of length $n + 1$ of x ($x[0..n] = x$ if $|x| \leq n + 1$) and $x[i]$ denotes the ith bit of x, i.e., $x[0..n] = x[0] \cdots x[n]$.

3.1 Definition

A database application software package $\mathcal{F}$ that needs to be tested can be regarded as a Turing machine defined on binary strings. Its input can be divided into two parts, the database part x (with size of MBs or GBs) and the test case input y (with a relatively small size). The test case input y could be a simple SQL command or a script file such as a collection of SQL commands. As it is problematic to test $\mathcal{F}$ on the production database x, here we choose to test $\mathcal{F}$ on some mock data x'.

Private information in a database x could be defined as a Turing computable function $\tau(x, i)$ where i is the index of the private information. In another word, the private information in a database x is a sequence of binary strings: $\tau(x, 1), \tau(x, 2), \tau(x, 3), \ldots$, $\tau(x, m)$.

Example 1 *Consider the following confidential information: "The average balance range of term deposits from Asian people in a specific zip code area". We translate this property to a binary string. For a 90-bit binary string z, let $z[0..19]$ represent the zip code, $z[20..29]$ represent the background of the people (e.g., 0000000001 for Asia, 0000000002 for British, etc.), $z[30..59]$ and $z[60..89]$ represent the term deposit average balance lower bound and upper bound respectively for the people from $z[20..29]$ background and $z[0..19]$ zip code area. The above private information could be indexed as the first private information by letting $\tau(x, 1) = z$ if and only if the people from $z[20..29]$ background and $z[0..19]$ zip code area have average balance of term deposit in the interval* $[z[30..59], z[60..89]]$.

Output of the performance testing for $\mathcal{F}$ can be measured by a metric function $\mathcal{T}_{\mathcal{F}}$. In particular, $\mathcal{T}_{\mathcal{F}}$ can be regarded as a mapping from $\Sigma^* \times \Sigma^*$ to N. The metric function $\mathcal{T}_{\mathcal{F}}(x, y)$ could represent the time used by the software package $\mathcal{F}$ when running on database x with input test case y.

The goal of the synthetic data generation is to produce a synthetic data set x' from the production data set x such that metric function outputs $\mathcal{T}_{\mathcal{F}}(x, y)$ and $\mathcal{T}_{\mathcal{F}}(x', y)$ are approximately the same for most input test cases y. At the same time the synthetic data x' should contain no private information. Our above discussion can be formalized as in the following definition.

Definition 1. *Let $\gamma : \Sigma^* \times \Sigma^* \rightarrow N$ be a function denoting the acceptable metric difference for the testing purpose, $\tau : \Sigma^* \times N \rightarrow \Sigma^*$ be a Turing computable privacy function, $\mathcal{F} : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ be a Turing machine denoting the software package, and $\mathcal{T}_{\mathcal{F}} : \Sigma^* \times \Sigma^* \rightarrow N$ be a metric function defined for $\mathcal{F}$. We say that a probabilistic Turing machine $\mathcal{G} : \Sigma^* \rightarrow \Sigma^*$ is a* (γ, τ)-privacy preserving synthetic data generator *for $\mathcal{F}$ if the following conditions are satisfied:*

1. *For all $x \in \Sigma^*$, $|\mathcal{G}(x)| = |x|$.*
2. *(Performance similarity) $|\mathcal{T}_{\mathcal{F}}(x, y) - \mathcal{T}_{\mathcal{F}}(\mathcal{G}(x), y)| \leq \gamma(x, y)$ with overwhelming probability, where the probability is taken over all possible values for x, y, and internal coin tosses of $\mathcal{F}$ and $\mathcal{G}$.*
3. *(Privacy preserving) Let $\delta(\cdot) : N \to R$ be the acceptable level function of privacy leakage which is generally a negligible function. For each database x and each i, we have*

$$\sum_w |\mathrm{Prob}[\tau(x,i) = w|\mathcal{G}(x), \mathrm{priorK}] - \mathrm{Prob}[\tau(x,i) = w|\mathrm{priorK}]| \leq \delta(i),$$

where the sum is over all potential output w of $\tau(x, i)$, and priorK *is the prior knowledge that is known to the software tester before the testing.*

Remarks:

- In definition 1, we assume that all functions and Turing machines are defined for all inputs. In practice, there is no guarantee that a program will halt in finite many steps on all inputs. In particular, when a program contains bugs, then for some inputs it may run forever without a valid output. For these scenarios, we assume that when a Turing machine does not halt in expected time (which should be large enough) on an input, then it will halt and output a default result.
- In the item 1, we require that $|\mathcal{G}(x)| = |x|$. This requirement is only for the convenience of our analysis. In practice, the generated synthetic data set may have different (but approximately same) size than the original data set.
- In the item 3, the prior knowledge could include the schema definition of the database x and other prior knowledge about x.
- From the definition, it is straightforward to see that a privacy preserving synthetic data generator must be one-way. Otherwise one can compute x from $\mathcal{G}(x)$ and then compute the value of $\tau(x, i)$, violating condition 3 of Definition 1.

3.2 Infeasibility

In the following, we first show that, for given conditions, a (γ, τ)-privacy preserving synthetic data generator for $\mathcal{F}$ does not necessarily exist.

Statement 1 *(γ, τ)-privacy preserving synthetic data generator for $\mathcal{F}$ does not necessarily exist if the Turing machine $\mathcal{F}$'s running time depends on some bits of the input and these bits are indeed the private information identified by τ.*

Example 2 *Assume that, according to prior knowledge, the first bit $x[0]$ of the database x is uniformly distributed over Σ, $\mathcal{T}_{\mathcal{F}}$ is the time complexity of $\mathcal{F}$,*

$$\begin{aligned} \mathcal{T}_{\mathcal{F}}(0z, y) &= 4 \cdot \max\{\gamma(0z, y), \gamma(1z, y)\}, \\ \mathcal{T}_{\mathcal{F}}(1z, y) &= 2 \cdot \max\{\gamma(0z, y), \gamma(1z, y)\}, \\ \tau(x, 1) &= x[0], \end{aligned}$$

and $\mathcal{G}$ is a (γ, τ)-privacy preserving synthetic data generator for $\mathcal{F}$. Then in order to satisfy the condition 2 of Definition 1, we must have $\mathcal{G}(x)[0] = x[0]$ for almost all x. Otherwise, for $x = bz$,

$$|\mathcal{T}_{\mathcal{F}}(bz, y) - \mathcal{T}_{\mathcal{F}}(\mathcal{G}(bz), y)| = 2 \cdot \max\{\gamma(0z, y), \gamma(1z, y)\} \geq 2\gamma(bz, y) > \gamma(bz, y)$$

for any $b \in \Sigma$ and $z \in \Sigma^*$. Then for $w = \mathcal{G}(x)[0]$, we have

$$\mathrm{Prob}[\tau(x,1) = w|\mathcal{G}(x), \mathrm{priorK}] - \mathrm{Prob}[\tau(x,1) = w|\mathrm{priorK}] = 1 - \frac{1}{2} = \frac{1}{2}.$$

Thus the privacy leakage is larger than (or equal to) a non negligible value $\frac{1}{2}$ and the condition 3 of Definition 1 is not satisfied. This is a contradiction, which shows that no (γ,τ)-privacy preserving synthetic data generator for $\mathcal{F}$ exists.

Statement 2 *Even if there is no* conflict *between τ and the running time of $\mathcal{F}$, (γ,τ)-privacy preserving synthetic data generator for $\mathcal{F}$ may still not exist if the Turing machine $\mathcal{F}$ has some conflict with the performance requirement $\gamma(x,y)$.*

Example 3 *For any binary string $x \in \Sigma^*$, let int_x denote the positive integer whose binary representation is x. Assume that $\mathcal{T}_{\mathcal{F}}$ is the time complexity of $\mathcal{F}$ with the property that*

$$\mathcal{T}_{\mathcal{F}}(x,y) = 2 \cdot int_x \cdot \max\{\gamma(z,y) : |z| = |x|\},$$

$\tau(x,i)$ is a non-trivial privacy function (that is, a function whose output values depend on input values), and $\mathcal{G}$ is a (γ,τ)-privacy preserving synthetic data generator for $\mathcal{F}$. As we have noted in the previous paragraph, $\mathcal{G}$ should be one-way, thus $\mathcal{G}(x) \neq x$ for almost all x. Then

$$\begin{aligned} |\mathcal{T}_{\mathcal{F}}(x,y) - \mathcal{T}_{\mathcal{F}}(\mathcal{G}(x),y)| &= 2 \cdot |int_x - int_{\mathcal{G}(x)}| \cdot \max\{\gamma(z,y) : |z| = |x|\} \\ &\geq 2 \cdot \max\{\gamma(z,y) : |z| = |x|\} \\ &> \gamma(x,y) \end{aligned}$$

for almost all $x \in \Sigma^*$. Thus the condition 2 of Definition 1 is not satisfied, which is a contradiction. This shows that no (γ,τ)-privacy preserving synthetic data generator for $\mathcal{F}$ exists.

The above two statements essentially show that if the Turing machine $\mathcal{F}$ is a “distinguisher” of some privacy bits or a “distinguisher” of the input strings, then privacy preserving synthetic data generator does not exist. On the other hand, if the Turing machine $\mathcal{F}$ has approximately same metrics on all inputs of the same length, then any random mapping serves as the privacy preserving synthetic data generator for $\mathcal{F}$. That is, if $|\mathcal{T}_{\mathcal{F}}(x_1,y) - \mathcal{T}_{\mathcal{F}}(x_2,y)| \leq \gamma(x_1,y)$ for all x_1 and x_2, then any length-preserving random mapping is a (γ,τ)-privacy preserving synthetic data generator for $\mathcal{F}$. In this paper, we will concentrate on Turing machine $\mathcal{F}$ which do not go to the two extremes that we have just mentioned.

3.3 Problem Formulation

The results in the previous section show that, without the knowledge of the underlying data structures in x and the operation commands in $\mathcal{F}$, it is generally impossible to construct (γ,τ)-privacy preserving synthetic data generator for a generic Turing machine $\mathcal{F}$. In this section, we concentrate on the synthetic database generation and on database application software packages and assume the workload is given. Generally, we have two problems depending on different scenarios:

Problem 1. Given a private property τ, and $\delta(i)$, find a synthetic database x' to minimize

$$\sum_y |\mathcal{T}_\mathcal{F}(x,y) - \mathcal{T}_\mathcal{F}(x',y)| \cdot \mathrm{Prob}[y]$$

subject to $\sum_w \left| \text{Prob}[\tau(x,i) = w|x', \text{priorK}] - \text{Prob}[\tau(x,i) = w|\text{priorK}] \right| \leq \delta(i)$.

Problem 2. Given acceptable performance metric $\gamma(x,y)$, and private property τ, find synthetic database x' to minimize

$$\sum_i \sum_w \left| \text{Prob}[\tau(x,i) = w|x', \text{priorK}] - \text{Prob}[\tau(x,i) = w|\text{priorK}] \right|$$

subject to $|\mathcal{T}_\mathcal{F}(x,y) - \mathcal{T}_\mathcal{F}(x',y)| \leq \gamma(x,y)$ for almost all y.

The two problems are related to each other. In the remaining part of this paper, we will focus on Problem 1. One should note that the subject condition in Problem 1 is generally not a linear inequality and the distribution of y is not uniform (i.e., Prob$[y]$ is not a constant). Thus the Problem 1 is not a linear programming problem and heuristic methods are needed to solve this problem.

4 Constructing (γ, τ)-Privacy Preserving Synthetic Data Generator

	Categorical				Numerical
	Zip	Background	Age	Gender	Balance
1	10001	Asia	20	M	10k
2	10001	Asia	30	F	15k
3	10002	British	20	M	50k
.	.	.	.	.	.
m	10001	British	25	M	80k

Table 1. An Example of ACCOUNT Database

In this paper, we assume that file organizations, sorted fields, and index structures of the production database x are not private information and the synthetic data generator will use these information to build the synthetic database $\mathcal{G}(x)$ in the same way that x has been built. We view the database as a multi-dimensional table with categorical attributes and numerical attributes and model it by using the *general location model*. Table 1 shows an example ACCOUNT database with m tuples. Assume the workload is given as follows and the private property is as Example 1:

Q1: INSERT INTO ACCOUNT VALUES ()
Q2: SELECT * FROM ACCOUNT WHERE Zip = z AND Background = b

It is clear that the distribution of underlying data affects the execution time of workload. In general data sets, some columns' distribution may be dependent on those of other columns, hence we would like to derive an approximate joint distribution of all columns which is used to generate synthetic data for performance testing. For example, the approximate joint distribution on $Zip, Background, Balance$ would satisfy the performance requirements of workload (Q1 and Q2). Our intuition is that, for database applications, if two databases x and x' are approximately the same from a statistical viewpoint, then the performance of the application on the two databases should also be approximately the same.

However, the joint distribution may contain sensitive information about private properties. In the next subsection, we will present an heuristic algorithm to derive approximate joint distribution based on the general location model and to check whether it contains confidential information.

4.1 A Greedy Algorithm

We view the database as a multi-dimensional table with categorical attributes and numerical attributes and model it by using the *general location model*. We also assume the general location model itself is not confidential information and only the parameters of the general location model are confidential. We can see from Equation 1 that the maximum likelihood estimates of $\theta = (\pi, \mu, \Sigma)$ can be fully derived from statistics (e.g., the frequencies of tuples which satisfy some conditions of categorical attributes, the mean and variance values of tuples which belong to same cell). Those statistics are not completely contained in database catalog[1]. However, it is straightforward to derive those statistics by imposing various queries if we are allowed to access the original data.

It is worth pointing out that we do not need to build the general location model at the finest level as those statistics are with very high complexity which is exponential to the size of contingency table and many statistics do not have effects on a given workload performance. Hence an approximate and condensed general location model on the subsets of attributes is sufficient for performance testing. Here the condensed model is derived from a condensed contingency table which is formed by a subset of categorical attributes (even with coarser domain values) which can be identified by SQLs in workload.

For the reason of convenience, we use $\mathcal{DB}^0_{\mathcal{F},x}$ to denote all the information (e.g., the distribution, rules and the priori knowledge priorK) needed to build database x to satisfy the condition 2 of Definition 1. We call the distribution $\mathcal{DB}^0_{\mathcal{F},x}$ the *performance characteristic* of a database x for the application $\mathcal{F}$. The heuristic method for solving the problem 1 could be given as follows.

- Step 1. Extract $\mathcal{DB}^0_{\mathcal{F},x}$ from the real database x based on workload and construct the estimated performance error function $\gamma(x, y)$.
- Step 2. Specify a list of confidential information $\tau(x, i)$ and an acceptable privacy leakage level $\delta(i)$ for each privacy indexed by i.

[1] In current commercial database datalog, only the simple statistics (e.g., mean, max, min etc.) of each single column are collected.

– Step 3. Check whether the performance characteristic $\mathcal{DB}^0_{\mathcal{F},x}$ leaks any privacy information defined in $\tau(x,i)$ in a non-acceptable level (that is, larger than $\delta(i)$) according to Definition 1 (see section 4.2).
– Step 4. If the privacy leakage is not acceptable, repeat the following until $\mathcal{DB}'_{\mathcal{F},x}$ leaks privacy information about $\tau(x,i)$ in the acceptable level defined by δ.
  - Step 4.1. The analyzer constructs a new characteristic $\mathcal{DB}'_{\mathcal{F},x}$ by perturbing θ as $\theta' = (\pi + \delta_\pi, \mu + \delta_\mu, \Sigma + \delta_\Sigma)$.
  - Step 4.2. The performance analyzer constructs a new estimated performance error function $\gamma'(x,y)$ according to the new distribution $\mathcal{DB}'_{\mathcal{F},x}$. This is generally larger than the previous $\gamma(x,y)$.
– Step 5. A synthetic data generator generates a synthetic data set using the distribution $\mathcal{DB}'_{\mathcal{F},x}$.

During Steps 1 and 4.2, we apply two test statistics discussed in section 2.2 to test whether the generated data x' using $\mathcal{DB}'_{\mathcal{F},x}$ has the same distribution with original data (assuming a sample is given). From the point of statistical view, we can never prove that two data sets come from a single distribution. On the contrary, we aim to disprove, to a certain required level of significance, the null hypothesis that two data sets are drawn from the same population distribution function. As there is no test statistics which can be directly used for the general location model, we decompose the general location model to two parts and use χ^2 test for multinomial distribution and $Kolmogorov - Smirnov$ test for multivariate normal distributions of each cell.

4.2 Privacy Analyzer

The results from the previous section show that in order to construct a privacy preserving synthetic data generator for the application $\mathcal{F}$, one first needs to extract the performance characteristic $\mathcal{DB}^0_{\mathcal{F},x}$ from the input database x. Then one can draw a random database x' according to the distribution $\mathcal{DB}^0_{\mathcal{F},x}$ and build the same file structures, sorted fields, and indexes for x'. If x' leaks no private information about x, then x' could serve for performance testing. However, x' constructed in such a way may leak some confidential information. In this section, we will address techniques to decide whether there are private information leakage in $\mathcal{DB}^0_{\mathcal{F},x}$ and, if the answer is yes, how to construct a new distribution $\mathcal{DB}'_{\mathcal{F},x}$ that contains no confidential information. Without loss of generality, we assume that the function τ is defined only for the first index. We write this confidential information as $\tau(x) = \tau(x,1)$ for short. In the following, we give some examples to illustrate how to decide whether there is information leakage about $\tau(x)$ in a distribution $\mathcal{DB}^0_{\mathcal{F},x}$.

Example 4 *Let l and m be two integers such that $n = l \times m$. For a database x, let $X[i] = x[il] \ldots x[il + l - 1]$ for $0 \leq i \leq m - 1$. In practice, $X[0], \cdots, X[m-1]$ may correspond only to one column of a table. For the reason of simplicity, in this example, we assume that the database has one table which has one column $X[0], \cdots, X[m-1]$. Assume that the following conditions hold:*

1. *According to* priorK*, $X[0], \ldots, X[m-1]$ follow a normal distribution $N(\mu, \sigma)$ and μ is uniformly distributed over $100 \leq \mu \leq 500$;*

2. *$\tau(x) = \frac{\sum_{i=0}^{m-1} X[i]}{m}$; and*
3. *according to $\mathcal{DB}^0_{\mathcal{F},x}$, $X[0], \ldots, X[m-1]$ follow the normal distribution $N(\mu_0, \sigma_0)$.*

Then

$$\begin{array}{l}\left|\text{Prob}[\tau(x) = \mu_0 | \mathcal{DB}^0_{\mathcal{F},x}] - \text{Prob}[\tau(x) = \mu_0 | \text{priorK}]\right| \\ = \left|(1-\varepsilon) - \frac{1}{400}\right| \ = \ \frac{399}{400} - \varepsilon\end{array} \qquad (2)$$

for some small ε. Obviously $\mathcal{DB}^0_{\mathcal{F},x}$ leaks significant information about $\tau(x)$. Thus we need to modify the performance characteristic $\mathcal{DB}^0_{\mathcal{F},x}$. One potential solution is to pick a random value v, and use a new distribution $N(\mu_0 + v, \sigma_0)$. This may have further impact on the application performance and we need to re-compute the new performance error function $\gamma(x, y)$ for this new distribution. If we choose $|v| \leq t$, then for any $\mu_0 + v - t \leq w \leq \mu_0 + v + t$,

$$\begin{array}{l}\left|\text{Prob}[\tau(x) = w | \mathcal{DB}'_{\mathcal{F},x}] - \text{Prob}[\tau(x) = w | \text{priorK}]\right| \\ = \left|\left(\frac{1}{2t} - \varepsilon\right) - \frac{1}{400}\right| = \frac{1}{2t} - \frac{1}{400} - \varepsilon\end{array} \qquad (3)$$

for some small ε. When t is large enough, the value in equation (3) is small enough so that the sum on all w is less than the pre-specified value δ [2]*.*

The example in the previous paragraph shows a heuristic method to modify the performance characteristic to meet the privacy requirements. In order to construct a confidential-information-free distribution $\mathcal{DB}'_{\mathcal{F},x}$ from $\mathcal{DB}^0_{\mathcal{F},x}$, we first need to identify those parts of $X[i]$ on which the value of $\tau(x)$ depends. Then we need to perturb the elements in $\mathcal{DB}^0_{\mathcal{F},x}$ that have impacts on the values of those $X[i]$. After these perturbation, we get a new distribution $\mathcal{DB}'_{\mathcal{F},x}$ and we can check whether the information leakage is acceptable.

We close this section by giving a formula for evaluating $\text{Prob}[\tau(x) = w | \mathcal{DB}'_{\mathcal{F},x}]$ for the new distribution $\mathcal{DB}'_{\mathcal{F},x}$, where w is any given string. Note that this computation is necessary for checking whether the information leakage is acceptable.

Let DBco denote the conversion method, that is used to convert $\mathcal{DB}^0_{\mathcal{F},x}$ to $\mathcal{DB}'_{\mathcal{F},x}$, together with the prior knowledge. Then given $\mathcal{DB}'_{\mathcal{F},x}$, one can evaluate the probabilities of possible values $\mathcal{DB}$ of $\mathcal{DB}^0_{\mathcal{F},x}$, given $\mathcal{DB}'_{\mathcal{F},x}$. Using Bayes formula, one can compute the posterior probabilities:

$$\text{Prob}[\mathcal{DB} | \mathcal{DB}'_{\mathcal{F},x}, \texttt{DBco}] = \frac{\text{Prob}[\mathcal{DB}] \cdot \text{Prob}[\mathcal{DB}'_{\mathcal{F},x} | \mathcal{DB}, \ \texttt{DBco}]}{\text{Prob}[\mathcal{DB}'_{\mathcal{F},x} | \texttt{DBco}]}$$

Note that the probabilities $\text{Prob}[\mathcal{DB}]$ of possible values $\mathcal{DB}$ of $\mathcal{DB}^0_{\mathcal{F},x}$ can be easily computed when the distribution of the general location model's parameters, $\theta = (\pi, \mu, \Sigma)$, are given (e.g., we can assume θ has a uniform distribution over a specified range $[\theta_l, \theta_u]$). The probability $\text{Prob}[\mathcal{DB}'_{\mathcal{F},x} | \texttt{DBco}]$ can be computed using the formula:

$$\text{Prob}[\mathcal{DB}'_{\mathcal{F},x} | \texttt{DBco}] = \sum_{\mathcal{DB}} \text{Prob}[\mathcal{DB}] \cdot \text{Prob}[\mathcal{DB}'_{\mathcal{F},x} | \mathcal{DB}, \texttt{DBco}].$$

[2] Note that the computation in the equation (3) is only for illustration purpose and is not exact. At the end of this section, we will give exact evaluation formulae for the computation of posterior probability.

Thus the posterior probability $\text{Prob}[\tau(x) = w|\mathcal{DB}'_{\mathcal{F},x}]$ could be computed as:

$$\text{Prob}[\tau(x) = w|\mathcal{DB}'_{\mathcal{F},x}] = \sum_{x',\, \mathcal{DB}} \left(\text{Prob}[\mathcal{DB}|\mathcal{DB}'_{\mathcal{F},x}] \cdot \text{Prob}[x'|\mathcal{DB}] \cdot \text{Prob}[\tau(x) = w|x']\right),$$

where we omitted the prior knowledge `DBco` from the formula.

As there may have exponentially many w's, we need to use discretization method and divide possible w's into discrete intervals and compute their probabilities.

5 Other Issues Related to Our Work

5.1 Privacy Breach Issues

In section 1, we briefly described researches on privacy leakage in related areas such as statistical database and privacy preserving data mining. We observed that existing definitions for privacy leakage are not sufficient for our research. Our definition of privacy leakage is closely related to the statistical indistinguishability concept in cryptographic research [8]. It is straightforward to check that our definition covers both straight (upward) privacy breach and inverse (downward) privacy breach defined in [6]. However, our definition is more general as it also covers the collective privacy breaches: for each individual event, the observable probability difference is small, but, collectively, the sum of these observable probability differences are large enough. This kind of collective privacy breaches could be important for many applications.

5.2 Statistical Database and Privacy Preserving Synthetic Data Generation

The problem of privacy issues in statistical databases have been extensively studied (see, e.g., [1]). Recently, Dinur and Nissim [5] have tried to give a formal definition of privacy issues in statistical databases and they have obtained several impossibility results and several feasibility results. Their main feasibility result is based on the assumption that the adversary has no prior knowledge about the database. That is, they assume that the database is a uniform distribution over Σ^n. This is generally an impractical assumption since in practice the adversary have much prior knowledge about the database. However, several techniques in the paper [5] is helpful in understanding the privacy preserving synthetic data generation issues and can be used for privacy preserving data generation.

For example, [5, Theorem 5] provides a feasibility results for privacy preserving statistical databases if one assumes that the statistical database is a uniform distribution on binary strings. Specifically, [5, Theorem 5] says that for a $t(n) > polylog(n)$ time bounded adversary, one can achieve statistical database privacy by perturbing the query output by an order of $O(\sqrt{t(n)})$. In the following, we apply the techniques in the proof of [5, Theorem 5] to privacy preserving synthetic data generation. Assume that the values of an attribute in the live production database x is distributed uniformly and the application software performance depends on the values of $\sum_{i\in I} x[i]$ for different kinds of $I \subseteq \{0, \ldots, n-1\}$ and the private information is the re-identification of any specific entry in the database. Then one can construct a performance characteristic $\mathcal{DB}^0_{\mathcal{F},x}$ as follows. Fix a value $t(n)$ (for example, let $t(n) = n$), let $R = t(n) \cdot \log^{\mu} n$ for some

$\mu > 0$, and let $I_0, \ldots, I_{t(n)}$ be a sequence of random chosen subsets of $\{0, \ldots, n-1\}$. For each $j \leq t(n)$, let a_{I_j} be defined as follows:

1. If $|I| < \sqrt{R} \cdot \log^2 n$ then $a_{I_j} = 0$.
2. Otherwise, $a_{I_j} = \sum_{i \in I_j} x[i] + \mathcal{E}$, where $\mathcal{E} = \sum_{i=0}^{R-1} w[i] - R/2$ for a random chosen $w \in \Sigma^R$.

Let $\mathcal{DB}^0_{\mathcal{F},x}$ be constructed in such a way that any database x' drawn according to $\mathcal{DB}^0_{\mathcal{F},x}$ satisfies

$$\left| \sum_{i \in I_j} x'[i] - a_{I_j} \right| \leq \sqrt{t(n)}.$$

Then a similar proof as in the proof of [5, Theorem 5] shows that the performance characteristic $\mathcal{DB}^0_{\mathcal{F},x}$ leaks no private information about x.

5.3 Generating Synthetic Data via Statistical Queries

In certain cases, the performance characteristic $\mathcal{DB}^0_{\mathcal{F},x}$ of the original database could only be extracted from the real database x via statistical queries to x. If the answers to the statistical queries is not perturbed and no modification to $\mathcal{DB}^0_{\mathcal{F},x}$ has been done, then the existence of a database satisfying the distribution is guaranteed to exist since x itself satisfy the distribution. However, if the answers to the queries has been perturbed or, in order to remove the confidential information contained in the performance characteristic, $\mathcal{DB}^0_{\mathcal{F},x}$ has been modified, there are several concerns for this procedure.

The first problem is related to the existence of a sample database for the revised distribution. The new distribution may contain contradiction and the sample space could be empty. If this case happens, one may try to remove the contradiction from the distribution. This modification could have further impact on the application performance.

The second problem is related to the efficiency of synthetic data generation. It may take exponential time to generate a synthetic data set according to a given distribution. For example, if the distribution requires that for each index subset $I \subseteq \{0, \ldots, n-1\}$, the database x' satisfies some condition on $\sum_{i \in I} x[i]$, then one may have to try exponentially many instances to output x'. For a specific class of distributions, one may design generic optimized algorithms to construct the synthetic data efficiently.

6 Conclusion

We studied the problem of generating synthetic data for database application performance testing while preserving privacy. We presented a model for quantifying the privacy leakage and application performance metric difference by using the general location model. Our infeasibility results show the strict privacy preserving synthetic data generator does not necessarily exist when the application workload is a "distinguisher" of some privacy properties or a "distinguisher" of the input strings. A heuristic method was given for the relaxed problem, i.e., to construct the generator which satisfies performance requirements as many as possible while preserving all the privacy properties.

The analysis in this paper shows that it is a challenging problem to design efficient privacy preserving synthetic data generators. One open problem is to study complexity issues. Another topic for future work is to extend our approach for multiple tables and integrate with a-priori rules and constraints in databases.

References

1. N. R. Adam and J. C. Wortman. Security-control methods for statistical databases. *ACM Computing Surveys*, 21(4):515–556, Dec 1989.
2. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 439–450. Dallas, Texas, May 2000.
3. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. of FOCS*, 1995.
4. R.B. Dagostino and M.A. Stephens. *Goodness-of-fit Techniques*. New York Dekker, 1986.
5. I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. 22nd Symposium on Principles of Database Systems*, pages 202–210, 2003.
6. A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. 22nd Symp. on Principles of Database Systems*, pages 211–222, 2003.
7. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *JCSS*, 60(3):592–629, 2000.
8. O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.
9. Y.Lindell and B.Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.
10. B. Malin, L. Sweeney, and E. Newton. Trail re-identification: learning who you are from where you have been. In *Proc. of the LIDAP-WP12*. Carnegie Mellon University, 2003.
11. Niagara. http://www.cs.wisc.edu/niagara/datagendownload.html.
12. Quest. http://www.quest.com/datafactory.
13. J.L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman Hall, 1997.
14. C. J. Skinner. On identification disclosure and prediction disclosure for microdata. *Statistical Neerlandical*, 44:21–32, 1992.
15. X. Wu, Y. Wang, and Y. Zheng. Privacy preserving database application testing. In *Proc. ACM Workshop on Privacy in Electronic Society*, pages 118–128, 2003.