Privacy Aware Data Generation for Testing Database Applications

Xintao Wu, Chintan Sanghvi, Yongge Wang, Yuliang Zheng
University of North Carolina at Charlotte
{xwu, cpsanghv, yonwang, yzheng}@uncc.edu

Abstract

Testing of database applications is of great importance. A significant issue in database application testing consists in the availability of representative data. In this paper we investigate the problem of generating a synthetic database based on a-priori knowledge about a production database. Our approach is to fit general location model using various characteristics (e.g., constraints, statistics, rules) extracted from the production database and then generate the synthetic data using model learnt. The generated data is valid and similar to real data in terms of statistical distribution, hence it can be used for functional and performance testing. As characteristics extracted may contain information which may be used by attacker to derive some confidential information about individuals, we present our disclosure analysis method which applies cell suppression technique for identity disclosure analysis and perturbation for value disclosure.

1. Introduction

Database application software testing is by far the most popular activity currently used by developers or vendors to ensure high software quality. A significant issue in database testing consists in the availability of representative data.

Recently, the authors in [17, 18] have proposed a general framework for privacy preserving database application testing and investigated the method of generating synthetic data sets based on a-priori knowledge (e.g., constraints, statistics, rules etc.) about the current production data sets. In testing the functions of database applications, the generated data need to satisfy all the constraints (e.g., no-Null, uniqueness, referential integrity constraints, domain constraints, and semantic constraints) and business rules underlying the live data. In testing the performance of database applications, it will be imperative that the data is resembling real data in terms of statistical distribution since the statistical nature of the data determines query performance.

In addition to valid (in terms of constraints and rules) and resembling real data (in terms of statistical distribution), the generated data need to preserve privacy. To be privacy preserving, the generated data should not disclose any confidential information that the database owner would not want to reveal. There are several kinds of confidential information that a database owner would like to protect. An incomplete list could include: the existence of some fields in a table, some statistical data about the live database, some deterministic or non-deterministic business rules or constraints of the live database, and users' records in the tables. Since the synthetic database is generated from a-priori knowledge about the live database, how to preclude confidential information from a-priori knowledge becomes an important issue.

In this paper, we examine how to generate a synthetic database, which is similar to the production database and does not disclose confidential information, for privacy preserving database application testing. This issue is related to, but not identical to, the widely recognized problem of privacy preserving data mining (e.g., [1]). In the situation we present, we extract characteristics from production databases and use characteristics to generate a synthetic database. Our disclosure analysis is conducted at model level instead of tuple level.

Our contributions are as follows:

- First, the context in this paper is to use the general location model to model databases and investigate how to use model learned to generate synthetic database. As the model learned is the only means to generate data for release, all confidential information which attackers can derive is guaranteed to be contained in those parameters. Furthermore, as the search space of parameters is much smaller than the space of perturbed data, this approach is more effective and efficient.

- Second, we evaluate the effect of data distribution on workload performance using TPC benchmarks and show the data generated need to be statistically similar to original data in order to fulfill requirements of application testing.

- Third, we examine how to resolve the potential disclosure (both identity disclosure and value disclosure) of confidential information entailed in the general location model. We extend the concept of a uni-variate confidence interval to a multi-variate confidence region to measure privacy and confidentiality for multiple confidential attributes simultaneously.

The remainder of the paper is structured as follows. In section 2 we review the related work. We describe our privacy aware data generation system in section 3 and briefly revisit how to fit the general location model in section 4. In section 5, we present in detail how to screen out confidential information from characteristics. In section 6, we first evaluate the effect of data distribution on workload performance using TPC Benchmarks, and show the performance of our disclosure analysis. In section 7 we draw conclusions and describe directions for future work.

2. Related Work

Testing of database applications is of great importance since undetected faults in these applications may result in incorrect modification or accidental removal of crucial data. Although various studies have been conducted to investigate testing techniques for database design, relatively few efforts have been made to explicitly address the testing of database applications. The problem of database application testing can be categorized into three parts: database generation, input test cases preparation and test outcomes verification[2]. In this paper, we focus on database generation.

There have been some prior investigations into data generation. For example, Transaction Processing Performance Council has released a dozen of TPC Benchmarks and many researchers have evaluated those Benchmarks (e.g., [8, 10, 12]). There are also some other data generation tools (e.g., [13, 11]) available. However, both TPC Benchmarks and other data generation tools are built for assessing the performance of database management systems, rather than for testing complex real world database applications. They lack the required flexibility to produce more realistic data needed for application testing, i.e., the generated data also need to satisfy all the constraints and business rules underlying the live data.

To generate realistic data for database applications, the authors in [3, 4] investigate how to populate the database with meaningful data that satisfy database constraints. They present a tool which inputs a database schema definition, along with some additional information from the user, and outputs a valid database state. The tool can handle not-NULL, uniqueness, referential integrity constraints, and some domain constraints and semantic constraints. Most constraints are included in data schemas which are expressed by SQL data definition language (DDL). The tool parses the schema definition for the database underlying the application to be tested using PostgreSQL (http://www.postgresql.org), then collects relevant information about tables, attributes, and constraints from the parse tree. The generation technique was motivated by the category-partition testing technique.

The inherent challenge of generating data for database applications is the tradeoff between similarity and privacy preservation. If the data is too synthetic (e.g., completely uniform distributions), it runs the risk of being rejected for not capturing he interesting patterns of a real data set. Conversely, if it employs data from the real world directly, it risks the violation of privacy issues. In terms of performance testing, using a large amount of *resembling* data is necessary to guarantee its satisfied performance when software is deployed. The generated data need to resemble real data in terms of statistical distribution in order to fulfill requirements of applications testing. The authors in [14] points out the importance of providing meaningful, representative data with realistic skew, sparcity and data distributions for benchmarking database system performance as the current development of DBGEN for TPC-D used a relatively simple, third-normal-form schema and modest requirements in terms of data complexity and scaling (e.g., all data was uniformly distributed). Zheng et al. in [19] show that artificial data sets have very different characteristics from the real-world data sets and hence there is a great need to use real-world data sets as benchmarks for association rule mining. As many databases maintain data on sensitive or confidential information such as income and assets for real customers. It is imperative to guarantee the data generated can not disclose any private or confidential information.

3. System Overview

Figure 1 shows the architecture of synthetic database generation system. The system is intended to generate a synthetic database using various characteristics extracted from the production database for testing database applications, instead of generating benchmark for use in the performance evaluation of DBMS. We assume databases are based on the relational model in our paper. A relational database is a set of relation schemas together with a set of integrity constraints which restrict the possible values of the database states.

We would like automate the data generation process as much as possible. We use the same approach as in [3] to extract various constraint information from schemes which are defined by DDL. It is desirable that the generated data in synthetic databases also satisfy the constraints. A major advantage is that our system can extract more complex

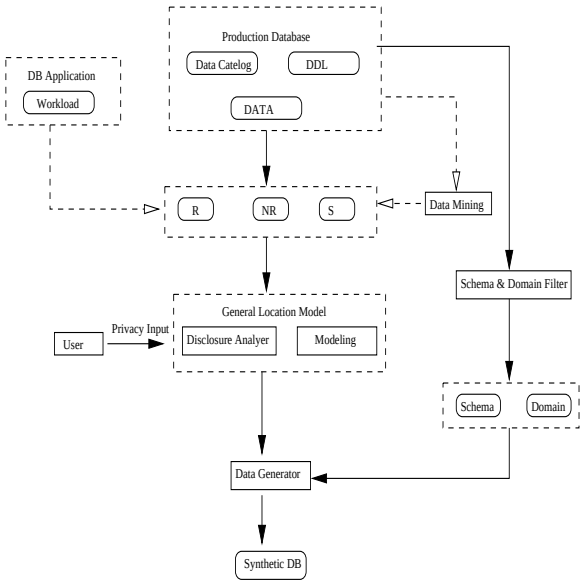


Figure 1. Architecture of data generation system

characteristics (e.g., statistics and rules) in addition to constraints from data catalog and data when the underlying production database is available. As we discussed in introduction, even if one synthetic database satisfies all constraints, it does not mean it can fulfill users' testing requirement as it may have different data distribution than the production database. Hence our approach extracts characteristics (e.g., statistics, rules) from the production database, fits general location model using characteristics extracted, and generates synthetic database using model learnt. As shown in Figure 1, the characteristics of production databases can be extracted from three parts: DDL, Data Dictionary, and Data. In order to ensure that the data is close looking or statistically similar to real data, or at least from the point of view of application testing, we need to have the statistical descriptions, $\mathcal{S}$, and non-deterministic rules, $\mathcal{NR}$, of real data in production databases. These two sets describe the statistical distributions or patterns of underlying data and may affect the size of relations derived as a result of the evaluations of queries the application will need to execute. Our intuition is that, for database applications, if two databases are approximately the same from a statistical viewpoint, then the performance of the application on the two databases should also be approximately the same [1]. Furthermore, our system includes one disclosure analysis component which helps users remove those characteristics which may be used by attackers to derive confidential information in the production database. As all information used to generate synthetic data in our system are contained in characteristics extracted, our disclosure analysis component can analyze and preclude the potential disclosure of confidential information at the characteristics (i.e., statistics and rules) level instead of data level.

[1] Here we assume that file organizations, sorted fields, and index structures of the production database x are not private information and the synthetic data generator use these information to build the synthetic database in the same way that production database has been built.

4. Database Modeling via the General Location Model

Our approach is to derive an approximate statistical model from the characteristics (e.g., constraints, statistics, rules, and data summary) of the real databases and generate a synthetic data set using model learned.

In our system, we use the general location model to describe the general data set which may contain a number of categorical attributes (e.g., Zip, Race, Age, Gender in Table 1) and a number of numerical attributes (e.g., Balance, Income, InterestPaid in Table 1). Here we assume SSN and Name are confidential information and should be marked. The general location model is defined in terms of the marginal distribution of categorical attributes and the conditional distribution of numerical attributes given each cell determined by categorical attribute values. The former is described by a multinomial distribution on the cell count with parameter π when we summarize the categorical part as a multi-dimensional contingency table.

The general location model assumes the numerical attributes (i.e., Balance, Income, InterestPaid) of tuples in each cell follow a multivariate normal distribution with its parameters μ, Σ, where μ is a vector of means and Σ is a covariance matrix. For example, the Bal-

Table 1. An Example of Mortgage Dataset

			Categorical				Numerical		
	SSN	Name	Zip	Race	Age	Gender	Balance	Income	InterestPaid
1			28223	Asian	20	Male	10k	85k	2k
2			28223	Asian	30	Female	15k	70k	18k
3			28262	Black	20	Male	50k	120k	35k
.			.	.	.	.	.	.	.
n			28223	Black	25	Male	80k	110k	15k

ance, Income and InterestPaid of customers in the cell $\{28223, Asian, 20, Male\}$ follow a 3-variate normal distribution with parameter μ_1, Σ_1, where μ_1 is a 3-vector means and Σ_1 is a 3×3 covariance matrix while those in the cell $\{28223, Asian, 20, Female\}$ may follow a 3-variate normal distribution with different means and covariance matrix. The cell $\{28223, Asian, 20, ALL\}$, which denotes a cell in 4-dimensional cube, contains all the customers with Zip = 28223, Race = Asian, and Age = 20. For formal description of the general location model, refer [18] which also presents how to extract various characteristics from production databases and how to fit the general location model using characteristics extracted. In this paper, we focus on how to analyze the model and check whether it contains confidential information.

It is straightforward to see we can easily generate a dataset when the parameters of general location model are given. Generally, it involves two steps. First, we estimate the number of tuples in each cell d and generate x_d tuples, where x_d is the number of entries in cell d. All x_d tuples from this cell have the same categorical attribute values inherited from the cell location of contingency table. Second, we estimate the mean and covariance matrix for those tuples in this cell and generate numerical attribute values based on the multi-variate normal model.

5. Disclosure Analysis

Disclosures which can occur as a result of inferences by attackers include two classes: identity disclosure and value disclosure. Identity disclosure relates to the disclosure of the identity of an individual in the database while value disclosure relates to the disclosure of the value of a certain confidential attribute of that individual.

Given characteristics $\mathcal{DB} = \{\mathcal{S} \cup \mathcal{R} \cup \mathcal{NR}\}$ and a set of confidential information $\bar{\mathcal{DB}} = \{\bar{S} \cup \bar{R} \cup \bar{NR}\}$, our initial problem is to find a $\hat{\mathcal{DB}}$ such that 1) $\hat{\mathcal{DB}} \subseteq \mathcal{DB}$ and 2) no confidential information in $\bar{\mathcal{DB}}$ can be entailed from $\hat{\mathcal{DB}}$ within a given bound. We can see this initial problem is very complex as rules and statistics have different formats. In our system, we use the general location model, which is built from rules and statistics, to generate the final data. So all the information are contained in the parameters of the general location model, i.e., $\theta = (\pi, \mu, \Sigma)$ [2].

From section 4, we know 1) π is only used for multinomial distribution and can be estimated using $\hat{\pi}_d = \frac{x_d}{n}$, where x_d is the number of entries in cell d; and 2) μ, Σ is only used for multi-variate normal distribution of numerical attributes for those entries in a given cell d. As π_d is only related to categorical attributes and μ, Σ are only related to numerical attributes, we can analyze them separately. In the remainder of this subsection, we discuss in detail how to check whether π_d incurs identity disclosure and whether μ, Σ incurs value disclosure.

5.1. Identity Disclosure

During the data generation process, data have been de-identified by suppressing SSN and Name so not to disclose the identities of the individuals to whom the data refer. However, values of other released attributes, such as Zip, Race, and Age can also appear in some external table (e.g., voter list as shown in Table 3) jointly with the individuals' identities, and can therefore allow them to be tracked. For example, the individual Alice can be identified and linked to Mortgage table when there is only one Asian female with age 30 and living in the 28223 area, thus revealing that her confidential financial information. While the above example demonstrated an exact match, in some cases, linking can allow the identification of a restricted set of individuals to whom the released information could refer.

In order to preserve the confidentiality of individuals, we typically will not release any rule which involves few records. However, attackers may still be able to derive or estimate the values of some confidential cells by analyzing some cells from the released characteristics. In our scenario, confidential information may even exist at aggregate level. For example, in the table which records the number of patients visiting physicians to receive treatments, the in-

[2] Here we assume the general location model itself is not confidential. In other words, attacker may know our synthetic data is generated by using general location model.

Table 2. Reidentifying anonymous data by linking to extenral data

Name	Address	City	ZIP	DOB	Sex	Party	...
.	.	.	.	.	.	.	.
Alice	9201 University City Blvd	Charlotte	28223	03/18/74	Female	democrat	.
.	.	.	.	.	.	.	.

formation on Patient-Doctor and Doctor-Treatment are not sensitive and are publicly accessible. However, the Patient-Treatment information is sensitive, and so, confidential. This problem is referred as determining upper and lower bounds on the cells of the cross-classification given a set of margins [6, 5]. Upper and lower bounds induced by some fixed set of marginal on the cell entries of a contingency table are of great importance in measuring the disclosure risk associated with the release of these marginal totals. If the induced upper and lower bounds are too tight or too close to the actual sensitive value in a confidential cell entry, the information associated with that cell may be disclosed.

In our system, from characteristics $\mathcal{DB} = \{\mathcal{S} \cup \mathcal{R} \cup \mathcal{NR}\}$, we extract a set of cells, $\mathcal{C}^0$, and the number of entries in each cell $c \in \mathcal{C}^0$. From a list of private rules and statistics, $\bar{\mathcal{DB}} = \{\bar{S} \cup \bar{R} \cup \bar{NR}\}$, we similarly extract a list of confidential cells, $\mathcal{C}^1$. For each confidential cell $c \in \mathcal{C}^1$, a confidential range $[x_c^l, x_c^u]$ which contains the true value of the number of entries, x_c, is derived. $[x_c^l, x_c^u]$ here denotes the confidential range which database owner does not want attackers to predict. It is clear that predicting confidential value within a smaller confidential range constitutes compromise. Now our identity disclosure problem is to find a set of cells, $\mathcal{C}^2$, which can be released for data generation, such that 1) $\mathcal{C}^2 \subseteq \mathcal{C}^1$ and 2) no confidential information x_c ($c \in \mathcal{C}^1$) can be predicted in range $[x_c^l, x_c^u]$ from the information contained in $\mathcal{C}^2$. As this problem is NP-hard, in our system we apply similar heuristics as presented in [7] to remove confidential information contained in $\mathcal{C}^1$ one by one. Basically, it identifies those cells contained in $\mathcal{C}^0$ which need to be suppressed in order to hide the specific confidential information in $\mathcal{C}^1$. We present the details in Appendix A.

5.2. Value Disclosure

Value disclosure represents the situation where attackers are able to estimate or infer the value of a certain confidential numerical attribute of an entity or a group of entities with a level of accuracy than a pre-specified level. Here an entity or a group of entities can be characterized by cell they locate in. Attackers may use various techniques to estimate and predict the confidential values of individual customers. The accuracy with which attackers are able to predict the confidential attribute determines whether disclosure occurs. The greater the accuracy, the closer the estimates are to the true value, and the higher the chance of disclosure. In general, database owner may specify some accuracy levels for some confidential numerical attributes. If attackers are able to estimate a given confidential attribute with a level of accuracy that is greater than that pre-specified, partial disclosure is said to occur.

In our scenario, all numerical attribute values are generated from multi-variate normal distributions. As we discussed before, multivariate normal distribution itself is not considered confidential information, only the parameters μ, Σ which are used for data generation may contain confidential information. We expect database owner specifies a confidential range $[z^l, z^u]$ (z is a confidential numerical attribute) for an entity or a group of entities. Our problem is to make sure that a given set of μ, Σ, which are used for data generation, can not be used by attackers to derive confidential values for an entity or a group of entities in a small confidential bound.

$$P(\beta_1 \le z \le \beta_2) = \Phi(\frac{\beta_2 - \mu}{\sigma}) - \Phi(\frac{\beta_1 - \mu}{\sigma})$$
$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{(2\pi)}} e^{-\frac{v^2}{2}} dv \qquad (1)$$

Equation 1 shows the probability of variable z, which follows a 1-dimensioanl normal distribution $N(\mu, \sigma^2)$, resides in range $[\beta_1, \beta_2]$. If the confidence interval $[\hat{z}_l, \hat{z}_u]$ derived by snoopers are close to the confidential range $[z^l, z^u]$ specified by database owner, we say value disclosure occurs.

$$d(z \mid \hat{z}) = \frac{[z^l, z^u] \cap [\hat{z}^l, \hat{z}^u]}{[z^l, z^u] \cup [\hat{z}^l, \hat{z}^u]} \qquad (2)$$

Equation 2 defines the measure of disclosure for one confidential attribute. Here compromise is said to occur if $d(z \mid \hat{z})$ is greater than τ, specified by database owner. The greater the $d(z \mid \hat{z})$, the closer the estimates are to the true distribution, and the higher the chance of disclosure. In this case, we need to replace the parameters $[\mu, \sigma]$ with a modified $[\mu', \sigma']$ using perturbation approach.

As multiple confidential attributes are present in our scenario, in the following we apply some known results about

density contour of multi-variate normal distribution from statistics.

Proposition 1 (Constant probability density contour) *([9], page 134) Let $\mathcal{Z}$ be distributed as $N_p(\mu, \mathbf{\Sigma})$ with $\mid \mathbf{\Sigma} \mid > 0$. Then, the $N_p(\mu, \mathbf{\Sigma})$ distribution assigned probability $1 - \alpha$ to the solid ellipsoid $\{\mathbf{z} : (\mathbf{z} - \mu)^{'} \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu) \leq \chi_p^2(\alpha)\}$, where $\chi_p^2(\alpha)$ denotes the upper (100α-th) percentile of the χ_p^2 distribution with p degrees of freedom. The ellipsoid is centered at μ and have axes $\pm c\sqrt{\lambda_i}\mathbf{e}_i$, where $c^2 = \chi_p^2(\alpha)$ and $\mathbf{\Sigma}\mathbf{e}_i = \lambda_i \mathbf{e}_i, i = 1, \cdots, p$.*

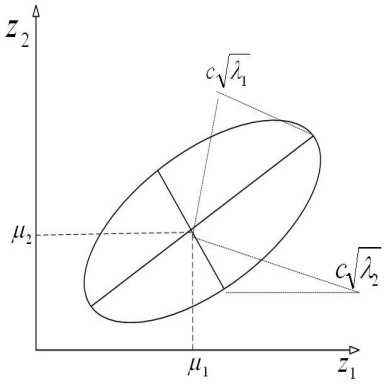


Figure 2. A constant density contour for a bivariate normal distribution

The multi-variate normal density is constant on surfaces where the squared distance $(\mathbf{z} - \mu)^{'} \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu)$ is constant c^2. The chi-square distribution determines the variability of the sample variance. Probabilities are represented by volumes under the surface over regions defined by intervals of the z_i values. The axes of each ellipsoid of constant density are in the direction of the eigenvectors of $\mathbf{\Sigma}^{-1}$ and their lengths are proportional to the the square roots of the eigenvalues (λ_i) of $\mathbf{\Sigma}$.

The ellipsoid $\{\mathbf{z} : (\mathbf{z} - \mu)^{'} \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu) \leq \chi_p^2(\alpha)\}$, which is yielded by the paths of $\mathbf{z}$ values, contains a fixed percentage, $(1 - \alpha)100\%$ of customers. In our scenario, snoopers may use various techniques to estimate and predict the confidential values of individual customers. However, all confidential information which snoopers can learn is the bound of ellipsoid.

Figure 2 shows one constant density contour containing 95% of the probability under the ellipse surface for one bi-variate $\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$, which follows a bi-variate normal distribution $N(\mu, \Sigma)$ with $\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$ and $\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$. $\lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}$ is the eigenvalues of covariance matrix $\mathbf{\Sigma}$ and two axies have length of $c\sqrt{\lambda_1}$ and $c\sqrt{\lambda_2}$ respectively, here $c = 2.45$ as $\sqrt{\chi_2^2(0.05)} = \sqrt{5.99} = 2.45$. We can see the major axis of ellipse is associated with the largest eigenvalue (λ_1).

To compute the projection of one ellipsoid on each axis, we have the following results as shown in Proposition 2.

Proposition 2 (Simultaneous Confidence Intervals) *Let $\mathbf{Z}$ be distributed as $N_p(\mu, \mathbf{\Sigma})$ with $\mid \mathbf{\Sigma} \mid > 0$. The projection of this ellipsoid $\{\mathbf{z} : (\mathbf{z} - \mu)^{'} \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu) \leq \chi_p^2(\alpha)\}$ on axis $\mathbf{z_i} = (0, \cdots, 1, \cdots, 0)^{'}$ (only the ith elemenet is 1, all other elements are 0) has bound:*

$$[\mu_i - \sqrt{\chi_p^2(\alpha)\sigma_{ii}}, \quad \mu_i + \sqrt{\chi_p^2(\alpha)\sigma_{ii}}]$$

See Proof in Appendix B.

To check whether a given distribution of $\mathbf{z}$ may incur value disclosure, our strategy here is to compare the disclosure measure $d(z \mid \hat{z})$ with τ, specified by the database owner. If disclosure occurs, we need to modify parameters $\mu, \mathbf{\Sigma}$. As we know from Proposition 2, the mean vector μ determines the center of ellipsoid or the center of projection interval while the covariance matrix $\mathbf{\Sigma}$determines the size of ellipsoid or the length of projection interval. As the change of μ will significantly affect the data distribution (it will affect the accuracy of analysis or mining subsequently), in the remainder of this paper we focus only on how to change variance matrix $\mathbf{\Sigma}$ to satisfy users' security requirements.

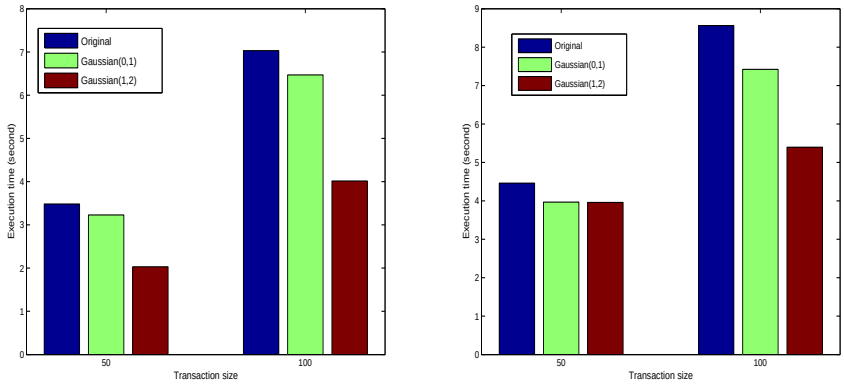
Now we are able to provide adequate security for each individual numerical attribute independently. However, there may exist some non-confidential attributes and linear combination attributes even exist. Although our strategy is able to provide adequate security for individual attributes, the security it provides for linear combinations of attributes could be significantly lower. We are currently investigating how to evaluate and prevent value disclosure when these combinations exist.

6. Experimental Evaluation

The experiments were conducted in a DELL Precision 340 workstation with one 2.4G processor, 1Gbytes of RAM, and 100Gbytes hard disk. The operating system is Microsoft Windows 2000 and the database system is Oracle9i.

6.1. The Effect of Data Distribution on Workload Performance

We used two benchmark datasets, TPC-C [15] and TPC-H [16] from Transaction Processing Performance Coun-



(a) Original TPC-C Workload

(b) Modified TPC-C Workload

Figure 3. Comparison of workload execution time on TPC-C with varied distributions

cil. The TPC-C benchmark is widely used for comparing databases running a medium complexity online transaction processing workload. Its workload is primarily a transaction processing workload with multiple SQL calls per transaction. The original TPC-C workload includes five transaction types: New Order (43%), Payment(44%), Order Status(4%), Delivery(5%), and Stock Level(4%). See [15] for a more thorough treatment.

An important aspect of TPC-C is that it specifies a non-uniform random number generation function to generate tuple-ids. The non-uniform random number generating function was defined in [15] as follows:

$$NU(A, x, y) = (((rand(0, A) \mid rand(x, y)) + C) \% (y - x)) + x \quad (3)$$

where

$rand(x, y)$ stands for randomly selected within $[x, y]$

C is a constant within $[0..A]$

A is a constant chosen according to the size of the range $[x, y]$

exp-1 $\%$ exp-2 stands for exp-1 modulo exp-2

exp-1 $\mid$ exp-2 stands for the bitwise logical OR operation between exp-1 and exp-2

In our experiment, we first generated one TPC-C dataset using the above default distribution and then replaced the original distribution with two Gaussian distributions. The first one is a standard Gaussian distribution with mean 0 and standard variance 1 while the second with mean 1 and standard variance 2. The number of warehouses (w) was set as 10 for all three datasets. Figure 3(a) shows the comparison of workload execution time on TPC-C generated by these three distributions. Note that execution time on Gaussian(1,2) is 40% less than that on original distribution. We also changed the composition of TPC-C workload as follows: New Order (30%), Payment(30%), Order Status(15%), Delivery(25%), and Stock Level(10%). We got similar results as shown in Figure 3(b). We observe most transactions in TPC-C workload are insert transaction and data distribution has relatively less effect on execution time of insert transaction.

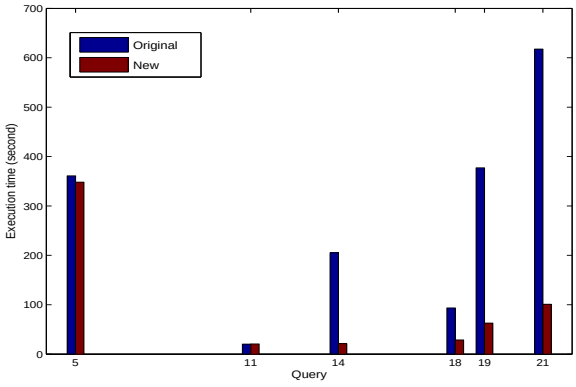


Figure 4. Comparison of query execution time on TPC-H with varied distributions

Our second experiment was done with TPC-H. TPC-H is an ad-hoc decision support benchmark. One important aspect of TPC-H is that knowledge about query workload is not assumed, i.e., database administrator does not know which queries will be executed against database. We first generated a 1GB dataset with default distribution (scale factor SF =1) and then generated a new 1GB dataset with a modified distribution. In the new dataset, Orders and LineItem tables were generated in a manner as fol-

lows: every second customer is not assigned any order and PARTKEY is generated randomly from [1 ..SF*2,000] [3]. We run queries 5, 11,14,18,19, and 21 on these two data sets. Figure 4 shows comparison of query execution time. Note the execution times of query 14, 18, 19, 21 on two data sets are significantly different (10 times different) while that of query 5 and 11 are close. We observe most queries in TPC-H workload are aggregate ones and query 14, 18, 19 and 21 access Orders and LineItem tables.

6.2. Performance of Disclosure Analysis

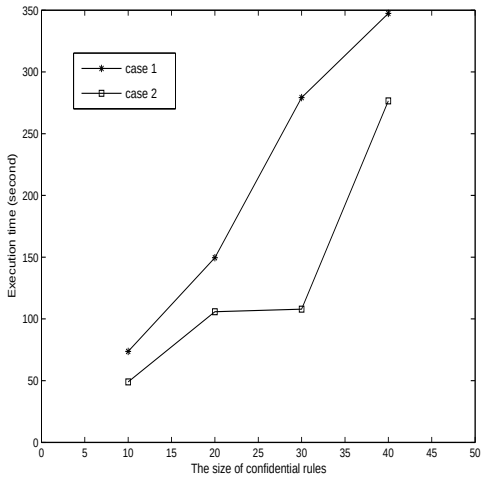


Figure 5. Comparison of disclosure analysis performance vs. the size of confidential rules

The Mortgage dataset we used in our experiment contains nearly 250k tuples. The phase of generating data from the general location model is very fast compared with disclosure analysis. We extracted 1091 and 1561 rules from Mortgage dataset (shown as case 1 and case 2 respectively in figure 5). For each case, we specified the number of confidential rules as 10, 20, 30 and 40 respectively. Figure 5 shows the performance of our disclosure analysis. We can see disclosure analysis takes less than 6 minutes. As we screened confidential rules one by one, the execution time is almost linear of the size of confidential rules.

7. Conclusion and Future Work

In this paper we investigated how to conduct disclosure analysis on the general location model which is used to generate synthetic data. Hence the synthetic database generated has similar distributions or patterns as the production database while preserving privacy. There are some aspects of this work that merit further research. Among them, we are trying to figure out how to better screen out confidential information from released characteristics, especially when linear combinations exist among numerical attributes. We will also conduct a complete study on how different data distributions affect workload performance using TPC Benchmarks. Another area for future work is centered on refining the architecture of the data generator itself. This could include changes to allow further use of real world data sources (e.g., historical data) for increased realism and more rapid adjustment to emerging data trends or perturbation.

[3]The original setting was every third customer is not assigned any order and PARTKEY is generated randomly from [1..SF*200,000], refer [16] for details

8. Acknowledgments

This work was supported in part by U.S. National Science Foundation CCR-0310974. The authors would like to thank Songtao Guo, Jing Jin and Amol Kedar for useful discussions.

References

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 439–450. Dallas, Texas, May 2000.

[2] M. Chan and S. Cheung. Applying white box testing to database applications. Technical report HKUST-CS99-01, 1999.

[3] D. chays, S. Dan, P. Frankl, F. Vokolos, and E. Weyuker. A framework for testing database applications. In *Proceedings of the ISSTA*. Portland, Oregon, 2000.

[4] D. Chays, Y. Deng, P. Frankl, S. Dan, F. Vokolos, and E. Weyuker. Agenda: a test generator for relational database applications. Technical report, Polytechnic University, 2002.

[5] A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables given marginal totals and decomposable graphs. *PNAS*, 97(22):11885–11892, 2000.

[6] A. Dobra and S. E. Fienberg. Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation. *Statistical Journal of the United Nations ECE*, 18:363–371, 2001.

[7] J. Fagan. Cell suppression problem formulations- exact solution and heuristics. August 2001.

[8] J. Gray, P. Sundaresan, S. Englert, K. Baclawaski, and P. J. Weinberger. Quickly generating billion-record synthetic databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 243–252, June 1994.

[9] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 1998.

[10] S. Leutenegger and D. Dias. A modeling study of the tpc-c benchmark. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 22–31. Washington, D.C., May 1993.
[11] Niagara. http://www.cs.wisc.edu/niagara/datagendownload.html.
[12] M. Poess and J. Stephens. Generating thousand benchmark queries in seconds. In *Proceedings of the 30th VLDB Conference*, pages 1045–1053, 2004.
[13] Quest. http://www.quest.com/datafactory.
[14] J. Stephens and M. Poess. Mudd: A multi-dimensional data generator. In *Proceedings of the 4th International Workshop on Software and Performance*, pages 104–109, 2004.
[15] TPC-C. *TPC-Benchmark C*. Transaction Processing Performance Council, 1998.
[16] TPC-H. *TPC-Benchmark H*. Transaction Processing Performance Council, 2003.
[17] X. Wu, Y. Wang, and Y. Zheng. Privacy preserving database application testing. In *Proceedings of the ACM Workshop on Privacy in Electronic Society*, pages 118–128, 2003.
[18] X. Wu, Y. Wang, and Y. Zheng. Statistical database modeling for privacy preserving database generation. In *Proc. of the 15th International Symposium on Methodologies for Intelligent Systems*, pages 382–390, May 2005.
[19] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the 7th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406. San Francisco, CA, August 2001.

A. Cell Suppression

Several cell suppression problem formulations have been given in [7]. It is not directly applicable to our questions since some values in the original contingency table are unknown in our case. In the following, we modify these formulations so that they could be used in our case.

Let $a = \{a_i\}_{i=1}^n$ be a contingency table. Some of these values a_i may be undefined. If a_i is undefined, then we write $a_i = *$. Let H be an m by n matrix describing the linear constraints on the feasible contingency table. That is, a feasible contingency table y should satisfy

$$\begin{aligned} Hy &= 0 \\ y &\geq 0. \end{aligned}$$

Let $P = \{i_1, \ldots, i_p\} \subset \{1, \ldots, n\}$ be the original specified suppressions. For each $i_k \in P$, there is a lower bound requirement l_k and an upper bound requirement u_k. The interpretation of these bounds is as follows. For each $i_k \in P$, the dataset owner does not want the public to infer from the published data that the value of c_{i_k} lies in $(a_{i_k} - l_{i_k}, a_{i_k} + u_{i_k})$. In order to achieve this goal, some other cells of the contingency table may need to be suppressed also. Thus the aim of the suppression is to find a set $C = \{j_1, \ldots, j_c\}$ that is as "small" as possible and that, for each $k = 1, \ldots, p$, there are two feasible contingency tables y and z with the following properties:

1. $y_{i_k} \geq a_{i_k} + u_{i_k}$,
2. $z_{i_k} \leq a_{i_k} - l_{i_k}$,
3. $y_i = z_i = a_i$ for $i \notin P \cup C$.

In some cases, only the bound u_{i_k} is given. Then we do not need to meet the requirement for the existence of z.

A.1. Exact Solution

We use a binary variable array $x = (x_1, \cdots, x_n)$ to represent whether each cell is suppressed. That is,

$$x_i = \begin{cases} 1 & \text{if } i \in P \cup C \\ 0 & \text{otherwise} \end{cases}$$

Let c be an array of subjective weight on cells and y^k, z^k be variables representing the potential feasible contingency tables for the condition $i_k \in P$. Then the suppression problem could be formulated as the following questions.

$$\text{minimize } c^T x \tag{4}$$

subject to

$$\begin{cases} a_i - a_i x_i \leq y_i^k \leq a_i + x_i T \text{ for } a_i \neq * \\ a_i - a_i x_i \leq z_i^k \leq a_i + x_i T \text{ for } a_i \neq * \\ y_{i_k}^k \geq a_{i_k} + u_{i_k}, k = 1, \ldots, p \\ z_{i_k}^k \leq a_{i_k} - l_{i_k}, k = 1, \ldots, p \\ Hy^k = 0; Hz^k = 0, k = 1, \ldots, p \\ x_i = 1 \text{ for } i \in P \\ x \in \{0,1\}^n; y^k \geq 0, z^k \geq 0, k = 1, \ldots, p \end{cases} \tag{5}$$

where T is a large enough integer. The above mixed integer programming (MIP) could be split into small MIPs. That is, for each $i_k \in P$, one can construct an MIP and solve it.

A.2. Heuristic methods

When the number of variables increase, it may be infeasible to solve the MIPs in the previous section. Fagan [7] recommended several heuristic mehtods. In the following, we describe one that is useful for our problem.

Let y and z be n-ary variables, and c be the subject value such that it takes 0's on known parts of $P \cup C$. We hope that $a + y - z$ will be the feasible contingency table witnessing the fact that the public cannot infer the upper bound of a_{i_k} within an error of u_{i_k} if we suppress these cells j with $y_j \neq z_j$. That is, we want to have $a_{i_k} + y_{i_k} - z_{i_k} = a_{i_k} + u_{i_k}$ and keep the nonzero entries in $y - z$ as small as possible according to the subjective weight. The heuristic formulation is as follows:

$$\text{minimize } c^T(y + z) \tag{6}$$

subject to

$$\begin{cases} z_i \le a_i \text{ for } a_i \ne * \\ H(y-z) = 0 \\ y_{i_k} - z_{i_k} \ge u_{i_k} \\ y \ge 0, z \ge 0 \end{cases} \tag{7}$$

For the lower bound, we have the similar linear programming formulations as follows.

$$\text{minimize } c^T(y+z) \tag{8}$$

subject to

$$\begin{cases} z \le a \\ H(y-z) = 0 \\ z_{i_k} - y_{i_k} \ge l_{i_k} \\ y \ge 0, z \ge 0 \end{cases} \tag{9}$$

Remark. In some cases, it may be possible that the bounds l_{i_k} and u_{i_k} are not given directly. That is, instead of giving l_{i_k} and u_{i_k}, only $L_{i_k} = a_{i_k} - l_{i_k}$ and $U_{i_k} = a_{i_k} + u_{i_k}$ are given. When a_{i_k} is given, then one can easily compute l_{i_k} and u_{i_k} directly. If a_{i_k} iis unknown, then one may estimate the values of l_{i_k} and u_{i_k} roughly as

$$l_{i_k} = u_{i_k} = \frac{U_{i_k} - L_{i_k}}{2}.$$

B. Proof of Simultaneious Confidence Internvals

Proposition 1 (Simultaneous Confidence Intervals) *Let $\mathcal{Z}$ be distributed as $N_p(\mu, \mathbf{\Sigma})$ with $|\ \mathbf{\Sigma}\ | > 0$. The projection of this ellipsoid $\{\mathbf{z} : (\mathbf{z}-\mu)^{'}\mathbf{\Sigma}^{-1}(\mathbf{z}-\mu) \le \chi_p^2(\alpha)\}$ on axis $\mathbf{z_i} = (0, \cdots, 1, \cdots, 0)^{'}$ (only the ith elemenet is 1, all other elements are 0) has bound:*

$$[\mu_i - \sqrt{\chi_p^2(\alpha)\sigma_{ii}}, \quad \mu_i + \sqrt{\chi_p^2(\alpha)\sigma_{ii}}]$$

Proof. The general result concerning the projection of an ellipsoid onto a line in a p-dimensional space is shown as follows ([9], page 203).

For a given vector $\ell \ne \mathbf{0}$, and $\mathbf{z}$ belonging to the ellipsoid $\{\mathbf{z} : \mathbf{z}^{'}\mathbf{A}^{-1}\mathbf{z} \le c^2\}$ determined by a positive definite $p \times p$ matrix $\mathbf{A}$, the projection (shadow) of $\{\mathbf{z}^{'}\mathbf{A}^{-1}\mathbf{z} \le c^2\}$ on ℓ is $c\frac{\sqrt{\ell'\mathbf{A}\ell}}{\ell'\ell}\ell$ which extends from $\mathbf{0}$ along ℓ with length $c\sqrt{\frac{\ell'\mathbf{A}\ell}{\ell'\ell}}$. When ℓ is a unit vector, the shadow extends $c\sqrt{\ell'\mathbf{A}\ell}$ units, so $|\ \mathbf{z}^{'}\ell\ | \le c\sqrt{\ell'\mathbf{A}\ell}$.

From the above, we know the projection of an ellipsoid $\{\mathbf{z} : \mathbf{z}^{'}\mathbf{A^{-1}}\mathbf{z} \le c^2\}$ on a given unit vectior ℓ has length $len = c\sqrt{\ell'\mathbf{A}\ell}$. We replace $\mathbf{A}$ with

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & . & . & . & \sigma_{1p} \\ \sigma_{12} & \sigma_{22} & . & . & . & \sigma_{2p} \\ . & . & . & . & . & . \\ \sigma_{i1} & \sigma_{i2} & & \sigma_{ii} & & \sigma_{ip} \\ . & . & & & . & . \\ \sigma_{p1} & \sigma_{p2} & . & . & . & \sigma_{pp} \end{pmatrix}$$

, replace ℓ as $\mathbf{z_i} = (0, \cdots, 1, \cdots, 0)^{'}$, and replace c as $\sqrt{\chi_p^2(\alpha)}$, then we get the length of projection as $len = \sqrt{\chi_p^2(\alpha)\sigma_{ii}}$. Considering the center of this ellipsoid, we have the bound as $[\mu_i - \sqrt{\chi_p^2(\alpha)\sigma_{ii}}, \quad \mu_i + \sqrt{\chi_p^2(\alpha)\sigma_{ii}}]$.