

# Survey of Techniques for Securing Statistical Databases

JASON M. SCHATZ  
*University of California at Davis*

Abstract:

*Research in Controlling Inference in statistical databases (SDBs) has been under way for over 20 years, and certain methods of formalizing the inference problem have become very well established. This paper gives an introduction to a number of these methods. It begins with an explanation of the standard notation for statistical queries. It then gives a definition of what it means for a SDB to be compromised and discusses the most common taxonomy used to classify techniques for preventing compromise. The final section of the paper will describe a number of specific inference control systems, then conclude with a discussion of future directions for research.*

## Introduction

There are many implemented examples of SDBs that provide statistical resources. The Census Bureau database is the canonical model, but insurance companies, HMO's, universities, and many other types of organizations keep databases that contain valuable information. In most of these examples the records on which statistics are calculated contain sensitive data.

There is ideally a tacit understanding between the entity maintaining a SDB and statistical researchers, to the effect that all statistics are public, but the contents of individual records are not. Unfortunately, even though access control methods and a limited querying language keep a statistical researcher from directly reading a single record, it is easy to infer the contents of specific records from statistical data. A SDB that does not take steps to control inference is relying on the good will of its users to keep sensitive records secure. From the tension between the conflicting goals of providing statistics, and securing individual records, we get the field of inference control in statistical databases.

## Part I: Overview of the Inference Control Problem

### Formalizing the notion of a Statistical Query

A Statistical Database (SDB) differs from a more typical database primarily in its very limited querying interface. Querying is typically limited to operations such as count, sum, mean, and a few other statistical calculations, which are performed on subsets of the relation. SDB underlying data schemes do not necessarily differ from more typical databases. All popular formalizations of SDBs assume data is organized according to a relational scheme.

For the sake of simplicity, a SDB can be modeled as a single relation  $R$ . In actuality most SDBs are implemented on large relational databases with many relations. Consider  $R$  to be the flat table produced from joining all relations in the database; the universal relation. A statistical query is concerned with finding an aggregate property of a subset of the relation  $R$ . For example, a query on a SDB maintained by an auto insurance company might be informally written as “How many car accidents which involving red cars occur before 6:00AM” ( See table 1 for an example  $R$ ).

Formally specifying a statistical query requires a notation for specifying subsets of a relation, and a notation for specifying which aggregate statistic of that subset will be calculated.

The table  $R$  is defined as a table with  $N$  records. Each record has  $m$  attributes and for each record the  $i^{\text{th}}$  ( $i = 1, \dots, m$ ) field contains the value of the  $i^{\text{th}}$  attribute  $A_i$ .

A subset of  $R$  is selected by a characteristic formula  $C$ , which is loosely defined as a logical formula over the values of the attributes of  $R$  using the operations  $\wedge$  (and),  $\vee$  (or), and  $\neg$  (not) [3]. Table 1 shows a set of auto accident records that might be kept by an insurance company. The subset of records involving red cars and accidents that occur before 6:00AM can be specified by the characteristic formula

$$C = (\text{Color of Automobile} = \text{Red}) \wedge (\text{Time} < 0600)$$

$X_C$  denotes the set of records that satisfy characteristic formula  $C$ . Although  $C$  itself is not a set, it is often easier to read statements which abuse set notation slightly by writing  $C_1 \subseteq C_2$  to denote  $X_{C_1} \subseteq X_{C_2}$ , and  $|C|$  to denote  $|X_C|$  [3].  $D$  is the characteristic formula that is true for all records, therefore  $C \subseteq D$  is true for all characteristic formulas  $C$ .

**Table 1**

Customer Name	Customer Age	Auto Manufacturer	Color of Automobile	Time	At Fault	DUI
J. Parks	21	Honda	Blue	1330	0	1
G. Nguyen	18	Audi	White	0300	0	0
P. Lindstrom	35	Honda	Yellow	1700	1	0
C. Coffee	67	Toyota	Blue	1800	1	1
K. Kuhnhausen	35	Chevrolet	Red	1200	0	0
C. Parks	20	Honda	White	0900	0	0
E. Easterly	21	GM	Silver	1230	0	0

P. Lindstrom	35	Honda	Red	0530	0	1
C. Warner	41	Toyota	Red	0400	1	1
C. Jong	53	Chevrolet	Green	0730	0	0
J. Boucher	24	Volkswagon	Gold	2100	1	1
C. Warner	34	Honda	Blue	1100	0	0

There are two query types that will be used as examples in this paper. They are COUNT and SUM.  $COUNT(C)$  returns the number of tuples in C.  $SUM(A_i, C)$  returns the sum of the values of attribute  $A_i$  for all tuples in C. Using the characteristic formula defined above for C, and table 1 for R gives the following statistics.

$$COUNT(C) = 2$$

$$SUM(Age, C) = 76$$

Note that we could denote  $MEAN(A_i, C)$  as  $SUM(A_i, C) / COUNT(C)$ .

$$MEAN(Age, C) = 38$$

In general,  $Q(C)$  will denote any query on the characteristic formula C.

### Defining the notion of Compromise

No statistical query constitutes a compromise of an SDB in its own right. Optimally a researcher with good intentions should be able to form any interesting characteristic formula, and perform any statistical measurement of the corresponding set of records. Unfortunately, it is easy to show that a user can form statistical queries that can be used to infer specific field values in the database. In general this is unacceptable. We say a SDB has been compromised when a user, through one or more queries, amasses information that can be used to infer the value of a specific field. The definition is refined by dividing compromises into two categories: exact, and partial [2].

An attribute's domain may have numerical values or Boolean values. We say an SDB is exactly compromised if a malicious user can infer the value 1 in a Boolean field, or the exact value of a numerical field.

We say a SDB has been partially compromised if a user can infer the value 0 for a Boolean field, or can estimate the value of a numerical field to a precision specified by the SDBA. If the exact value of a field is A, and the estimated value is A', then the SDB has been partially compromised if  $|A'/A| < k^2$  for a given k.

If a user is allowed to query the insurance SDB represented by table 1 without restrictions, he can easily deduce the value of specific fields, particularly if he has some knowledge of the table's contents a priori. For example, if a user knows that Patrick Lindstrom was in an accident involving his yellow car, and he wants to know if Pat was at fault, he could form the equation:

$$C = (\text{Name} = \text{P. Lindstrom}) \wedge (\text{Color} = \text{Yellow})$$

Then obtain the following query results

$$\begin{aligned} \text{COUNT}(C) &= 1 \\ \text{SUM}(\text{At Fault}, C) &= 1 \\ \text{SUM}(\text{DUI}, C) &= 0 \end{aligned}$$

The user has now surreptitiously deduced that Patrick was at fault in the accident, a specific compromise, and that he was not DUI, a partial compromise. The user can, in fact, determine any value in the record of this crash.

### Effective methods for compromising SDBs

The attacker in the previous example was able to compromise the SDB by finding a characteristic formula that gave him a query set of size one. This is not a very subtle attack, and it is thwarted if the SDBA sets a minimum query size  $k$ , so that all queries for which  $|C| < k$  are rejected. The SDBA must also set a maximum query size of  $N - k$ , to avoid a similar attack based on complements of sets.

There are many inference methods, however, which can be used to compromise a SDB regardless of the restrictions on set size. One example is the “tracker” method by Denning and Schlorer [4]. By properly using a tracker, a mischievous researcher can calculate the value of any query, including queries with a query set of size one, so long as the SDB’s minimum query size is set no higher than  $k = N/6$ .

Before discussing the details of using a tracker, it is worth noting the severity of this challenge to SDB administrators. If a user can compromise a SDB with a minimum query size no greater than  $k = N/4$ , then it is ineffectual to set a small constant minimum query size (ie.  $k = 10$ ) for any reasonably sized set of records. In fact, for a moderately sized SDB, with 80,000 records, the minimum query set would have to be greater than 20,000 records.

To use the tracker method, a snooper first must find a characteristic formula with a legal query set size. This formula is referred to as the tracker, denoted  $T$ , where:

$$2k \leq |T| \leq N - 2k$$

In general finding a suitable  $T$  is not difficult. For example, let us assume that the minimum query set size for the SDB managing table 1 is  $k = 12/6 = 2$ . The maximum allowable query size would be  $N - k = 10$ . The user needs to find a  $T$  such that  $4 \leq |T| \leq 8$ . A user might begin by guessing that roughly half of the records in table 1 involve drivers older than 25. She could test this by specifying the following  $T$ :

$$T = (\text{Age} < 25)$$

Testing her T by querying the SDB she would find that  $\text{COUNT}(T) = 5$ . It is within the necessary range.

Having obtained a workable tracker, the user can now calculate the value of any query. Using Q to denote any query (CONT, SUM, MEAN ...), the user first calculates Q(D) by summing  $Q(T) + Q(T^c)$ . For COUNT queries,  $\text{COUNT}(T) + \text{COUNT}(T^c) = N$ . For SUM queries,  $\text{SUM}(A_i, T) + \text{SUM}(A_i, T^c)$  is the sum over all the records of the attribute  $A_i$  ( $\text{SUM}(A_i, D)$ ). The user can now calculate  $Q(C)$  with the following equation:

$$Q(C) = Q(C \vee T) + Q(C \vee T^c) - Q(D)$$

As an example from table 1, the user could use the tracker method to determine that the query set for  $C = (\text{Name} = \text{P. Lindstrom}) \wedge (\text{Color} = \text{Yellow})$  has a size of one. She could use the method again to determine whether or not the driver was at fault.

$$\begin{aligned} \text{SUM}(\text{At fault}, C) &= \text{SUM}(\text{At Fault}, (C \vee \text{Age} < 25)) + \\ &\quad \text{SUM}(\text{At Fault}, (C \vee \text{Age} \geq 25)) - \\ &\quad \text{SUM}(\text{At Fault}, D) = 3 + 3 - 5 = 1 \end{aligned}$$

The user has partially compromised the SDB by inferring that Mr. Lindstrom was At Fault in his accident. The deviant user made one query to find a tracker, and four queries to infer the value of the At Fault field. All five queries were of legal size.

The tracker method works by double counting the query results of the records specified in C. Note that the record corresponding to Lindstrom's crash in his yellow car is selected by both  $(C \vee T)$  and  $(C \vee T^c)$ , but records not specified by C are not double counted. Subtracting out the query value for all records ( $Q(D)$ ) leaves only the value of the records which were double counted, or  $Q(C)$ .

Denning and Schlorer refer to the tracker in this example as a *general tracker*. They describe inference techniques they refer to as *double trackers*, and *union trackers*, and prove that with the later type, values to any query can be calculated even when the only allowable queries are those involving  $\frac{1}{2}$  the population [4]. Furthermore, they argue that empirical data shows that trackers can usually be guessed in one or two guesses for most implemented SDBs, and prove that a tracker can be found in  $O(\log_2 S)$  queries, where S is the number of distinct records possible. All told, their attack on query size thresholds as a SDB protection method leaves little doubt that stronger methods are required.

### **A taxonomy of inference control techniques**

A wide range of more powerful inference control techniques have been proposed. A common classification scheme supplied by Adam and Wortman [1] divides the techniques into four categories: query set restriction, data perturbation, output perturbation and conceptual approaches.

The simplest query set reduction technique, limiting query set sizes, has already been discredited. In general most control techniques have been shown to be ineffective. This paper will discuss a technique called Cell Suppression, which is of interest because it has been successfully employed by the Census Bureau. In general, query set reduction techniques which prevent exact compromise severely restrict the usefulness of an SDB.

Data perturbation prevents inference by creating a proxy database from the true database. All statistical queries are calculated on the proxy database. The individual cells of the proxy database contain values that are variations of their corresponding values in the true database. In general database perturbation schemes try to minimize bias in query results by distributing bias in data so that it will cancel out in large query sets.

Output perturbation schemes calculate responses to queries on the SDB, and then add variance to the result. They very low storage and computational overhead, and are fairly easy to implement since they do not effect the querying process. They do not benefit, however, from the averaging affect of data perturbation.

Notice the fundamental difference between the failings of query set reduction techniques and the failings data perturbation techniques. Query set reduction techniques cannot avoid inference, but they provide researchers with accurate responses to valid queries. Data perturbation techniques can prevent inference, but they cannot consistently provide useful query results. No technique in any of these categories provides an optimal solution, one which avoids inference and allows a researcher to collect accurate statistical data.

The taxonomy includes a final category for techniques that attempt to solve the inference problem by altering the conceptual level scheme on which the SDB is built. To my knowledge none of these schemes have been susesfully implemented, and there are no projects currently underway to implement them.

### **Evaluating the effectiveness of an inference protection technique**

A number of terms are commonly used in the SDB security community to discuss different measures of inference control technique effectiveness. This particular list selected from a list supplied by Adam and Wortman [1].

**Security:** Security is the most fundamental measurement of performance, and it refers to a systems ability to avoid both partial and exact compromise.

**Robustness:** Robustness concerns the assumptions a system makes about an attackers supplementary knowledge of the SDB. If an inference control system can insure security only under the assumption that a mischievous user has no supplementary knowledge of the records in the database, then we say the system is not robust.

**Bias:** Bias represents the variance between the actual value of a statistic, and the perturbed value returned by an SDB. For a query set reduction control system, there is no bias. Perturbation techniques attempt to minimize bias.

**Precision:** Precision refers to the variance in a system's bias. A useful perturbation technique must be able to limit bias to within a specified confidence interval.

**Consistency:** Consistency represents the ability of a perturbation based system to respond to queries without contradictions or paradoxes. When the values of data, or query responses are perturbed, there is a possibility that comparing responses to two queries will reveal arithmetic inconsistencies. Given a characteristic formula  $C$ , for example, the query  $SUM(A_i, C)$  divided by  $COUNT(C)$  might not be equal to the  $MEAN(A_i, C)$ . We refer to these inequalities as contradictions. Paradoxes refer to responses which violate properties of statistical queries. A negative response to a  $COUNT$  query is a paradox.

Finally, inference control methods can be evaluated by cost, both in terms of set-up up time, and processing overhead.

## **Part II – Introduction to specific inference control techniques**

The following descriptions of specific inference control techniques are summaries of descriptions from two sources, a survey by Adam and Wortman [1], and a chapter from "Database Security", by Castano et al [2]. They contain a description of the strategy employed by each system, and a discussion of their capabilities and shortcomings.

### **Approximate Data Swapping**

Approximate Data Swapping is a data perturbation method based on the notion that a significant amount of the valuable information in an SDB is captured in its first  $t$ -ordered statistics. A  $t$ -ordered statistic is a statistical quantity that can be calculated from  $t$  attributes. Taking an example from table 1, we would say  $COUNT(\text{Color} = \text{Red} \wedge \text{Make} = \text{Honda})$  is a two-order statistic.

The data swapping method creates a new database from an existing SDB by swapping attribute values between records without altering the values of any  $t$ -order statistics for some number  $t$ . Statistics of order higher than  $t$  are not necessarily equivalent to those of the original database. Increasing  $t$  decreases bias in higher order statistics. Unfortunately, increasing  $t$  also increases the ability of a snooper to compromise the database through inferences on queries based only on  $t$ -order statistics.

Because the algorithm used to generate the perturbed database requires a significant amount of computation, the data swapping method is not suitable for SDBs which are dynamically updated.

### **Random Sample queries**

The US Census Bureau uses a random sampling technique to prevent inference on their tabulated statistics. Each published query is based on a randomly selected sub-population of the query set. The Census Bureau uses this technique very successfully, but there is a fundamental difference between statically published queries, and dynamically calculated queries. The Census Bureau only needs to produce a single random sampling per query. In a SDB, a new sampling must be selected for each query. By repeating the same query many times and averaging the results, a snooper can eliminate the bias introduced through sampling, then apply standard inference techniques.

### **Fixed Perturbation**

Like data swapping, fixed perturbation techniques create an alternate database from the original data. The method can only be applied to numerical data. As the name implies, the alternate database is created by altering the value of each field by a randomly generated perturbation value.

Because the perturbation is done only once, repeated queries have consistent values. There is no averaging attack that will provide a snooper the original data values.

Perturbation values are Normally distributed with  $\mu = 0$ , and an appropriate standard of deviation  $\sigma^2$ . For a fixed query set and a given attribute, the sum of the perturbation values is expected to be zero. The bias introduced, however, by the system to SUM queries is not as low as this implies. Additional bias is introduced because attribute values in the characteristic formula are also altered. A characteristic formula will not select the same records in the perturbed database as in the original. Altering the query set can introduce bias that is difficult to calculate, but Norm Matloff has published calculations suggesting that under reasonable conditions bias might be as high as 50%.

### **Query Based Perturbation**

Query based perturbation is a data perturbation technique that does not require the creation of a proxy database. For each query made on the SDB, a perturbation function is applied to all attributes that affect the resulting value. The bias introduced by the function is not stored in the database, and is different for individual queries.

For COUNT queries the bias function is applied to all attributes used in the characteristic equation. The altered query set has a biased count. For other aggregate functions the query set is selected on the unbiased data. Once the correct records have been selected, the biasing function is applied to attributes involved in the aggregate function. This

technique does not introduce the query set bias that makes fixed perturbation difficult to implement, but it is vulnerable to attacks that average the value a large number of queries. This technique also has the disadvantage of allowing inconsistencies and paradoxes in query results.

## **Rounding**

Rounding is a result based perturbation technique. All queries are calculated on unbiased data, then the results are altered before being returned to the user. A result based perturbation that added only a randomly generated bias would have a very poor tradeoff between security and precision. The rounding technique does not involve random bias generation, rather, results are rounded to nearest multiple of a value  $b$ , set by the SDBA. The technique is not vulnerable to averaging attacks since query results are equal for repetitions of the same query. There are, however, a number of more complicated techniques which allow exact compromise. Like many inference control schemes, rounding must be used in concert with other techniques to provide security.

## **Conclusions**

To date, there is no effective solution to the inference control problem that can be applied to a wide range of SDBs. No generally applicable solution provides both a high level of security, and unbiased statistics for a rich set of queries. There has been, however, some measure of success found by relaxing the goals of inference control in two ways; by using a definition of security that is less difficult than fully avoiding exact and partial compromise, and by tailoring inference control solutions to specific categories of SDBs.

As an example of a relaxed security goal, Adam and Wortman [1] offer the following redefinition of inference control: a SDB should prevent exact disclosure, and provide statistical-disclosure control. Statistical disclosure control is an idea put forth by Dalenius [2], which uses statistical properties to describe the accuracy to which the value of a field can be inferred. An inference protection system is said to have the property of statistical disclosure control if it guarantees that an attacker must make a large number of queries to infer the value of a field within a small variance. The SDBA should be able to decide what large and small mean in this context. He should be able to set parameters  $k$  and  $c$  such that if a user makes fewer than  $k$  queries, he can infer the value of a field with a variance not less than  $c$ . The SDBA can then set  $k$  and  $c$  to values such that the attacker has to make impractical number of queries to compromise the database.

This relaxed definition of security can be used to provide a favorable analysis of random query set sampling techniques, query set perturbation techniques, and other techniques which might otherwise be dismissed as being vulnerable to averaging attacks. In general, defining fine grained security goals seems to be a promising technique for exploring situations in which inference control methods provide adequate, if not perfect, security.

A second, and similar, approach for finding solvable inference control goals might be to examine SDBs which offer a functionality more limited than the general case. The Census Bureau's data, for example, can be protected with a random sampling technique because it is queried only once. Similarly, SDBs with only a single secure attribute can be protected through many data perturbation schemes. Presumably there are many limited operating conditions which allow for inference protection, and are still useful.

Finally, there is still work to be done in auditing SDB users to detect query patterns which indicate misuse. Similarly to the problem of computer misuse in general, if it is not possible to prevent inference, it is very useful to be able to detect it, and to look into means of developing both legal and social deterrents.

## References

- 1) N. Adam and J. Wortmann. Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys, Vol. 21, No 4, December 1989.*
- 2) Dalenius, T. 1977 Towards a methodology for statistical disclosure control. *Statistik Tidskrift 15, 429 – 444.*
- 3) Castano, Fugini, Martella, Samarati. Database Security. *ACM Press Books, Adison-Wesley Publishing. 1996.*
- 4) F. Chin and G. Ozsoyoglu. Statistical Database Design. *ACM transactions on Database Systems, Vol. 6, No. 1, December 1981, Pages 113-139*
- 5) D. Denning and J. Schlorer. A Fast Algorithm for Calculating a Tracker in Statistical Database. *ACM Transactions on Database Systems, Vol. 5, No. 1, March 1980*
- 6) J. F. Truab and Y. Yemini. The Statistical Security of A Statistical Database. *ACM transactions on Database Systems, Vol. 9, No. 4, December 1984, Pages 672-679*