A Comparative Study of Query-Set Size and Fixed-Data Perturbation as Two Techniques to Secure Statistical Databases

Shouki A. Ebad
Faculty of Computing & IT
Northern Border University
Rafha 91911, P.O. Box 840
Saudi Arabia

ABSTRACT
A statistical database (SDB) is a database that contains a large number of individual sensitive records, but is intended to supply only statistical summary information to its users. A SDB suffers from the inference problem, a way to infer or derive sensitive data from non-sensitive data. In this study, two security techniques of SDBs, Query-Set Size and Fixed-Data Perturbation are selected to review and compare each other. As a result, no one is a perfect solution for the inference problem. The selection of technique depends on some factors mentioned in this paper.

General Terms
Statistical database, database security, sensitive data.

Keywords
Inference problem, query-set size, fixed-data perturbation

1. INTRODUCTION
A statistical database (SDB) is a database that contains a large number of individual sensitive records, but is intended to supply only statistical summary information to its users, not information referring to some specific individual[1][2]. In other words, the only permissible queries are those that apply some statistical functions such as count, sum, or average to some subset of the records in the database.

Security problems in SDBs arise from the wish to provide statistical information without compromising sensitive information about individuals. This is also referred to as *inference problem*. Inference in the SDB means the possibility of obtaining confidential information on single entities, by taking advantage of a sequence of statistical queries issued against a set of entities stored in the SDB [1][2][3]. The need for confidentiality is extremely important is several applications such as census, military, and health care [2][4]. Techniquesto prevent or avoid statistical inference in SDB could be classified in two approaches: Query-restriction and Perturbation. In order to review and compare these approaches, we chose one technique from each approach: Query-Set Size and Fixed-Data Perturbation.

A "*statistical filter*" is an initial solution to secure SDBs. The filter processesall queries before handing them to the "normal query processor." The statistical filter ensures that [1]:

- A user can only access aggregate data, and
- A user cannot access any directly identifying attribute (e.g., name or Social Security number).

However, a statistical filter is not sufficient to prevent inference, since the released statistical always maintain a trace of the data that have been used in the computation, skilled users can get unauthorized information. Consider a user first querying the SDB about the *average* salary of the women employees of a certain department, and then the same user querying the number of women employees. If this *count* returns the value 1, then the user obtains (infers) the salary of the woman employee by means of legal statistical queries.

The purpose of this study is to describe the inference problem in SDBs and to compare in detail two of the interesting techniques emerged to secure sensitive data in such DBs.Section 2 discusses the concept of sensitivity in data. In Section 3 and Section 4 we present Query-Set Size and Fixed-Data Perturbation techniques respectively. A comparison of the two techniques are made in Section 5; such a comparison is made in terms of security vs. precision, features of computer system, consistency, processing cost, and attribute domains. We summarize the paper in Section 6 with offering some directions for future work. Finally, the references of the study are listed in Section 7.

2. SENSITIVE DATA
Sensitive data [5] is data that should not be made public. Determining which data items and fields are sensitive depends on the individual database and the underlying meaning of data. A public library databases, for instance, contain no sensitive data; defense-related databases are sensitive completely. Nothingsensitive and everything sensitiveare the easiest to handle because they can be covered by access controls to the database as a whole.

The problem is the case in which some but not all of the elements in the database are sensitive. For example, acompany databaseshown in Table 1 containsemployees'data: name, job, location, salary, and tax. Name and location are probably the least sensitive; salary and tax the most; job somewhere in between. hence, many people may have legitimate access to name or location, some to job, and few to salary or tax. Indeed, knowledge of the existence of some fields, such as salary, may itself be sensitive. Thus, security concerns not only the data elements but also their meaning.Although they are all highly sensitive, salary and tax attributes may not have the same kinds of access restrictions. In other words, a few people should be authorized to display each field, but no one be authorized to see both fields. However, to determine which data are sensitive is not easy task [5] and the situation would be complicated by a desire to share non sensitive. Some factors can make data sensitive such as (1) the owner of data, (2) the source of data, and (3) the relation among the data values. More examples for these factors are presented by some researchers [5].

Table 1: Company Database

Name	Job	Loc	Salary	Tax
Ahmed	System Analyst	Riyadh	2000	6
Kareem	Accountant	Dammam	3500	5
Sami	Programmer	Dammam	2000	4
Abdullah	Programmer	Jeddah	3000	4
Fadi	Manager	Jeddah	5000	10
Kamal	System Analyst	Riyadh	3500	6
Adam	System Analyst	Dammam	2000	6
Fareed	Accountant	Jeddah	4000	5
Rashed	Manager	Dammam	4500	10
Samer	Manager	Riyadh	5000	10

Generally, the definition of sensitivity is not precise; it is open to more research. The answer to the question "*what do we mean by 'sensitive data'?*" is not easy as before the current era due to existence of some new technologies such as social networks that played a role in degree of privacy of individuals. Such issue reflects the tension between the usability and privacy of individuals [6]. This point would discussed in Section 5.

3. QUERY-SET SIZE

One of the more obvious techniques is to limit the number of cases (individuals) that qualify as a result of a query, called a *query set*. Back to the company database in Table 1, it is necessary to know some of the characteristics of an individual record. This is not unreasonable since the individual may be personally known. For example, it may be known from outside sourcesthat Sami is a programmer and works in Dammam.

Assume that query Q1 identifies Sami as follows:

Q1:count all where Loc='Dammam' AND Job='Programmer'

In order to find out salary of Sami, guess could be used as in query Q2:

Q2:count all where Loc='Dammam' AND Job='Programmer' AND Salary=2000

Q2 provides the information that the salary of Sami is 2000. In addition, with no guess, skilled users could find the salary of Sami using the aggregate statistic *'sum'* as in query Q3:

Q3:sum Salary where Loc='Dammam' AND Job='Programmer'

As you have seen, finding salary i.e., sensitive data is simple because the query size is one. A sensible restriction on DBMS (Database Management System [2]) is therefore to respond to queries only where the set size is greater than some integer *k*.However, this restricting query set size to exclude small values is not sufficient because large values close to the size of database n, total number of records, allow compromise [2]. For example, query Q4 and query Q5:

Q4:count all where Loc='Dammam' OR Not 'Dammam'

Q5:count all whereNot (Loc='Dammam' AND Job='Programmer')

The set size of Q4 and Q5 is 10 and 9 respectively, the difference between two sizes determines the number of employees in Dammam, namely one: Sami. In such a case, an unauthorized user could use the aggregate statistic *'sum'* to find the salary of Sami as in query Q6 and query Q7:

Q6: sumSalary whereLoc='Dammam' AND Not 'Dammam'

Q7: sum Salary where Not(Loc='Dammam' AND Job='Programmer')

The difference between set sizes of Q6 and Q7 is salary of Sami.

A restriction is then added to prevent such a compromise. The query set size must therefore be restricted to the range [*k*, *n*-*k*] where $1 \leq k \leq n/2$ [1][5][7] such that if the query set is out of this range, then the statistical operation should not be applied over it and the query is refused. Hence, the previous queries, Q1 up to Q7, are not admissible since they violate this restriction.Compared to the previous models, this restriction achieves more security. However, this is inadequate to avoid compromise. Consider the queries Q8 and Q9 queries:

Q8: count all where Loc='Dammam'

Q9: count all where Loc='Dammam' AND Not Job='Programmer'

Assuming k=2, the set size of Q8 and Q9 is 4 and 3 respectively; therefore, it is in the range of [*k*, *n*-*k*] the user can deduce there exist exactly one programmer works in Dammam, who must therefore be Sami (since the user already known that this description fits Sami.) salary of Sami is thus given by the difference between query Q10 and query Q11:

Q10: sum Salary where Loc='Dammam'

Q11: sum Salary whereLoc='Dammam' AND Not Job='Programmer'

The predicate Loc='Dammam' AND Not Job='Programmer' is called an individual *tracker* for Sami[3][4], because it enables the user to track down Sami information. Generally, if the user knows a predicate P that identifies some specific record R, and if P can be expressed in P_1 AND P_2, then the predicate P_1 AND NOT P_2 is a tracker for R (provided that predicates P_1 and (P_1 AND NOT P_2)are admissible, i.e., both identify result sets with size in range *k* to *n*-*k*.) This is because the record set identified by P is identical to the difference between the set identified by P_1 and that by P_1 AND NOT P_2, Figure 1 illustrates this point.

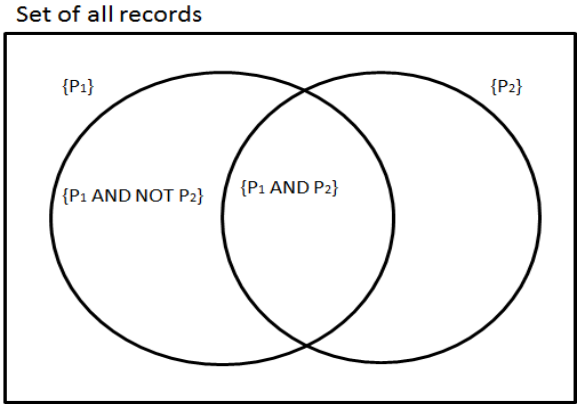

Fig 1 The individual tracker P1 AND NOT P2: Set (P) = Set(P1 AND P2) = Set(P1) – Set (P1 AND NOT P2)

Theindividual tracker may be for three predicates P_1, P_2, and P_3 is P_1 AND NOT (P_2 AND P_3)since Set(P_1 AND P_2 AND P_3) = Set(P_1) – Set (P_1 AND NOT (P_2 AND P_3). Some researchers [4] discussed trackers in detail.

4. FIXED-DATA PERTURBATION

In this section, we discuss the second technique: Fixed-Data Perturbation. Perturbation methods often prevents inference by introducing some type of modification during the processing of a statistical query, with the aim of releasing more statistics than the restriction-based techniques. The modification could be applied either to the records stored in the SDB used for computing the requested statistic (it is discussed below), or to the trueresult before releasing it to the user.

The Fixed-Data Perturbation technique is to perturb the values of the database by a small *error* or noise. If X_i is the true value of field *i* in the database, e_i is a random error term added to X_i. The result of the sum query will be$\sum_{i=1}^{n} X_i + e_i$. Malicious users can not improve their estimates of statistics by repeating queries because the perturbation error e_i of field i is fixed [1]. However, the perturbed value (X_i +e_i) requires additional storage at the computer system. one solution for this drawback is perturbing the elements of X (true value) every time they are accessed [7]. To guarantee that any query receives the same answer, the perturbation of X_i may be obtained using a pseudorandom generator with a fixed *seed* [9].

In general, perturbation via additive error is a common technique in the disclosure control literature [8]. a variant on this technique is proposed by some researchers [1] in which the additive perturbation should be replaced with multiplicative perturbation since the additive perturbation does not prevent the sensitive data properly. Assume the average salaries of Table 1 is 10,000 and a 2000 perturbation may be suitable to *hide* information about salaries that do 10,000 too far. What if salary of Sami is 70,000? In such a case, we need to apply a multiplicative perturbation instead because perturbation of 2000 is insufficient to protect the record of Sami.

5. DISCUSSION

In this section, we compare between the two techniques, Query-Set Size and Fixed-Data Perturbation, in terms of security versus precision, the features of computer systems, consistency, processing cost, and attribute domain. We then summarize the comparison in a table.

5.1 Security vs. Precision

Any security technique is perfect if it achieves maximum precision and high security. Neither Query-Set Size nor Fixed-Data Perturbation is perfect solution because no one is the superior in both aspects; security vs. precision [1][8].This issue highlights the trade-off between the usability and preserving privacy. Particularly, adding small random perturbation to the query result would be not suitable in a hospital database for example [3][4][6]. On one hand, the hospital would like allow medical research that is based on the information in the database. On the other hand, the hospital is legally obliged to protect the privacy of its patients, i.e., leak no information regarding the medical condition of any specific patient that can be "traced back" to the individual [6].

5.2 Characteristics of Computer Systems

Security technique should be based on the fact that an SDB is a database in which interrelated data about different kinds of populations are included; this fact should be taken into account during the designphase of SDBs. As SDB designers, we should also consider the features of used computer systems (Offline vs. Online, Static vs. Dynamic etc) since these features affect the complexity of SDB security problems. Comparedto other perturbation techniques, Fixed-Data Perturbation is well suitable method for online, dynamic SDBs in such a way that users accessing only the perturbed database but not the original. All modifications affecting the original database can immediately be reflected into the perturbed values.

5.3 Consistency

Consistency represents the lack of contradictions and paradoxes [1].Contradiction arises when different responses are gotten to repetitions of the same query. (A difference in answers to repetitions of the same queries that is due to changes in the real world is not, however, considered an inconsistency.) As an example of paradox is a negative response to a count query. Consistency is a desirable feature of any security method. The drawback of Fixed-Data Perturbation the presence of paradoxical values.

5.4 Processing Cost

Processing cost [1] is the CPU time and storage requirements of the method during query processing. This cost is a significant factor in some SDBs like online, dynamic SDB. In the Fixed-Data Perturbation , there are two choices for the Database Administrator (DBA) to implement the perturbed data, either implement them once, at the time data are entered into the system, or generate these values every time they are accessed. In the first choice, the CPU time would below but the online storage would be high since the original SDB is also kept online, while in the second choice, additional storage requirement is avoided with high CPU time. Even though Query-Set Size technique does not require any more cost related to the perturbed data, it requires more time to check all cases to find trackers.

5.5 Attribute Domain

Query-Set Size technique can be applied to both types of attributes, numerical and categorical [1]. In the other hand, Fixed-Data Perturbation is applied to numerical attributes only.

5.6 Summary of the Comparison

Table 2 summarizes the advantages and disadvantages between the two techniques in terms of the above criteria.

Criterion	Query-Set Size	Fixed-Data Perturbation
Security	Low	High
Precision	High	Low
Computer System	Offline, Static Database	Online, Dynamic Database
Consistency	High	Low
Processing Cost	High because of checking all tracker cases	Case 1: if the implementation of the perturbed data is once: time is low and storage is high Case 2: if the implementation of the perturbed data is every time: time is high and storage is low
Attribute Type	Categorical &Numerical	Numerical

6. SUMMARY AND FUTURE WORK

A statistical database is one that contains sensitive data but it responds to queries with aggregate statistics. In this paper, the inference problem in SDBs is stated with some examples to illustrate the sensitivity in data. Two techniques to secure SDBs from the inference problemare reviewed. In particular, Query-Set Size and Fixed-Data Perturbation. Also, a detailed comparison between them is presented here. As a result, there seems to be no single general solution. Each technique can be applicable to different circumstances and needs; such circumstances are mentioned in the paper. Definition of data sensitivity needs more research because the current definitions are imprecise.

7. REFERENCES

[1] Adam N. R. and Wortmann J. C., K. 1989.Security-Control Methods for Statistical Databases: A Comparative Study.ACM Computing Surveys. (Dec. 1989), 515-556.

[2] Elmasri R. and Navathe S. B. 2011. Fundamentals of Database Systems, 6th Edition, Addison-Wesley

[3] Gollman D. 2006.Computer Security, 2nd Edition, Wiley & Sons Ltd.

[4] Simpson A., Power D., and Slaymaker M. 2006. On tracker attacks in health grids, SAC'06 – ACM (Apr. 2006), 209-216

[5] Pfleeger C. P. and Pfleeger S. L., 2006. Security in Computing, Prentice Hall, 4th Edition.

[6] Dwork C. and Yekhanin S. 2008. New Efficient Attacks on Statistical Disclosure. Advances in Cryptology-CRYPTO, (Aug. 2008), 469 – 480.

[7] Traub J. F., Yemini Y., and Wozniakowski H. 1984. The statistical security of a statistical database,"ACM Transactions on Database Systems(TODS),(Dec.1984), 672-679.

[8] Chawla S., Dwork C., McSherry F., Smith A, and Wee H. 2005. Towards privacy in public databases, in Proceedings of the 2nd International conference on Theory of Cryptography (TCC'05), 363-385.

[9] Knuth D. The Art of Computer Programming, 1997. Vol. 2, Addison-Wesley.