frequency is required. It is shown that it is very difficult and almost impossible to isolate any single record value pertaining to an individual.

To my wife Johanne.

ACKNOWLEDGEMENTS

I wish to express my gratitude to V.S. Alagar, who patiently guided and encouraged me during my stay at Concordia. His advice and knowledge have helped me initiate, pursue and complete this research. Professor Alagar also read and criticized many versions of this thesis. The final version owes much to him; the mistakes are my own.

To all my friends and colleagues in the Department of Computer Science, thank you for your help and encouragement. I would also like to thank Professor Alagar and the Department of Computer Science for their financial assistance.

Last but not least, thank you, Johanne, for your patience and understanding.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 IMPORTANCE OF CONFIDENTIALITY IN STATISTICAL DATA BASES.

All government statistical offices maintained and operated by countries around the world are governed by regulations of statistical acts of the respective countries. These acts typically include two fundamental provisions : on the one hand they empower the statistical offices to collect information from respondents and provide for penalties to respondents if they refuse to provide the requested information; on the other hand they require that the statistical offices do not disclose in any way the information provided by individual respondents. [10]

The need to forbid the disclosure of individual information is vital since without this assurance to the respondents, the statistical offices would find that their data sources would dry up rapidly. Statistical offices carefully scrutinize their publications to insure that there is no disclosure of information about individual respondents. This has never been an easy task. Yet, in the past the technical limitations of users and the amount of work needed for disclosure put a low ceiling on the amount of individual information obtainable from these publications. Under these

circumstances the problem of scrutinizing the tabulations, prior to their release, for potential disclosure was more manageable.

Since the advent of computers the problem of security of statistical data bases has become much more difficult to manage. Computers have now become such powerful tools in the hands of users and producers of data that the confidentiality problem has assumed numerous dimensions of interest and importance. Computers enable users to apply analytical techniques to the publications of statistical offices that would have been impossible in the past. They also provide statistical agencies, as well as users, with a tool for quickly processing, storing and retrieving information from a variety of separate files, possibly collected over long periods of time. Computers have caused a tremendous increase in the amount of information collected for statistical purposes. This increases geometrically the magnitude of the problem of checking tabulations for disclosure. This problem must be dealt with effectively not only because of the legal requirements but also because statistical offices must insure individual privacy in order to have public cooperation, without which reliable statistics can not be collected.

Privacy can be defined as "the right to determine what information about ourselves we will share with others" [10]. The concern about privacy, therefore, centers around the question of making such information available to others,

possibly unknown to the respondent, without his or her consent, thereby increasing the knowledge of others about him. The concern is real and the danger is also real. Since the information collected by the agencies is usually fully identifiable, it is capable of being used for purposes other than those for which it was collected and by agencies other than the collector. Therefore the possibility of combining information about the same individual, collected by different agencies, is high. Since more and more information about the same individual can be compiled by the same agencies, even without the individual's knowledge, there is a great need for strict regulations concerning the disclosure of such information.

The public benefits directly and indirectly from the collection and the legitimate uses of statistics by governments, businesses, nonprofit organisations, academic users, etc; yet the public is also concerned about the increasing burden of providing the required information and about the real possibility of the misuse of the data provided by them.

In section 1.2 we discuss "Computer based record keeping" together with some of the important government recommendations of safeguard requirements and in section 1.3 we point out some instances of the potential violations of data security.

1.2 COMPUTER BASED RECORD KEEPING SYSTEMS.

In this section we will discuss the different aspect of computer based record keeping along with the general government recommendations of safeguard requirements of such published systems. See [26] for an extensive description.

Since several organizations are requiring the use of computer based record keeping, there exists a tremendous need to collect, screen and process large volumes of information than ever before.

The cost of setting up an automated record keeping system is very high. So, the manager of a newly acquired system may have strong economic incentives to spread the initial cost over as large a data processing volume as he can. This leads to the collection of more information than is usually needed by the organization. Therefore, the first effect of computer based record keeping is that an organization is forced to collect and store large amounts of information about each individual.

The second effect of computerization on personal data record keeping is that it facilitates access to data within an organization and across boundaries that normally separate organizations [26]. This enables large organizations to form dossiers on individuals by collecting information from many different files. For an example we can look at the "National

Driver Register" in the U.S. which provides a central data facility containing the names of individuals whose driver licences are denied or withdrawn by a state. This enables a state to deny issuing a licence to an individual whose licence has been revoked by another state.

The third effect of automating record keeping is that the information contained in personal records is available to the technicians who operate the system. Therefore individual information is available to these new record keepers who need not and should not know this information.

There was a time when information about an individual tended to be elicited in face to face contacts involving personal trust. Nowadays an individual must increasingly give information about himself to faceless institutions, for handling by strangers. Sometimes the individual does not even know that an organization maintains a record about him, much less contest its accuracy, control its dissemination, or challenge its use by others.

We will now distinguish two categories of personal data systems: administrative systems and statistical reporting and research systems. The essential difference between the two categories is functional. The first category maintains information on individuals for the purpose of affecting them directly as individuals; while the second category maintains data about individuals exclusively for statistical reporting

or research, and is not intended to be used to affect any individual directly.

We define an "automated personal data system" as a collection of records containing personal data that can be associated with identifiable individuals, and that are stored in whole or in part, in computer accessible files. Data can be associated with identifiable individuals by means of some specific identifier, such as name, or because they include personal characteristics that make it possible to identify an individual with reasonable certainty. "Personal data" includes all data that describes anything about an individual.

We will now discuss general recommendations of safeguard requirements for administrative personal data systems which are originally put forward in [26].

Any organization of an administrative automated personal data system shall be required to identify one person who is responsible for the system. This will guarantee that there will be someone with authority to whom a dissatisfied data subject can go to clarify any information that is stored in his record. All employees that have any function related to the system must be aware of all the safeguard requirements and rules that govern the personal data system. The organization must take reasonable precautions to protect data in the system against unauthorized access to the data,

including theft or malicious destruction of data files.

These safeguards must also be upheld when transferring identifiable personal data to another system. The system should keep a record of access to and use made of any data in the system, including who accessed the data. This will allow an organization to detect improper dissemination of personal data. The organization must maintain the data in the system with such accuracy, completeness, timeliness and pertinence as is necessary to assure accuracy and fairness to the individual whose information is stored in the system.

Finally, an organization must eliminate data from computer accessible files when the data is no longer needed or useful to his immediate needs. This will assure that obsolete data is not available for routine use.

The following recommendations pertain to the rights of individuals about whom information is stored in the data bank.

Any organization maintaining an administrative automated personal data system shall inform an individual asked to supply personal data for the system whether he is legally required, or may refuse, to supply the data requested. The individual shall have the right to request from the system, the information, if any, that is kept in his record. The organization must assure the individual that no use of individually identifiable data is made that is not within the

stated purposes of the system as understood by the individual. The individual shall be informed, upon his request, about the uses made of data about him, including the identity of all persons and organizations that have made use of this data; and an organization must maintain procedures that will allow an individual to contest the accuracy and completeness of the data pertaining to him.

These are the main recommendations put foward in [26]. They will allow an individual to control the accuracy and completeness of the information that an organization has on him. These safeguards will also incite organizations to keep accurate and complete information, and to purge irrelevant and obsolete data.

The second category of personal data system is identical to the system discussed above, however the administration also uses the system for statistical reporting and research uses. These uses may be secondary to the system, yet the privacy of an individual might be compromised in the process of statistical reporting and therefore will need some additional safeguards for such system.

Since one advantage of automating administrative records is the capability thereby acquired for fast data retrieval and manipulation, a growing number of administrative data systems will be put to such additional uses. The personal data collected by organizations for administrative purposes

should be limited, ideally, to data that are demonstrably relevant to decision making about individuals, however since personal records can easily be used as a statistical data base it was found that a large amount of additional information could be easily and cheaply obtained on an application form, or some other record of an administrative transaction. It was found that decisions to collect personal data are being made without consideration of whether they will in fact serve the purposes for which they are being collected or because at some point someone thought they might be useful to have.

Most disturbing of all, it was found that personal data in excess of those clearly needed for making decisions about individuals are sometimes collected in a way that makes them seem prerequisite to the granting of rights, benefits, or opportunities [26].

There is also reason to believe that failure to separate the needed information from the information for statistical purposes may influence decisions made about individuals. For example, "Race" and "Sex" are no longer asked on many application forms because of their acknowledged influence on some types of decision making about individuals.

In [26], the authors recommend five principles to be followed when an organization uses its personal data system for statistical purposes.

They are

(1) When application forms or other means of collecting personal data are used, the mandatory or voluntary character of an individual's responses should be made clear.

(2) Personal data directly influencing an individual's right and personal data collected for statistical purposes should be stored separately.

(3) The amount of secondary information should be kept at a minimum.

(4) Proposals to use administrative records for statistical reporting and research should be subjected to careful scrutiny by persons of strong statistical and research competence.

(5) Any published findings and reports should meet the highest standards of error measurement and documentation.

These principles should be followed as closely as possible, and the authors of [26] recommend that the following two safeguard requirements be added to the ones mentioned above.

(1) Any report concerning administrative automated personal data system that is published should be made

available for independent analysis.

(2) Assure that no data made available for independent analysis will be used in a way that might cause injury to any individual data subject.

The next type of automated record keeping is the data base that is strictly used for statistical reporting and research. Included in this category are medical data banks used for research, census data banks and several others. We will discuss, in general, the usefulness and vulnerability of such systems, as well as the needed or necessary safeguards in order to assure the privacy of the individuals about whom information is stored in the data bank.

Data collected for statistical and reseach purposes are a must in present day society. For example, collecting data on patients having a certain disease can help researchers find cures; the government, through the census board, collects data on economic and social levels in different regions, therefore enabling it to administer its program with fairness to the regions. This causes a very large amount of information to be collected, not all of which identifies individual data subjects, but of those that do, the ones dealing with controversial social and political issues are particularly vulnerable to misuse in the absence of specific statutory requirements. Unless people get, and believe

assurances that the information they provide for statistical reporting and research will be held in strictest confidence and used in ways that will not result in harm to them as individuals, they will inevitably become either less willing or less reliable participants in surveys and experiments.

The recommendations of safeguard requirements mentioned for personal data systems would also apply to systems used exclusively for statistical reporting and research.

The following safeguard requirements should also be observed for statistical systems. These are

(1) Inform all employees, having anything to do with the system, about all the safeguard requirements and all the rules and procedures of the organization in order to assure compliance with them.

(2) Specify penalties to be applied to any employee who discloses any information, collected by the organization, that could harm any individual in the data base.

(3) Take reasonable precautions to protect data in the system from any anticipated threats to the security of the system.

(4) Make no transfer of individually identifiable personal data to another system without specifying and receiving assurances that the safeguard

requirements will be followed.

(5) Have the capacity to make fully documented data readily available for independent analysis.

These are the major safeguards recommended in [26] and the authors feel that in order to maintain the security of a statistical data base, these safeguards should be observed very closely.

1.3 POTENTIAL VIOLATIONS OF DATA SECURITY.

The tradionally best known type of potential violation of data security may occur through a breach of secrecy oath by an employee of a statistical office. Other security breaches may occur when an unauthorized person unlawfully makes a copy of a computer tape containing statistical information, or when copies of confidential data are mailed out inadvertently to the wrong respondent, or when confidential reports are misfiled in a library open to the public.

As long as people are involved in the processing of data there may be accidents, apart from the conscious breaches of security mentioned above. However, in this section, we are interested in security breaches that occur when users can infer information about an individual from information acquired from publications of statistical offices.

For example, information pertaining to fewer than three respondents would result in direct statistical disclosure since any one of the two respondents could subtract his own report from the published aggregate and would thus deduce the quantity reported by the other. Most statistical offices have guidelines to protect against such simple disclosure. However, it seems that advances in the theory of record linkage, together with the increasing capacity and power of modern computers, represent a new development compelling the statistical offices to reevaluate their approaches to disclosure checking. By this we mean that a user could have information pertaining to a respondent from previous publications, hence enabling him to infer information about that respondent by cross tabulating the results of the two publications. This is called residual disclosure.

In the next section, we will discuss various models of a statistical data base, along with the different query types. In addition to this we will show how information can be deduced from the responses obtained by querying the data base. The various restrictions imposed on a data base in order to protect them will also be studied.

1.4 DATA BASE MODELS AND DEFINITIONS.

A statistical data base system is comprised of records containing information about individuals. By individual we mean a single person, company, group or society. For example, a data base could contain records pertaining to companies involved in the exportation of raw materials, while another data base could have records pertaining to the individual workers employed by these companies.

A record is essentially a collection of information about a particular individual and has three main components: identification key, attributes and data field (more than one data field is possible).

The identification key is an identifier that is unique to each record such as names, social security number or any other identifier that can uniquely be associated to an individual. In some cases, a set or subset of the attributes can be considered as an indentification key. This occurs when the subset of attribute values is so specific that it refers to only one individual.

An attribute is a characteristic about an individual such as sex, age, marital status, etc. Each attribute can have two or more values. For example, the attribute sex can have the value "Male" or the value "Female". From this it follows that, for all individuals, each attribute in a record must

have one of the possible values. The last component of a record is the data field section. A data field (one or more for each record) of a record contains some quantitative information pertinent to the recorded individual. This might be the salary earned by the individual, the amount of charitable donations or his contributions to one or more political parties.

Associated with each data base is a querying system which allows users to get summaries of the information stored in the data base. These systems are, in most cases, of the "Question and Answer" type. By this we mean that the user asks a question (query) to the data base, which searches the records, computes an answer (response) and returns it to the user. There are many different query types, all of which will be discussed later in this section.

The security problem we are interested in, in this thesis, is mainly due to this type of querying. Since the system will not respond when only one "KEY" is specified, it is not possible to gather information on an individual by asking such simplistic queries. However a user of the system can ask a series of permissible queries and from the response to the queries try to infer any sensitive information on an individual. Recent research has shown that a fairly "intelligent" user can break the confidentiality constraints by constructing vulnerable yet permissible queries.

When such unauthorized information has been deduced from the responses to queries, the data base is said to be COMPROMISABLE. We distinguish between partial and full compromisability and V and E-compromisability and these definitions are as follows: a data base has been partially compromised, if a user can infer from the responses to valid queries any information, (either an unknown attribute or a value of a data field) previously unknown to him, about any individual whose record is in the data base. Full compromisability occurs when all the information in the data base can be inferred.

We say that a data base is (partially, fully) E-compromisable if by knowing the responses to queries we may deduce the existence of some (or all) records in the data base. Similarly we call a data base V-compromisable if the value of any field, previously unknown, can be deduced from the responses to allowable queries.

As mentioned earlier there are many different query types, but they can be categorized into two main groups: Key based queries and Attribute based queries.

Let us first look at key based querying systems. These querying systems are used in conjunction with data bases whose records must have the identifier field and data fields specified. Some attribute fields may be present, however they are not used by the querying system. In these data

bases there are N records containing information about N individuals. Each record is identified by its unique key and these keys are known (at least some of them), by the user. Since the existence of a record in the data base is known, E-compromisability is not considered in such a data base. The record associated with each key, will have one or more data fields that store information about the individual who is identified by the key. Therefore the existence of a record must be known by the user in order for him to be able to access it. This is done by specifying its key or identifier. A user will only have to specify on what data field he wants statistics and about whom, and the data base management system will return the appropriate response.

An example of a key based query is

$$\text{AV SALARY}(\text{Key}_1,\text{Key}_2,\ldots,\text{Key}_k).$$

Here the system will return the average salary of the data persons identified by the keys $\text{Key}_1,\text{Key}_2,\ldots,\text{Key}_k$. In most cases the size of the query (the number of keys allowed in a query) is fixed for all queries. If we were to let the size of the queries vary, then a user could compromise the data base very easely. For example, by asking the two queries $\text{SUM SALARY}(\text{Key}_1,\text{Key}_2,\text{Key}_3)$ and $\text{SUM SALARY}(\text{Key}_1,\text{Key}_2)$ a user can determine the salary of the individual associated with Key_3, by subtracting the response of the second query from the response of the first query. This is why in most cases, the size of the query must be fixed. As we shall see

later fixing the query size is necessary but not sufficient to protect confidentiality.

The second category of queries is attribute based queries. These queries are used with data bases that have records with attribute fields and data fields. The identifier field may be present but it is not used by the system when attribute based queries are used. An attribute (or characteristic) based query is a query that requests statistics about all the individuals in the data base that satisfy or match the attributes specified in the query. The attribute based query will have the form $Q(v_1,v_2,v_3,...)$ where the v_i's are the values specified for attribute$_i$. Any number of attribute values can be specified. For example the query Q(Sex=male) is a query that requests information about all the males in the data base; and the query Q(Sex=male,Status=citizen) request information about all the persons in the data base who are male and are citizens. The first query specifies the value of one attribute while the second query specifies a value for two attributes. For these queries there are two main statistics available: counts and data field summaries. The count queries return the number of records satisfying the query, while data field summaries are similar to the information returned to key based queries. In this type of data base we must consider both V and E-compromisabilities.

Considering queries that request information on the data fields, as mentioned above, there are many different types. Some queries ask the data base to return the sum of the data fields of the records satisfying the query, while others ask for the average, or the median value, or the largest, or the smallest.

EXAMPLE 1.1

Consider a database in which each record stores the information (NAME,SALARY,CONTRIBUTIONS) regarding one individual. Assume that key based queries alone are permitted.

IDENTIFIER	SALARY	DONATIONS
JOHN	$21K	$200.00
PAUL	18K	175.00
ANN	19K	50.00
JACK	32K	75.00
MARY	23K	250.00
LUCY	16K	25.00
PETER	25K	100.00
DAVID	19K	75.00

If the number of keys permitted per query is 4, a few examples of key based queries are:

QUERY	RESPONSE
SUM SALARY(JOHN,PAUL,JACK,LUCY) =	$87K
AVE DONATION(PETER,DAVID,MARY,ANN)=	118.75
MEDIAN SALARY(ANN,MARY,LUCY,PAUL) =	18K
AVE SALARY(JOHN,JACK,PETER,DAVID) =	24.25K

As we can see, a user must know of the existence of John,Paul,Ann,etc...,, in order to be able to ask these queries. By existence we mean that every user is supposed to know that the NAME field is part of every record in the data base and in addition should know the various names that occur in these fields across the various records. We will see a little later that such a data base is very easy to compromise if no restrictions are placed on the permissible queries.

EXAMPLE 1.2

Consider a data base in which each record stores the information (SEX, PARTY AFFILIATION, SALARY, POLITICAL CONTRIBUTION) pertaining to one individual.

	ATTRIBUTES		DATA FIELDS	
RECORD NUMBER	SEX	PARTY AFFILIATION	SALARY	POLITICAL CONTRIBUTION
N1	F	LIB	$16K	$280.00
N2	F	PC	18K	100.00
N3	M	PC	24K	200.00
N4	F	NDP	19K	300.00
N5	M	PC	21K	75.00
N6	F	LIB	20K	175.00
N7	M	PC	19K	225.00
N8	F	LIB	23K	250.00

In this data base model the identifier keys have been suppressed from the user's view. However some data bases using attribute based querying keep the identifier for purposes of updating the records, although no user will have access to this field. The data base in Example 1.2 is a

typical example of a data base that will permit attribute based queries. The data base has two attribute fields: SEX and PARTY AFFILIATION. The attribute SEX has two values, while the attribute PARTY AFFILIATION has three values. As we see the number of values each attribute can have is not fixed. This occurs in most (if not all) real data bases. The data base also has two data fields:SALARY and PARTY CONTRIBUTION.

The following are examples of attribute based queries on the data base in Example 1.2.

	QUERY	RESPONSE
1.	SUM SALARY(SEX=F)	$96K
2.	AVE POL.CONT.(SEX=M and PARTY=PC)	$166.67
3.	COUNT(SEX=M and PARTY=(LIB or PC))	3

The first query requests the total salary of all the data persons in the data base with SEX=Female. Therefore the response is the sum of the data field SALARY in the records N1,N2,N4,N6 and N8. The second query asks for the average political contribution of all data persons that have SEX=M and PARTY AFF.=PC. The response is the average value of the data field POL.CONT. in the records N3,N5 and N7. The third query asks the question " How many data persons in the data base have the attribute SEX=M and the attribute PARTY AFF.=LIB or PC? "

A count query returns the number of records that satisfy the CHARACTERISTIC FORMULA of the query. By characteristic formula we mean an arbitrary logical formula using attribute values as terms connected by the logical operators AND, OR and NOT. SEQUEL is an example of a query language permitting such formulas. For example consider the formula (MALE and (PC or LIB)). This refers to all MALES that are either PC or LIB.

In some cases a data field may be used as an attribute. For example, referring to Example 1.2 we can ask the following query: COUNT(MALE and SAL>20K) and the response to this query is 2. This query ask how many males have a salary greater than 20K.

In such a data base, a user need not know any information about the existence of particular records in order to query the data base.

As mentioned earlier, a statistical data base that permits key based queries is not very secure. It seems natural to impose restrictions on the type and number of key based queries that can be permitted in the system.

The first restriction must be the fixing of query size. This prevents disclosure of unauthorized information by simply subtracting one response from the other. However, this restriction does not prevent disclosure and as we will demonstrate below it is nevertheless quite simple to

compromise such a data base.

Using the data base in Example 1.1 and restricting the size of the queries to three, it can be shown that it is possible to set up a series of queries that will enable a user to infer information about individuals. This set of queries and responses is converted into a system of linear equations which is easily solved by any known technique.

Let us suppose that a user wishes to know the salary of Paul. Since k (the size of the query) is 3, the user knows he will need k+1=4 carefully chosen queries in order to set up a system that is solvable for the quantity he is after. By asking the 4 following queries

SUM SALARY(PAUL,ANN,JACK)

SUM SALARY(PAUL,ANN,JOHN)

SUM SALARY(PAUL,JACK,JOHN)

SUM SALARY(ANN,JACK,JOHN)

he will get the responses 69K, 58K, 71K and 72K respectively.

Setting X_1=Paul's salary

X_2=Ann's salary

X_3=Jack's salary

X_4=John's salary

the user can write down the query-response sequence in the form

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} 69K \\ 58K \\ 71K \\ 72K \end{bmatrix}$$

Since the determinant

$$\begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix} \neq 0$$

the system has a unique solution for the unknowns. Solving this system we see that Paul's salary is 18K, we also determine the salaries of Ann, Jack and John.

As illustrated in this example, just imposing a restriction on the number of keys per query will not prevent compromisability in the hands of a knowledgeable user. It seems that we will need some more sophisticated methods to prevent a total disclosure.

In the past many proposals have been put forward but most are either ineffective or too hard to implement. We briefly review many of these proposals below.

In [3,4,5,15] a threat monitoring scheme has been proposed which limits the number of overlapping keys in any sequence of queries. In other words we have a data base consisting of N records that allows queries about the sum of

any subset of size k. Then we add the further restriction that no two queries may overlap in more than r, $0<r<k$, positions. This seems almost impossible to implement because a user may query the data base over a long period of time or may share responses to queries with other users. Assuming that this restriction is possible to implement, security is still not assured as demonstrated by the following example.

Let k=3 and allow no more than 1 key in common between queries. We can infer the value of John's salary (in Example 1.1) by the following five queries:

	QUERY	RESPONSE
1.	SUM SAL(JOHN,PAUL,ANN)	58K
2.	SUM SAL(JOHN,JACK,MARY)	76K
3.	SUM SAL(JOHN,LUCY,PETER)	62K
4.	SUM SAL(PAUL,JACK,LUCY)	66K
5.	SUM SAL(ANN,MARY,PETER)	67K

Let R_i be the response to $query_i$. John's salary can be computed as follows.

$$\text{John's salary} = \frac{1}{3}(R_1+R_2+R_3-R_4-R_5)$$
$$= \frac{1}{3}(58+76+62-66-67)K=21K$$

We can see that security is not always assured when we restrict the amount of overlap. In chapter 2 we discuss in detail the results of [3,4,5,15] and show why in most cases

restricting the overlap between queries is not the answer to the security problem.

As remarked earlier, sum and average queries are very powerful. Instead of giving exact answers to the sum or average seeking queries, it may be desirable to introduce an element of uncertainty in the response. For example the user may be given the median value of the k records specified in the query instead of the exact answer. This seems promising since while the median does give the value corresponding to one person, it supplies no information about other members of the data base. However compromise is still possible, as the following theorem [2] shows.

THEOREM 1.1

"Every sufficiently large data base can be compromised by selection queries in no more than $M(k) \leq 4k^2$ queries."

The proof of this theorem is given in [2]. Thus, no large data base with selection queries is secure. By selection queries we mean queries of the form:

"What is "P" of the following list of k data persons?" Where P is any selection query predicate such as MAX, MIN,MEDIAN,etc...

In [13] the authors propose a security method where the query returns the weighted sum of the data elements to which it applies. The queries, in such a system, have the

following form:

$$Q(key_1, Key_2, \ldots, Key_k) = \sum_{i=1}^{k} a_i z_i$$

The z_i's are the data elements corresponding to the records identified by the Key_i's, the weights a_i are unknown and k is fixed for all queries. The authors show that if the value of one z_p is known then it may be possible to determine the values of the remaining data elements. In chapter 2 we show how this is possible and illustrate the method with some examples.

Another security proposal has been to distort the responses slightly while maintaining the integrity of the answers given. This distortion may be achieved by adding "noise" to all answers in a manner that does not vastly change their implication. In the most simple form of this type of lying, we allow only queries involving k individuals, restrict the overlap to 1 between queries and return results at random. By appealing to results from matching theory, we can show that k^2-1 queries can be successfully answered without compromising the data base [3]. However, using a simple strategy based on geometry, k^2 well chosen queries generally suffice to compromise. While this number is possibly large enough to discourage all but the most malicious of users, it is obtained for a model more restrictive than realistic; "real" systems will surely be more vulnerable.

Let us now look at security proposals for data bases using attribute based queries. To illustrate the complexity of the problem let us first look at a simple example.

Suppose we wish to determine whether Mr. X earns $30,000 per year or more and we know that his data is stored in a statistical data bank. Suppose also that we know that Mr. X is a 28 year old teacher with a B.Sc. Degree, has 3 children and lives in Montreal. When we ask the data base the query

"How many data persons in the data base have the following properties:

Age: 28

Education level: B.Sc.

Sex: male

Number of children: 3

profession: teacher

City: Montreal"

Assume we get the response "14 people". If we ask the same query but add the characteristic "earns more than $30,000 a year" to the list of characteristics and the data base returns the response "14 people" again. With these two queries we know that Mr. X's salary exceeds $30,000 a year. We have therefore inferred unauthorized information from the data base even though the responses to our queries are only counts. This shows that even if specific identifying information such as names are not stored in the data base, a

user can, using only count queries, compromise the data base. Therefore, queries involving counts must also be considered in any security proposal.

The probability of compromising increases as the counts get smaller; since the probability that all 14 male teachers of age 28 with a B.Sc. having 3 children and living in Montreal have a salary in excess of $30,000 is small. However if we can isolate an individual then we can deduce any characteristic of this individual by simply adding more attributes, one at a time, to our query. For example, if the response to the above query is "1", we ask the query:

How many people are there having the following characteristics?

Age: 28

Education level: B.Sc.

Number of children: 3

City: Montreal

Profession: Teacher

Is an alcoholic

If the answer to this query is "1" then Mr. X is an alcoholic; if the answer is "0" then he is not. This shows that when we can isolate an individual it is always possible to compromise either by inferring that Mr. X has a certain characteristic or not.

A security method to protect a data base against such a compromise has been proposed in [14] where queries involving small counts are not answered. Thus a restricted m-response is a response such that

$$\text{COUNT}(Q)=\begin{cases} X & X\geq m \\ \text{Undefined} & X<m \end{cases}$$

$$\text{SUM}(Q)=\begin{cases} Y & X\geq m \\ \text{Undefined} & X<m \end{cases}$$

Where Y is the true sum.

Even if responses to small counts are undefined it is still possible to compromise the data base. Schlorer showed that compromise may be possible even for large values of m using a technique called "trackers" [24].

As we will see in chapter 2, trackers are very powerful tools that a user can easily use to compromise a data base. In [6,16,23,24], it has been shown that in every data base trackers can be found and that the amount of work needed to find a tracker is small. To illustrate the concept of trackers, suppose m=3 for the data base in Example 1.2; i.e. no responses are given to queries which involve fewer than 3 or more than 5 individuals. Suppose we know that Jane Doe is a member of the PC party, and we wish to determine her salary.

We attempt to do this by asking the query SAL(F and PC) which has an undefined response. The reason why the response

is undefined is that it involves fewer than m=3 records. A user must first know whether the characteristic formula (F and PC) uniquely identifies Jane Doe or not. This is determined by the query COUNT(F and PC) for which the response is undefined.

If the response to this query was to be given exactly (=1), then the user knows that any information obtained using the formula (F and PC) pertains to Jane Doe. A tracker is a tool by which a user can obtain answers to queries for which the responses are not given by the system. For example when COUNT(F and PC)<m=3 (the threshold set by the system), a user can ask two carefully chosen queries for which the responses are obtainable in the system and from such responses the true answer of COUNT(F and PC) can be computed. This is illustrated below:

QUERY	RESPONSE
COUNT(F)	5
COUNT(F and not(PC))	4

Now

COUNT(F and PC)=COUNT(F)-COUNT(F and not(PC))

=1

The formula (F and not(PC)) is known as the tracker in this case. A user now knows that the formula (F and PC) uniquely identifies Jane Doe, and by the same approach may determine her salary as demonstrated below:

SUM SAL(F and PC)=SUM SAL(F)-SUM SAL(F and not(PC))

= 96K - 78K

= 18K

In chapter 2 we will review some of the recent research in this direction, i.e. finding trackers. It will be shown that for any statistical data base system, trackers can be found quite easily and hence it is very difficult to guarantee security under user inference even when severe restrictions are placed on queries of small or large counts.

All the proposals mentioned above demonstrate how difficult it is to secure statistical data bases. All these results, although quite interesting and important, are purely negative results in the sense that they do not contribute a positive solution to the data base security problem.

The methods that we discuss in this thesis contribute a positive solution to the data base security problem. A willful and malicious user can always gather information on an individual from sources external to the data base and may spend any amount of resources in trying to infer more about an individual from the system. It is highly possible that another computer systematically generates and modifies queries which are input to the data base in an effort to overpower the system. Keeping this in mind our results are to be considered positive only with caution.

An outline of the thesis and results are as follows:

In chapter 2 we discuss the existence of trackers and their efficiency. We will also review, in detail, the recent results obtained in [1,3,4,6,8,9,12,13,14,15,16,17,22,23,24] where the authors propose different security mechanisms and study their efficiency.

Chapters 3 and 4 contain results of our investigation. In chapter 3 we consider a data base that allows key based queries and give a method that protects the data base. The main idea here is to formulate the problem in terms of a linear algebraic system and give methods by which 1) the system of equations can be made an "indeterminate" system and hence unsolvable; 2) errors can be introduced in the responses so that by prefixing the error in the responses it is possible to control the error in the inferred values. Empirical evidences suggest that for a "reasonably small" data base (probably only key based querying is realistic in such a system) we have an effective way of securing the data base.

In chapter 4 we concentrate our efforts on cancelling the threat of trackers. To do this, the querying system returns a "fixed interval" for count queries instead of true counts. Since trackers need true counts in order to compromise a data base, it will be shown in chapter 4 that such an approach effectively nullifies the threat of trackers along with other

possible threats to the data base.

Finally chapter 5 summarizes the main results of the thesis.

CHAPTER 2

RECENT STUDIES OF INFERRENCE CONTROL MECHANISMS

In this chapter we will discuss in detail the recent research done in the area of security of statistical data bases. We will show the advantages and disadvantages of each of the proposals made and indicate why they can not adequately secure a data base. As we mentioned in the last chapter, most of the research in this area has been to study methods that systematically aim at breaking the security rather than methods to effectively safeguard a statistical data base. Despite the negative tone, this research is valuable because the nature of the threat must first be understood before effective counter-measures can be built.

Compromise occurs when a user infers, from the responses of one or more authorized queries, confidential information of which he was neither previously aware of nor supposed to know. Most of the attacks are based on isolating a single data element by intersecting several query sets. By query set we mean the set of records that satisfies a query. To defend against any unauthorized intruder's attack we outline four broad categories of actions:

1. Set controls on the sizes of query sets.
2. Set controls on the overlaps of query sets.

3. Distort the data or the query response.

4. Give response by sampling from the data base.

In the next few sections we will discuss the design, implementation and effectiveness of these methods.

2.1 CONTROLS ON THE SIZE OF QUERY SETS.

Let us now give a more rigourous definition of QUERY SET SIZE. A query set S_i is the set of records that is referred to by query Q_i. In the case of the key based query Q(John,Ann,Mary) the query set is the records associated with the identifiers John, Ann and Mary. The query set of an attribute based query is the set of records that have the same attribute values as those specified in the query. Therefore, query set size means the number of records in the query set.

We shall first concern ourselves with data bases that permit key based queries and then discuss controls in data bases that allow only attribute based queries.

In data bases that permit key based queries it is very important to control and set a limit on the size of query sets. Otherwise it is fairly easy to compromise the data base. For example, a user simply asks the query: AVE SAL(John). This would be a valid query since the size of the query set is not restricted. If there are no

restrictions, the system will simply give the true average salary, which in this case is the true salary of John.

This example suggests that it is necessary to put restrictions on the number of keys specified in the query in order to protect against full disclosure. However as we shall see in section 2.2, such a restriction is not sufficient to protect the data in the system.

For now let us discuss controls on the size of query sets in a data base using attribute based queries.

In [14] Chin studies a specific kind of data base and query model on which many restrictions are imposed. The data base model is defined as follows. Every record in the data base contains a key of k bits which uniquely identifies it. In other words this model uses attribute based queries, but where every record is uniquely identified by its characteristics. The key is made up of k bits, one bit per attribute.

Below is an example of a data base under this model.

EXAMPLE 2.1

RECORD	INCOME	SEX	MARITAL STATUS	ATTENDANCE	(DATA FIELD) CONTRIBUTION
R1	1	1	1	0	$200.00
R2	0	1	0	1	500.00
R3	0	0	0	1	300.00
R4	1	0	0	1	400.00
R5	1	0	0	0	25.00
R6	0	1	0	0	75.00

The (0,1) values in the attribute columns are interpreted as follows:

INCOME : 1= > 20,000 ; 0= $\leq$ 20,000

SEX : 1= male ; 0= female

MARITAL STATUS: 1= single ; 0= married

ATTENDANCE : 1= present ; 0= absent

Note that each attribute has one of two values, and each set of attribute values are distinct from each other. Therefore, the key (0101) identifies the record R2 uniquely.

The querying system for this data base model is quite simple. An s-query is a sequence of 0's, 1's and *'s of length k with exactly s 0's and 1's. A key (or record) matches a query if the key and the query agree in every position in which the query does not have a *. For example, the query Q(1*10) matches the keys (1010) and (1110). Two types of queries are permitted and are distinguished by the way the responses are made.

$S(Q_i)$: is the sum of the data fields of the records whose keys match query Q_i.

$C(Q_i)$: is the number of keys matching Q_i.

An important design decision of this model is that any query Q_i (either partially or fully specified) will not be answered if $C(Q_i)<2$.

More precisely the response to a query is defined as:

$$C(Q_i) = \begin{cases} X & X \geq 2 \\ \text{undefined} & X < 2 \end{cases}$$

$$S(Q_i) = \begin{cases} Y & X \geq 2 \\ \text{undefined} & X < 2 \end{cases}$$

where X is the number of records matching the query Q_i, and Y is the sum of the data fields of those records.

An intruder who is armed with some piece of information and tries to obtain much more sensitive information from the data base should first know the existence (or non-existence) of the information regarding the individual(s) he is after. If by asking a sequence of queries (count queries) for which the responses are defined (or undefined) the user deduces the existence (or non-existence) of an individual's record in the data base, we say that the data base is E-compromisable. The next step is to infer the data value associated with a record whose existence has been inferred. If such an inference is made through a sequence of queries (sum or average seeking queries) then we say that the data base is V-compromisable.

Certainly there exist several levels of E-compromisability. If we infer the non-existence of a record, we say that there is negative E-compromise although the user "gains" some information. If during the process of inference, the user E-compromises a record but he does not

know for sure ALL the attributes of that record we say that there is partial E-compromise. If every record in the data base is fully identified through inferential mode we say that the data base is strongly E-compromisable. When one or more records (but not all) are either fully or partially compromised we say that the data base is weakly E-compromisable. It is important to prevent compromisability at the weakest level.

The data base model proposed in [14] is very restricted and hence very sensitive as the following results show.

THEOREM 2.1

If the existence of one record is known, then the existence of all other records can be inferred.

Proof: Suppose we know the existence of record I, which has a key of k bits (a sequence of 1's and 0's of length k). Let the record J have a key that differs from I's key in only one position, then there exists a (k-1)-query which is only matched by I and J, namely the query which agrees with I and J except in the position where I and J differs; the query has a * in that position. Thus the record J exists if and only if the response to this query is defined. If the response to this query is not defined then the record J is not present in the data base. For a general and rigourous proof refer to [14].

This result shows that any weak E-compromisability implies strong E-compromisability in this data base model. We feel that such a model is totally unrealistic.

The next result is similar to Theorem 2.1 but deals with V-compromisability.

THEOREM 2.2

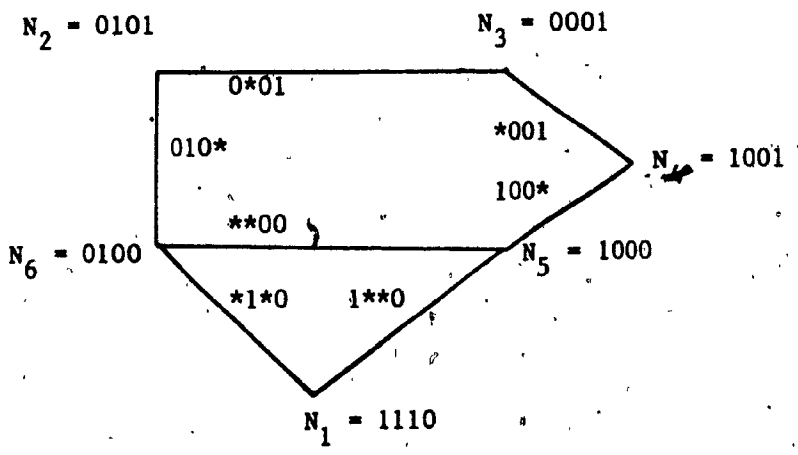
If the value (data value) associated with an existing record is known, then the value associated with every record in the data base can be inferred.

The proof of this theorem follows directly from the proof of Theorem 2.1, for a complete proof refer to [14].

The next theorem gives necessary and sufficient conditions for EV-compromisability.

We first define a graph associated with the data base called QUERY GRAPH. By this we mean an undirected graph G=(V,E), where V, the set of vertices, is the set of keys of the existing records and there is an edge (i,j) in E (the set of edges) if and only if there exists a query Q such that i and j are the only records satisfying Q. Figure 2.1 represents the query graph for the data base in Example 2.1.

FIGURE 2.1



THEOREM 2.3

If the existence of $record_i$ is known for some i, then the data base is EV-compromisable if and only if either

(a) the query graph for the data base has at least one odd cycle, or

(b) there exists a query Q such that COUNT(Q) is odd and at least 3.

Proof: We briefly summarize the proof as given in [14].

The proof of part (a) is straight forward. We know the existence of one record and using Theorem 2.1 we can then infer the existence or non-existence of all the other records. From the definition of the query graph and the assumption that there is at least one cycle of odd length, it follows that we have a system of N equations, where N is odd. Let the system be as follows:

$$\begin{aligned}
\text{Value}(\text{Key}_1)+\text{Value}(\text{Key}_2)&=\text{SUM}(Q_1)\\
\text{Value}(\text{Key}_2)+\text{Value}(\text{Key}_3)&=\text{SUM}(Q_2)\\
&\vdots\\
\text{Value}(\text{Key}_N)+\text{Value}(\text{Key}_1)&=\text{SUM}(Q_N)
\end{aligned}$$

where N is the size of the cycle.

Since N is odd and is at least 3, the system of equations has the determinant

$$\text{DET}\begin{vmatrix} 1 & 1 & 0 & \dots & 0 & 0 & 0\\ 0 & 1 & 1 & \dots & 0 & 0 & 0\\ & & & \ddots & & & \\ 0 & 0 & 0 & \dots & 0 & 1 & 1\\ 1 & 0 & 0 & \dots & 0 & 0 & 1 \end{vmatrix} = 2$$

Therefore there exists a unique solution to this system, i.e. V-compromisability is established.

To illustrate this, consider the following three queries on the data base in Example 2.1.

SUM CONT.(**00)=VALUE(R6)+VALUE(R5)= $100.00

SUM CONT.(1**0)=VALUE(R5)+VALUE(R1)= 225.00

SUM CONT.(*1*0)=VALUE(R6)+VALUE(R1)= 275.00

Rearranging in matrix form we get:

$$\begin{bmatrix} 1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} V(6)\\ V(5)\\ V(1) \end{bmatrix}=\begin{bmatrix} 100.00\\ 225.00\\ 275.00 \end{bmatrix}$$

where V(1),V(5) and V(6) represent the values of the contribution field in records R1,R5 and R6 respectively. Solving this system of equations we see that V(1)=200.00, V(5)=25.00 and V(6)=75.00, which are the exact values of the

data field of these records.

The proof of part (b) is also very simple. Since COUNT(Q)$\geq$3 and odd for some Q, then there exists one position in Q, say i, at which two of them differ. Thus position i in Q must be a "*". Define Q' to be the same as Q except in position i where it has a "1", and Q'' to have a "0" in that position. Thus, Q' or Q'' must be defined. Therefore the value of a record (the existence of each record is known) can be deduced by SUM(Q)-SUM(Q') or SUM(Q)-SUM(Q'') when N=3. Hence the data base is compromisable.

Again using the model in Example 2.1 we have the following instances of compromisability.

EXAMPLE 2.2

Suppose that a user knows an individual with the characteristic key (0001) and wishes to determine the amount of her contribution. The following three queries will suffice to determine this amount.

Q =SUM CONT.(*00*)=V(R3)+V(R4)+V(R5)= $725.00

Q''=SUM CONT.(000*)= UNDEFINED

Q'=SUM CONT.(100*)=V(R4)+V(R5) = 425.00

Therefore the amount of contribution of record R3 is $Q-Q'$=$725.00-425.00=$300.00.

EXAMPLE 2.3

Let Q=SUM CONT.(**0*) be a query on the data base in Example 2.1. This query is matched by 5 records and the response is $1300.00. Here N (the size of the cycle) is > 3 and odd. Two cases can occur here. Either Q' or Q'' is defined. If this is the case then some value can be inferred by either $Q-Q'$ or $Q-Q''$, as in Example 2.2. The other case occurs when both Q' and Q'' is defined. When this happens, one of Q' or Q'' is odd and at least 3. Therefore, some value can be inferred using the same process as in Example 2.2.

We mention two related results and the reader is referred to [14] for their proofs.

THEOREM 2.4

If Key_i and Key_j differ in ℓ positions and the existence of the record with Key_i is known then the record with Key_j can be inferred in at most $2^{\ell}-1$ queries.

THEOREM 2.5

If Key_i and Key_j differ in ℓ positions and the existence of the record with Key_i is known, then on the average no more than $(1+p^{-1})^{\log_2 \ell}$ queries will be sufficient to determine the existence of the record with Key_j, where p is the density of the data base, i.e. the ratio between the number of existing records and the number of possible keys (2^k).

We just comment here that all these results, although having an interesting mathematical flavour, are either trivial or unreasonable from the point of view of the underlying model. Moreover, it can be shown that a data base in this model is EV-compromisable even when the existence of no records is known. Consider the following case.

A user wishes to know if R1 is in the data base and if so what is his political contribution. Assume that the user knows that R1 is a single male; therefore he can ask the query COUNT(*11*) for which the response is undefined. Using "trackers" he can ask the following two queries:

COUNT(*1**)=3

COUNT(*10*)=2

With these two responses he can infer that COUNT(*11*) is 1 by

COUNT(*11*)=COUNT(*1**)-COUNT(*10*)

= 3 - 2

= 1

Therefore the user knows that R1's record is in the data base. Now by asking the next two queries:

SUM CONT.(*1**)= $775.00

SUM CONT.(*10*)= 575.00

he can determine R1's contribution by

SUM CONT.(*11*)=SUM CONT.(*1**)-SUM CONT.(*10*)

= $775.00 - 575.00

= $200.00

As this example shows, EV-compromisability is achieved without the pre-knowledge of the existence of any records which is a stronger result than the ones mentioned in [14].

With m=2, where m is the response level (if there are fewer than m records matching the query Q, then Q is undefined), the results found are trivial and the data base is easily compromised. The following is an attempt to extend the known results to the case m=3.

Let us first define m,N,k. As seen above, m is the response level; k is the number of bits in the keys and N is the number of records in the data base. Since m=3 note that every query must have at least 2 "*"'s because a query with only one "*" can, at most have 2 records satisfying it.

It is known that by using trackers most data bases can be compromised. What we will attempt to show here is "how many data bases can be compromised" and "how many can not be compromised". By "how many" we mean the number of possible data bases that exist for a fixed N and k.

For example consider the case N=7 and k=3. We have 8 possible distinct data bases; i.e. there are 8 different data bases that have 7 records and keys of length 3.

A data base is considered compromisable if and only if it is the only data base that gives responses $(X_1,X_2,\ldots,X_t)$ to all possible t queries; i.e. the set of responses to a set of

queries is unique to that data base.

EXAMPLE 2.4

Consider the following two data bases:

DATA BASE 1

NAME	AT1	AT2	AT3
N1	0	0	0
N2	0	0	1
N3	0	1	0
N6	1	0	1
N7	1	1	0
N8	1	1	1

DATA BASE 2

NAME	AT1	AT2	AT3
N1	0	0	0
N2	0	0	1
N4	0	1	1
N5	1	0	0
N7	1	1	0
N8	1	1	1

Here k=3 and N=6 for both data bases and the response level is m=3. Therefore there are only 7 allowable queries. These are

Q(***)

Q(1**) Q(0**)

Q(*1*) Q(*0*)

Q(**1) Q(**0)

The response to the query Q(***) is 6 for both data bases. The 6 other queries have response=3 for both data base 1 and data base 2. Therefore it is impossible to determine the exact occurrences of records, i.e. there is no E-compromisability either weak or strong.

Trackers here are also useless, because we can not determine which records exist and which ones are not in the data base. Even if we know the existence of one record (even

up to 4) we can not determine which of the remaining keys do exist.

Using this approach, and with the aid of the computer, we designed a program that finds the responses to all queries of all the data bases for a fixed N and k when m=3. The program then compares the responses. If the set of responses is found to be unique then there is only one data base that corresponds to these responses and thus we say that this data base is E-compromisable. If the set of responses is not unique then the data bases that correspond to these responses are not compromisable because a user can not deduce which data base he is dealing with.

The results of the program for k=3 and k=4 are given below:

TABLE 2.1 (k=3)

N	NUMBER OF DATA BASES COMPROMISABLE	NUMBER OF DATA BASES NOT COMPROMISABLE
8	1	0
7	8	0
6	12	16
5	24	32
4	14	56
3	0	56
2	0	28
1	0	8

TABLE 2.2 (k=4)

N	NUMBER OF DATA BASES COMPROMISABLE	NUMBER OF DATA BASES NOT COMPROMISABLE
16	1	0
15	16	0
14	120	0
13	560	0
12	1780	40
11	4048	320
10	7128	880
9	9920	1520
8	10940	1930

As we can see most of the data bases are compromisable. When k=3 the number of records ($\leq 2^k$) is small, therefore we do not have much to work with, this is the reason why the number of data bases that are not compromisable is relatively high. But as k increases, the number of records increases rapidly and the ratio of the number of data bases that are compromisable to those that are not is high.

From these results we can conclude that restricting the response level in such a data base model does not help very much. We also found one significant result which we state as:

THEOREM 2.6

If $N \geq \frac{3}{4}(2^k)+1$ then the data base is always compromisable when the response level m=3.

Proof: When k=3 we showed, in TABLE 2.1, that any data base that has $N \geq 7$ records can be totally E-compromised. Let

$C = \text{COUNT}(*** \ \ldots \ \ ***) = N$

$C_i = \text{COUNT}(**\ldots *1* \ldots **)$

$C_i' = \text{COUNT}(**\ldots *0* \ldots *)$

Note that $C = C_i + C_i'$, $i=1,k$, therefore when $C \geq \frac{3}{4}(2^k)+1$, either C_i or C_i', $i=1,k$, is greater than or equal to $\frac{3}{4}(2^{k-1})+1$. By induction on k=3, we can determine the existence or non existence of the records that have either a "1" (if $C_i \geq \frac{3}{4}(2^{k-1})+1$), or a "0" (if $C_i' \geq \frac{3}{4}(2^{k-1}+1)$, in position i, $i=1,k$. This will allow us to determine the existence or non existence of 2^k-1 records. To determine the existence of the 2^k-th record we simply subtract the number of known existing records from N. If this is 0 then the 2^k-th record does not exist; if this is 1 then the 2^k-th does exist. This completes the proof.

In [17] Kam and Ullman use the same data base model and add one more restriction to it; namely that every possible key is in the data base. By this they mean that if a key has k binary attributes then there exist 2^k records in the data base and each record is uniquely identified by its key. The results under these conditions are trivial. The assumptions and restrictions imposed on the data base models in [14] and [17] are very stringent and make these data base models very impractical.

Even in a more realistic data base, controlling the query set sizes alone does not defend against compromisability. In [6,18,23,24] it is assumed that a query is not answered unless $k \leq \text{COUNT}(Q) \leq N-k$ where COUNT(Q) is the number of records satisfying the query Q and k is the response level. Unfortunately, this restriction is often easily subverted (even for k near N/2) by a simple snooping tool called the "TRACKER" first introduced in [24]. In the following discussion on trackers we will be referring to the data base in Example 1.2.

In chapter 1 we defined a characteristic formula to be a logical formula using attribute values as terms connected by the logical operators AND,OR and NOT. Let C be a characteristic formula and X_C be the query set; i.e. the set of records whose attributes match those in the formula C. This data base model has attribute fields and data fields, therefore there are two main query types: COUNT(C) which returns the size of the query set, $|X_C|$; and SUM(C) which returns the sum of the values in the data field for records in X_C. The strategy of the querying system is as follows.

$$\text{COUNT}(C) = \begin{cases} |X_C| & k \leq |X_C| \leq N-k \\ \text{undefined} & \text{otherwise} \end{cases}.$$

$$\text{SUM}(C) = \begin{cases} \sum_{i \in X_C} V_i & k \leq |X_C| \leq N-k \\ \text{undefined} & \text{otherwise} \end{cases}$$

Schlorer [24] showed that if a query has a characteristic formula C for which COUNT(C) is not defined, it is possible to calculate COUNT(C) from two answerable queries involving the parts of C.

Suppose that a formula C believed to identify an individual I can be decomposed into the product $C=A \cdot B$, such that COUNT(A) and $\text{COUNT}(A \cdot \overline{B})$ are both answerable; i.e. $k \leq \text{COUNT}(A \cdot \overline{B})$ and $\text{COUNT}(A) \leq N-k$. The formula $T=A \cdot \overline{B}$ is called the "INDIVIDUAL TRACKER" of I. COUNT(C) can then be calculated using the following equation:

$$\text{COUNT}(C)=\text{COUNT}(A)-\text{COUNT}(T) \qquad (2.1)$$

If COUNT(C)=1 then the value of the data field can be computed from

$$\text{SUM}(C)=\text{SUM}(A)-\text{SUM}(T) \qquad (2.2)$$

For example, let us refer to Example 1.2 and set k=3. Suppose a user wishes to know the political contribution of Jane Doe. He suspects that Jane Doe is in the data base and knows that she is a member of the PC party. Asking the query COUNT(F•PC) and getting an undefined response he attempts to use trackers to compute the response.

The characteristic formula $C=F \cdot PC$ is decomposed into $C=A \cdot B$ where $A=F$ and $B=PC$. The formula $T=A \cdot \overline{B}$ is then $T=F \cdot \overline{PC}$. To verify that Jane Doe is the only individual characterized by C, the user applies equation (2.1)

$$\begin{aligned} COUNT(F \cdot PC) &= COUNT(F) - COUNT(F \cdot \overline{PC}) \\ &= 5 - 4 \\ &= 1 \end{aligned}$$

To determine her political contribution the user applies equation (2.2) on the data field POLITICAL CONTRIBUTION.

$$\begin{aligned} SUM(F \cdot PC) &= SUM(F) - SUM(F \cdot \overline{PC}) \\ &= \$1105. - 1005. \\ &= \$100. \end{aligned}$$

The user now seeks to know if individual I has characteristic "a" or not. Since $COUNT(C \cdot a) \leq COUNT(C) = 1 < k$, the user can not determine $COUNT(C \cdot a)$ directly. However with one additional query it can be computed as follows:

$$COUNT(C \cdot a) = COUNT(T + A \cdot a) - COUNT(T) \qquad (2.3)$$

If $COUNT(C \cdot a) = 0$ then I does not have characteristic "a" (negative compromise); if $COUNT(C \cdot a) = 1$ then I does have characteristic "a" (positive compromise).

In another example, using Example 1.2 again, let a user seek to know if Jane Doe's salary is greater than 16K. Applying equation (2.3) we get

$$\text{COUNT}(C \cdot a) = \text{COUNT}(T+A \cdot a) - \text{COUNT}(T)$$

$$\text{COUNT}(F \cdot PC \cdot SAL>16K) = \text{COUNT}(F \cdot \overline{PC} + F \cdot SAL>16K) - \text{COUNT}(F \cdot \overline{PC})$$

$$= 5 - 4$$

$$= 1$$

Therefore the user deduces that Doe's salary is greater than 16K.

In equation (2.3) we must show that COUNT(T+A•a) is such that the query has a well defined response. This is always true and the proof can be seen in [6].

The "INDIVIDUAL TRACKER" is based on the concept of using categories known to describe a certain individual to determine other information about that individual. A new tracker must be found for each person. The "GENERAL TRACKER" removes this restriction. It employs a single formula that works for the entire data base. No prior knowledge about anyone in the data base is required.

A general tracker is any characteristic formula T whose query set size is in the restricted sub-range (2k,N-2k); that is

$$2k \leq \text{COUNT}(T) \leq N-2k$$

Notice that a query with the characteristic formula T is always a legal query since its query set size is well within the range (k,N-k). Obviously k must not exceed N/4 if a

general tracker is to exist at all; in the worst case, k=N/4, T is a general tracker if and only if COUNT(T)=N/2. By symmetry, T is a general tracker if and only if $\overline{T}$ is a tracker. For example T=PC is a general tracker for the data base model in Example 1.2 when k=2.

The method of compromise, using general trakers, is described below.

The value of any undefined query can be computed as follows using any general tracker T.

If COUNT(C)<k use:

$$\text{COUNT}(C)=\text{COUNT}(C+T)+\text{COUNT}(C+\overline{T})-N \qquad (2.4)$$

If COUNT(C)>N-k use:

$$\text{COUNT}(C)=2N-\text{COUNT}(\overline{C}+T)-\text{COUNT}(\overline{C}+\overline{T}) \qquad (2.5)$$

Where $N=\text{COUNT}(T)+\text{COUNT}(\overline{T})$.

Because at least one of equation (2.4) or equation (2.5) is calculable, COUNT(C) can be evaluated with at most 4 queries beyond the 2 required to find N. This is proved in [6].

Referring to the data base in Example 1.2, a user seeks to know the political contribution of all the female PC party members. Since SUM CONT.(F•PC) is not defined, the user will use general trackers to compute the required response.

T=PC is a general tracker for the data base, and since COUNT($\overline{C}$+T)>N-k, k=2, the user must use the formula

$$COUNT(C)=COUNT(C+T)+COUNT(C+\overline{T})-N$$

The user first determines the size of the data base (N) and the total amount of all political contributions. This is done as follows

$$N=COUNT(\ T)+COUNT(\ \overline{T})$$
$$=COUNT(PC)+COUNT(\overline{PC})$$
$$=\quad 4\quad +\quad 4$$
$$=8$$

$$S=SUM(PC)+SUM(\overline{PC})$$
$$=\$600.00+1005.00$$
$$=\$1605.00$$

The user verifies that Jane Doe is the only female PC member by applying equation (2.4)

$$COUNT(F\cdot PC)=COUNT(F\cdot PC+PC)+COUNT(F\cdot PC+\overline{PC})-N$$
$$=\quad 4\quad +\quad 5\quad -8$$
$$=9-8$$
$$=1$$

He then calculates her contribution by applying equation (2.4) with summing queries as shown below

$$SUM(F\cdot PC)=SUM(F\cdot PC+PC)+SUM(F\cdot PC+\overline{PC})-S$$
$$=\$600.00\quad +\quad 1105.00\ -1605.00$$
$$=\$100.00$$

The compromise using general trackers is clearly a powerful technique that renders the attempt to secure a statistical data base by limiting the size of the query sets useless. In [6] it is shown that almost all data bases have a general tracker and in [23] the authors show how easy it is to find one.

When k>N/4, that is when more than half the range of query set sizes is disallowed, the general tracker is not guaranteed to work. Even if the data base could be proved secure from individual and general trackers when k>N/4, it may be susceptible to compromise by constructing "DOUBLE TRACKERS".

A double tracker is a pair of characteristic formulas (T,U) for which:

$$X_T \text{ is a subset of } X_U$$
$$k \leq \text{COUNT}(T) \leq N-2k$$
$$2k \leq \text{COUNT}(U) \leq N-k$$

Obviously $k \leq N/3$ if these conditions are to be met; in the worst case, $k=N/3$, $\text{COUNT}(T)=N/3$ and $\text{COUNT}(U)=2N/3$. Let the formula $A=C \cdot T$ and $B=\overline{C} \cdot T$. The method of compromise is as follows:

If COUNT(C)<k then use

$$\text{COUNT}(C)=\text{COUNT}(U)+\text{COUNT}(C+T)-\text{COUNT}(T)-\text{COUNT}(\overline{A} \cdot U)$$

If COUNT(C)>N-k then use

$$COUNT(C) = COUNT(\overline{U}) - COUNT(\overline{C}+T) + COUNT(T) + COUNT(\overline{B}\cdot U)$$

Because at least one of the equations can be evaluated, COUNT(C) (or SUM(C)) can be computed with at most 7 distinct queries and as few as 4 distinct queries. See [6,23] for details.

Assume that a data base has 300 records. Double trackers assure us that a user can isolate an individual even when only queries involving between 100 and 200 records are answered. In other words, even when two thirds of the query set sizes are forbidden, compromise is still possible. Compromise is not only possible but is achieved relatively fast as shown in [23].

Therefore, restricting the query set size does not prevent compromise in either key based systems or attribute based systems.

2.2 CONTROLS ON THE OVERLAP OF QUERY SETS.

If exact answers are given to queries for which query set size is restricted, there is always compromise. Naturally one wonders whether something better can be achieved by limiting the overlap in either key based queries or query set sizes in the case of attribute based querying. We shall see that such an attempt also proves to be useless.

The minimum overlap control restricts the response from queries that have more than a predetermined number of records in common with each prior query. No efficient implementation of this control is known: before responding, the query program could have to compare the current query group against every previous one. The results in [3,4,15] show that such an implementation would be futile in securing a data base.

In [3,4,15], the data base model used is key based (as in Example 1.1). The authors considered the problem of whether one can compromise such a data base if no two queries can overlap very much. We will assume that the data base has N records and that queries may be made about the sum of values in any subset of the records in the data base that consist of exactly k elements. We assume the further restriction that no two queries may overlap in more than r positions and that ℓ of the elements are known in advance. The authors of [4,15] studied the behavior of $S(N,k,r,\ell)$, the smallest number of queries that suffice to compromise the data base.

The following are some of the results of [4,15].

THEOREM 2.7

$$S(N,k,r,\ell) \geq 1 + \frac{k-(\ell+1)}{r} .$$

This is a lower bound for $S(N,k,r,\ell)$. The proof can be seen in [15].

The following are some specific results on the upper bound of the smallest number $S(N,k,r,\ell)$ of queries that are required to compromise a data base.

a)	$S(N,k,1,0) \leq 2k-1$	$N \geq k^2-k+1$
b)	$S(N,k,1,1) \leq 2k-2$	$N \geq (k-1)^2+2$
c)	$S(N,kp+d,p,2d-1) \leq 2k$	$N \geq k^2p+2\alpha$
d)	$S(N',kr,r,r-1) \leq S(N,k,1,0)$	$N' \geq rk^2$
e)	$S(N,k^2+1,k^2,0) \leq 2k+2$	$N \geq 2k^2+k+1$

We shall comment on the strength and weakness of this method. Consider the result in (a): $S(N,k,1,0) \leq 2k-1$, this says that for queries of size k, where no more than one common element can appear in the set of queries, a user can compromise the data base with at most 2k-1 queries. For example, consider the case k=3, the user can compromise the data base with at most 5 queries; i.e. $S(N,3,1,0) \leq 5$, where N must be at least 7.

As an example let us take the queries:

$$Q_1 = X_1+X_2+X_3 = S_1$$
$$Q_2 = X_4+X_5+X_6 = S_2$$
$$Q_3 = X_1+X_4+X_7 = S_3$$
$$Q_4 = X_2+X_5+X_7 = S_4$$
$$Q_5 = X_3+X_6+X_7 = S_5$$

The data field value of record R_7, represented by X_7, can be computed by:

$$X_7 = \frac{1}{3} (S_3+S_4+S_5-(S_1+S_2))$$

Result (b) states that when at most one overlap is allowed and that the user knows the value of one element (his own for example), he can compromise the data base in $2k-2$ queries. For example set $k=3$, then $S(N,3,1,1) \leq 4$ and N must be at least 6. To illustrate this consider the following set of queries:

$$Q_1=X_1+X_2+X_3=S_1$$
$$Q_2=X_1+X_5+X_6=S_2$$
$$Q_3=X_2+X_3+X_5=S_3$$
$$Q_4=X_2+X_4+X_6=S_4$$

Here we assume that X_1 is the known data value and the user wants to know the value of X_2. X_2 can be determined by:

$$X_2 = \frac{1}{2} (S_1+S_2-(S_3+S_4))$$

To the results in [4,15] we add the following results:

$$S(N,k,k-1,1)=2$$

The proof of this result is trivial. For example consider the case $k=4$; we then have $S(N,4,3,1)=2$ and N is greater or equal to 5. Assume that X_1 is the known element and we want to solve for X_3. This is done by asking the two queries:

$$Q_1=X_1+X_2+X_4+X_5=S_1$$

$$Q_2=X_2+X_3+X_4+X_5=S_2$$

and X_3 is determined by

$$X_3=X_1-S_1+S_2$$

The results of [4,15] show that controlling the overlap in queries does not help secure the data base and that the implementation of such a control would be very costly.

An effective method of preventing a user from isolating a record by overlapping queries is to partition the data base. Records are stored in groups and queries may apply to any set of groups, but never to subsets of records within the same group. It is therefore impossible to isolate a record. Partitioning has two severe practical limitations in dynamic data bases. First, the free flow of useful statistical information can be severely inhibited by excessively large groups or by ill-considered groupings. Second, forming and reforming groups as records inserted, updated and deleted from the data base can lead to costly bookkeeping.

2.3 DISTORTING THE DATA OR QUERY RESPONSE.

Several studies [8,12,16] have been made of rounding schemes for modifying the answers to queries. These methods aim to prevent inference by perturbing the responses. Under direct rounding, the answer to a query is rounded up or down by some small amount before it is released. Rounding by

adding a random value is not always secure since the correct answer can be deduced by averaging a sufficient number of responses to the same query. Rounding by adding a pseudo-random value that depends on the data is preferable because a given query always returns the same answer. The method can sometimes be subverted with trackers, by adding dummy records to the data base [16], or simply by comparing the responses to several queries in order to narrow the range of values containing the confidential value.[22]

In [1] Haq studies the behavior of a data base model that returns approximate answers to queries. The answer to a query is given by the system within an approximation of m. Haq also assumes that this approximation factor m, along with the upper and lower limits of a response is known to the user. Using these restrictions and assumptions the author found the following results:

THEOREM 2.8

There is an exact disclosure from a specific query q if and only if one of the following holds:

a) $A_q' = L_q - m$

b) $A_q' = U_q + m$

where L_q and U_q are the lower and upper limits of the response A_q and A_q' is the perturbed response.

For a proof of this theorem see [1].

This result is trivial as the following example will demonstrate.

Let m=5 and L_q=100, if the response A_q'=95 then condition (a) holds. We know that A_q' is within m of the true answer; therefore $90 \leq A_q \leq 100$. The user also knows that $A_q \geq 100$ since L_q=100, so we have $100 \leq A_q \leq 100$; therefore A_q (the true answer) must be equal to 100.

The results merely show that if a user has enough information he can compromise the data base.

A more realistic example of distorting the responses is given in [13]. Here the authors introduce the concept of "Weighted Sum" queries which has the form:

$$Q(Key_1, Key_2, \ldots, Key_k) = \sum_{j=1}^{k} a_j V_j$$

where V_j is the value of the data field of the record associated with Key_j and the weights a_j are unknown. Under these assumptions the following result was proven.

THEOREM 2.9

If V_p is known for at least one Key_p then it may be possible to determine

a) the value of k additional keys with no more than k(k+1) queries, and

b) the remaining N-k-1 elements of the data base with an

additional N-k-1 queries.

Proof: See [13].

Thus complete compromise is possible within k^2+N-1 queries. However, this result is only possible because the weights a_i are fixed by position and not associated with any one record. This would cause identical queries to have different responses. For example assume $V_1=10$, $V_2=15$, $a_1=3$ amd $a_2=2$. Using the query method in [13] we get:

$$SUM(Key_1, Key_2)=3\cdot 10+2\cdot 15=60$$

$$SUM(Key_2, Key_1)=3\cdot 15+2\cdot 10=65$$

However if one associates with every record a perturbation factor, i.e. the value of the data itself is perturbed, then exact compromise is not possible. The problem here is that in order to secure personal information one would have to give responses that are very imprecise, hence we may question the usefulness of such a data base.

Random rounding and perturbation can be useful when we are dealing with large numbers, since the error introduced has minimal effect on the data; however with small counts and small values the data can be severely distorted. Also we must be careful when dealing with tables and cross tabulations to make sure that totals and sub-totals add up correctly. See [8,9] for discussions related to random rounding.

2.4 RANDOM SAMPLING.

All the controls listed above are subverted by a single basic principal of compromise: the user can control the composition of each query set, thus enabling him to isolate a single record or value by intersecting query sets [22]. In random sampling, a user may apply responses to a set of records no longer selected by him. This prevents compromise by depriving the user of the ability to isolate a known record. For example, the 1960 U.S. Census was distributed on tape as a random sample of one record in 1000. A user would then have at best a 1/1000 chance of associating a given record with the right individual.

However, in a small to medium scale data base, a small fixed sub-sample would not be statistically significant and would not represent the current status of the data. For this reason, random sampling has been ignored as a possible inference control in modern statistical data base systems.

CHAPTER 3

PROTECTION OF KEY SPECIFIED DATA BASES

As we have seen in Chapter 2, controls such as restricting the overlap between any two queries, random perturbation and fixing the size of the query have not been successful in securing a data base that allows key specified queries in the system.

In Chapter 1, we introduced the concept of a selection query. A selection query is a key based query of the form: "What is P of the following list of k employees?" Where P is any legal selection query predicate such as MAX, MIN, MEDIAN, LARGEST, etc.

It has been shown in [2] that every sufficiently large data base can be compromised by selection queries (regardless of P) in no more than $m(k) \leq 4k^2$ queries. Thus no sufficiently large data base that allows selection queries can be secure.

It is, therefore, dramatically clear that protecting such a data base against compromise is very difficult. This leads us to consider what happens if the data base can "lie". A data base that lies is not bound to give truthful answers to queries. In [2] a strategy of lying is reported whereby for a query that requests the median salary of employees N_1, N_2,

N_3, N_4, the system returns simply the salary of one of the employees, say N_3, whether or not N_3 is the median. This approach seems entirely reasonable; however, the main result in [2] is surprising:

> Even a data base that lies in the above way (i.e. answers a selection query with the value of any key in the query) can always be compromised (and in relatively few steps).

This shows that any data base, permitting key specified queries, that returns exact values for all queries can always be compromised.

In section 3.1, an attempt is made to introduce error in the response of linear queries in such a manner that the level of error is minimum; but the errors in the inferred values can be maximized. Unfortunately such min-max method does not seem to work well all the time and we shall comment why this is so in section 3.1.

In section 3.2 we will discuss "forbidden query sets". This set contains queries that are forbidden, i.e. if the response to these queries are not given then the data base is secure. An algorithm that determines the content of this forbidden query set will be given and analyzed. We will discuss the implementation of this method on a small and medium size random data base and discuss the results.

The main thrust of this chapter is to give methods that lie on key based queries so that the integrity of the data base is not unduly affected, the response statistics are close to true statistics and the inferred individual values have large errors.

3.1 USING CONVEX LINEAR COMBINATIONS TO SAFEGUARD A KEY BASED DATA BASE.

In this section our discussion will be related to key based queries and the behavior of compromise when an error is introduced in the responses.

Consider the following example. A data base has 3 records (N=3) and queries specifying exactly 2 records are allowed. The following system can be set up to solve for individual values.

$$Q_1=AVE(V_1,V_2) = A_1= \frac{V_1+V_2}{2}$$

$$Q_2=AVE(V_1,V_3) = A_2= \frac{V_1+V_3}{2}$$

$$Q_3=AVE(V_2,V_3) = A_3= \frac{V_2+V_3}{2}$$

The values of V_1, V_2 and V_3 are computed as follows:

$$V_1= A_1+A_2-A_3$$

$$V_2= A_1-A_2+A_3$$

$$V_3=-A_1+A_2+A_3$$

We now assume that some error is introduced into A_i such that $A_i=A_i^T+\epsilon_i$ where A_i is the response to Q_i, A_i^T is the true answer and ϵ_i is the amount of error introduced in the response. The way the error is introduced is through convex linear combinations which will give us a weighted average of (V_1,V_2), (V_1,V_3) and (V_2,V_3) such that

$$\text{MIN}(V_i,V_j) \leq \text{weighted average of } (V_i,V_j) \leq \text{MAX}(V_i,V_j)$$

The weighted average of a query is computed as follows:

$$A(Q)=\frac{\alpha_1V_1+\alpha_2V_2+\ldots+\alpha_kV_k}{\alpha_1+\alpha_2+\ldots+\alpha_k}$$

So when k=2 we get the following system:

$$A_1=\frac{\alpha_1V_1+\alpha_2V_2}{\alpha_1+\alpha_2} \qquad A_1^T=\frac{V_1+V_2}{2} \tag{3.1}$$

$$A_2=\frac{\alpha_1V_1+\alpha_3V_3}{\alpha_1+\alpha_3} \qquad A_2^T=\frac{V_1+V_3}{2} \tag{3.2}$$

$$A_3=\frac{\alpha_2V_2+\alpha_3V_3}{\alpha_2+\alpha_3} \qquad A_3^T=\frac{V_2+V_3}{2} \tag{3.3}$$

We want to choose the α_i's such that

$$E_i=\left|\frac{A_i-A_i^T}{A_i^T}\right| \leq \epsilon$$

for all i, where ϵ is an upper bound on the amount of error we will introduce. Another criteria for choosing the α_i's is that we want to maximize

$$E_i'=\left|\frac{V_i'-V_i}{V_i}\right|$$

which is the resulting relative error in the inferred values. Here V_i' is the value of V_i inferred by the user from the erroneous responses A_1, A_2 and A_3.

Therefore our aim is to maximize E_i' while keeping $E_i \leq \epsilon$ for all i.

Let us now analyse the behavior of such a system when k=2. Without loss of generality, assume $V_1<V_2<V_3$. From equations (3.1), (3.2) and (3.3) we get

$$\frac{V_2-U_1}{U_1-V_1} < \frac{\alpha_1}{\alpha_2} < \frac{V_2-L_1}{L_1-V_1} \tag{3.4}$$

$$\frac{V_3-U_2}{U_2-V_1} < \frac{\alpha_1}{\alpha_3} < \frac{V_3-L_2}{L_2-V_1} \tag{3.5}$$

$$\frac{V_3-U_3}{U_3-V_2} < \frac{\alpha_2}{\alpha_3} < \frac{V_3-L_3}{L_3-V_2} \tag{3.6}$$

where

$$L_1=MAX(V_1,\ A_1^T(1-\epsilon)) \quad U_1=MIN(V_2,\ A_1^T(1+\epsilon))$$

$$L_2=MAX(V_1,\ A_2^T(1-\epsilon)) \quad U_2=MIN(V_3,\ A_2^T(1+\epsilon))$$

$$L_3=MAX(V_2,\ A_3^T(1-\epsilon)) \quad U_3=MIN(V_3,\ A_3^T(1+\epsilon))$$

Set $P=\frac{\alpha_1}{\alpha_3}$ and $Q=\frac{\alpha_2}{\alpha_3}$, hence $\frac{P}{Q}=\frac{\alpha_1}{\alpha_2}$. Substituting this into equations (3.4), (3.5) and (3.6) we get

$$G(1) < \frac{P}{Q} < H(1) \tag{3.7}$$

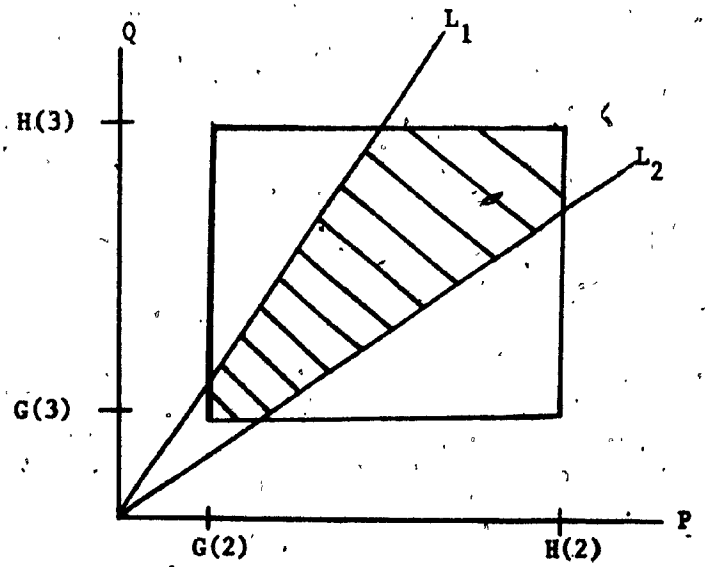
$$G(2) < P < H(2) \tag{3.8}$$

$$G(3) < Q < H(2), \tag{3.9}$$

where G(1) and H(1) correspond to the bounds of $\frac{P}{Q} = \frac{\alpha_1}{\alpha_2}$ in equation (3.4); G(2), H(2) for those in equation (3.5); and G(3), H(3) for those in equation (3.6). The six inequalities in equations (3.7), (3.8) and (3.9) must hold for any choice of the α_i's if we want to satisfy the condition $E_i < \epsilon$ for all i.

The following figure shows the region determined by the set of linear inequalities. For α_i's chosen in this region $E_i \leq \epsilon$ is true for all i.

FIGURE 3.1



The equations for $Line_1$ and $Line_2$ come from the constraint in equation (3.7):

$$G(1) < \frac{P}{Q} < H(1)$$

Therefore we have the following equations for $Line_1$ and $Line_2$:

$$Line_1 = P = H(1)Q$$

$$\text{Line}_2 = P = G(1)Q$$

We now have a bounded region for α_1, α_2 and α_3 and want to choose the α's within this bound such that

$$E_i' = \frac{V_i' - V_i}{V_i}$$

the error in the inferred values is maximized. V_i' for i=1, 2, 3 is computed as follows:

$$V_1' = A_1 + A_2 - A_3$$

$$V_2' = A_1 - A_2 + A_3$$

$$V_3' = -A_1 + A_2 + A_3$$

therefore

$$E_1' = \frac{|A_1 + A_2 - A_3 - V_1|}{V_1}$$

$$E_2' = \frac{|A_1 - A_2 + A_3 - V_2|}{V_2}$$

$$E_3' = \frac{|-A_1 + A_2 + A_3 - V_3|}{V_3}$$

Replacing the α's in equations (3.1), (3.2) and (3.3) by P and Q, where $P = \frac{\alpha_1}{\alpha_3}$ and $Q = \frac{\alpha_2}{\alpha_3}$ we get the following:

$$A_1 = \frac{PV_1 + QV_2}{P+Q}$$

$$A_2 = \frac{PV_1 + V_3}{P+1}$$

$$A_3 = \frac{QV_2 + V_3}{Q+1}$$

Substituting this into the E_i''s we get functions in terms of

P and Q. Therefore

$$E_1' = f_1(P,Q) = \frac{|(Q^2-P^2)(V_3-V_1)+Q(1-P^2)(V_2-V_1)|}{V_1(1+P)(1+Q)(P+Q)}$$

$$E_2' = f_2(P,Q) = \frac{|(Q^2-P^2)(V_2-V_3)+P(1-Q^2)(V_1-V_2)|}{V_2(1+P)(1+Q)(P+Q)}$$

$$E_3' = f_3(P,Q) = \frac{|(P(1-Q^2)(V_3-V_1)+Q(1-P^2)(V_3-V_2)|}{V_3(1+P)(1+Q)(P+Q)}$$

The problem is now one of optimization: we want to optimize (find the maximum) of $f_1(P,Q)$, $f_2(P,Q)$ and $f_3(P,Q)$ subject to the constraints in equations (3.7), (3.8) and (3.9).

EXAMPLE 3.1

For example, consider the following case: We have a system that allows queries about any two individuals. Therefore to set up a system of equations we will need 3 queries involving 3 records. Let the values of these records be 100, 200, 300 respectively. We then assume ϵ=0.10 (i.e. we allow at most a 10 percent error in the responses). We now have the following information:

$$V_1=100 \quad L_1=135 \quad U_1=165$$
$$V_2=200 \quad L_2=180 \quad U_2=220$$
$$V_3=300 \quad L_3=225 \quad U_3=275$$

Using these values we determine that the limits of the ratios of equations (3.4), (3,5) and (3.6) are

$$\frac{7}{13} < \frac{a_1}{a_2} < \frac{13}{7}$$
$$\frac{2}{3} < \frac{a_1}{a_3} < \frac{3}{2}$$
$$\frac{1}{3} < \frac{a_2}{a_3} < 3$$

and from this we get the constraints:

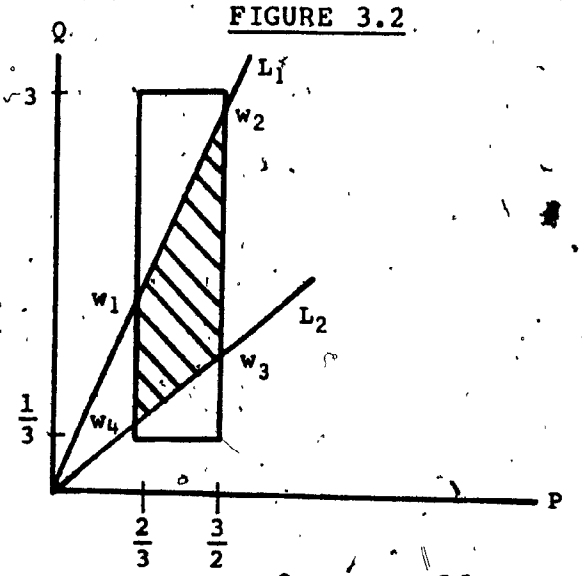
$$\frac{7}{13} < \frac{P}{Q} < \frac{13}{7}$$
$$\frac{2}{3} < P < \frac{3}{2}$$
$$\frac{1}{3} < Q < 3$$

The first constraint gives us $Line_1$ and $Line_2$, while the other two constraints limit the ranges on the P and Q axes.

Figure 3.2 shows the region in which we must choose the a's in order to garantee that $E_i < \varepsilon$ for i=1,2,3.

Using a well known optimization technique we find the maximums of f_1, f_2 and f_3.

FIGURE 3.2



MAX($|f_1|$) occurs at P= $\frac{2}{3}$ and Q= $\frac{26}{21}$ (W_1)

and at P= 1.5 and Q= $\frac{21}{26}$ (W_3)

MAX($|f_2|$) occurs at P= 1.5 and Q= $\frac{39}{14}$ (W_2)

and at P= $\frac{2}{3}$ and Q= $\frac{14}{39}$ (W_4)

MAX($|f_3|$) occurs at P= $\frac{2}{3}$ and Q= $\frac{14}{39}$ (W_4)

and at P= 1.5 and Q= $\frac{39}{14}$ (W_2)

As we see the maximums of the functions occur at intersection points. Therefore to determine at which points the maximum occurs, one need only evaluate the functions at each intersection point and compare the results. Let us now examine the maximums.

MAX(f_1)= 0.4032 at W_1 and W_3

MAX(f_2)= 0.0570 at W_2 and W_4

MAX(f_3)= 0.1953 at W_2 and W_4

This says that the maximum error that can be introduced in the inferred value of V_1 is 40.32 percent if we use W_1 or W_3 as the coordinates in determining α_1, α_2 and α_3. To determine the α's for any given W_i, we set $\alpha_1=P$, $\alpha_2=Q$ and $\alpha_3=1$, since $P=\frac{\alpha_1}{\alpha_3}$ and $Q=\frac{\alpha_2}{\alpha_3}$.

The choice of which $W=(P,Q)$ to use in determining the α's is made by choosing one of the maximums that occurs in the function f_i such that $MAX(f_i)$ is the minimum. In the above example W_2 or W_4 is chosen. Assume we choose W_2. Thus we will have $\alpha_1=1.5$, $\alpha_2=\frac{39}{14}$ and $\alpha_3=1$, and a user will receive as responses to the queries

QUERY	RESPONSE	(TRUE VALUE)	REL. ERROR
$AVE(V_1,V_2)$	=165.00	(150.00)	10.0%
$AVE(V_1,V_3)$	=180.00	(200.00)	10.0%
$AVE(V_2,V_4)$	=226.42	(250.00)	9.4%

From these three queries we have the system:

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} V_1' \\ V_2' \\ V_4' \end{bmatrix} = \begin{bmatrix} 165.00 \\ 180.00 \\ 226.42 \end{bmatrix}$$

which has a unique solution,

V_1'=118.58 (V_1=100; REL. ERROR=18.58 percent)
V_2'=211.42 (V_1=200; REL. ERROR= 5.71 percent)
V_3'=241.42 (V_3=300; REL. ERROR=19.53 percent)

This approach attempts to maximize the relative errors in the inferred values V_i', (i=1,2,3) while maintaining the response error below ε. This is possible when the three values of the system are known. For example, if we have a data base with 4 records, N_1, N_2, N_3 and N_4 and a user requests AVE(N_1,N_2). In order to determine the α's the system must know the value of the third record that is to be used. This can be either N_3 or N_4, therefore one must have an approach to determine the value of the third record.

We have considered two approaches; the first one will predetermine an α for each record (the α's are fixed); while in the second approach, the α's will be determined at the time of querying (the α's are not fixed for each record). We will assume that a data base has N records and a query can ask statistics (average) about any two of them at a time (this can be generalized for a query set size>2).

The first approach (fixed α's) is described in the following algorithm.

ALGORITHM 3.1

1) Order the records such that

$$V_1 \leq V_2 \leq \ldots \leq V_N$$

2) Divide the data base into k+1 groups as evenly as possible, where k is the size of the query set (k=2 in our case).

$G_1 = (V_1, \ldots, V_x)$
$G_2 = (V_{x+1}, \ldots V_{2x})$
$\vdots$
$G_{k+1} = (V_{kx+1}, \ldots V_N)$

where $x = \frac{N}{k+1}$.

3) Find a group representative for each group. For example we may have the group representative chosen as

a) Average $(REP_1 = AVE(V_1, \ldots, V_x))$

or b) Median $(REP_1 = MED(V_1, \ldots, V_x))$

where $x = \frac{N}{k+1}$.

4) Use the group representatives REP_i, $(i=1,k+1)$ to determine α_G for each group.

5) Determine individual α's for each record as follows:

for each record in $group_i$ do

$$\alpha_j = \alpha_{G_i} - \frac{D_j}{S}$$

where $D_j = V_j - REP_i$

and $S^2 = \Sigma D_j^2$.

Do (5) for each group G_i, $i=1,k+1$

6) Now all the α's are determined; answer a query $AVE(N_i, N_j)$ with

$$ANS = \frac{\alpha_i V_i + \alpha_j V_j}{\alpha_i + \alpha_j}$$

The maximization algorithm, which determines the best α's for certain values, is used only once (in step 4). These

values are best when the values in a linear system are close to REP_i, i=1,k+1. Therefore, if we have a system asking the queries:

$$AVE(REP_1, REP_2)$$
$$AVE(REP_1, REP_3)$$
$$AVE(REP_2, REP_3)$$

we would get "best" results. However, since in any query we will have records whose values differ from the group representatives, we cannot predict the error in the results.

The results are classified into two groups: queries that involve records within the same group and those which involve records not in the same group.

For queries involving records in the same group, the effect the α's have on the error depends only on the deviation introduced in step (5), so we have no control on the error introduced in the response (step 6). Therefore, the error in the inferred values is not maximized.

For queries involving records not in the same group the results are better, but since a value can differ significantly from its group representative, the control on the response error and inferred error is minimum.

We ran test on data bases containing 10 records and 100 records, using both the average and median value for the group representatives. The results of these test runs tend

to agree with the above observations that control on the error introduced in the response is lost and the error in the inferred values is not maximized. In some cases, the error in the response was as high as 60 percent. With such a lack of control over the amount of error introduced in the response, this approach would render the statistics of the data base useless.

The next approach is an attempt to regain more control over the amount of error introduced in the response. The algorithm does not predetermine the weights (α's) but computes their values at each query. Therefore the weight of a certain record is not fixed. By this we mean that two different queries involving the same record will not necessarily have the same weight. For example consider the queries

$$Q_1 = AVE(R_1, R_2) = \frac{\alpha_1 V_1 + \alpha_2 V_2}{\alpha_1 + \alpha_2}$$

$$Q_2 = AVE(R_1, R_3) = \frac{\alpha_1 V_1 + \alpha_3 V_3}{\alpha_1 + \alpha_3}$$

Here α_1 in Q_1 is not necessarily equal to α_1 in Q_2.

The concept of this approach is that if the records of a query are in the same group, then we subdivide the group until the records are in different groups. This will take care of the lack of control in the response error for queries involving records belonging to the same group. This approach is outlined in Algorithm 3.2 (this can be easily changed for

k>2).

ALGORITHM 3.2

1) Sort the records such that

$V_1 < V_2 < \ldots < V_N$

2) Read in a query $Q=AVE(R_i,R_j)$.

3) Initialize the boundaries of the data base m=1 and mm=N (initially all the records are in the same group).

4) Divide the data base (or group) into 3 groups.

5) if V_i and V_j are in the same group G then

set m =lower boundary of G

mm=upper boundary of G

else go to step (8).

6) If the size of the new group is less then 3, then extend one of its boundaries by 1.

7) Repeat from step (4).

8) (We now have 3 groups in which V_i and V_j are not in the same group)

Find a representative for each group

a) REP_i=average of group G_i, i=1,2,3

b) REP_i=median of group G_i, i=1,2,3

c) use the value of R_i and R_j to represent the

groups containing R_i and R_j and use either the average ot the median for the other group.

9) Compute α_{G_1}, α_{G_2} and α_{G_3}.

10) Compute individual α's for R_i and R_j

a) $\alpha_i = \alpha_{G_i} - \frac{D_i}{S}$ (for 8(a) or 8(b))

b) $\alpha_i = \alpha_{G_i}$ (for 8(c))

D_j and S is the same as in Algorithm 3.1

(repeat for α_j)

11) Return the query response as

$$ANS = \frac{\alpha_i V_i + \alpha_j V_J}{\alpha_i + \alpha_j}$$

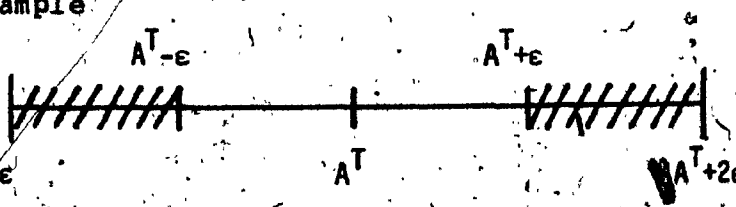
If step 8(a) or 8(b) is used, we encounter the same difficulties as in the previous method. However, these difficulties are less noticeable because the group sizes are smaller. This causes the difference between the actual record value and the group representative to be less, hence we get better results in general, but we still can not control the error in the response as much as we would like to.

If step 8(c) is used, we have complete control over how much error is introduced in the response to any query, since the α's for each query are computed using exact values of R_i and R_j along with a value representing the third group. This allows us to keep the error in the response within a certain

range, say plus or minus 10 percent. As we see, using 8(c) allows us to control ER (error in response), however we do not have much control on the error in the inferred values due to the fact that each query is answered independently from each other while to infer a value of a record more than one query is needed. However, if 8(c) and 10(b) are used, we get good results from test runs on sample data bases. By "good" we mean that most of the inferred values have error greater than the maximun error allowed in the query responses.

This whole concept of trying to maximize the error in the inferred values by controlling the error in the responses did not prove to be successful, because, as we have seen, we can only control the response error such that the response is within a certain range of the true response. This is accomplished in the second approach (using 8(c)) where we need to compute a new set of weights for every query, which is a lot of work. We can accomplish the same thing by finding the true response and then perturbing it in such a way that the response falls within the desired range.

For example



if we return the response such that it lies within the shaded areas we will achieve the same results (or results which are as good) with very little work.

set F with any system$_i$ is not the empty set and the size |F| is minimum. By this we mean that at least one query in every system belongs to the set F. A possible forbidden query set F for our sample data base in Table 3.7 is

F={Q(N1,N2,N3,N4),Q(N1,N2,N5,N6),Q(N3,N4,N5,N6)}

By dissallowing the true responses to three of the fifteen possible queries, we make it impossible for a user to infer any exact information by solving one of the six linear systems of equations.

To simplify the notation in this section we will let the records in the data base be identified by integers. In Table 3.7, the record identifiers N1 through N6 will be represented by the integers 1 through 6. A subset of the integers will therefore represent a query. For example SUM(1,2,3,4) represents the query SUM(N1,N2,N3,N4). A system, as defined earlier, involves k+1 queries and k+1 records; therefore the subset (1,2,3,4,5) represents System 1 of the systems listed above.

Let us now consider the complexity of determining the forbidden query set F. Our first aim is to include at least one query from every possible system in F. If the set F contains all the queries then this goal is achieved. The second aim of this problem is to determine a set F that has as few queries in it as possible. Let $C(q_i)$ be the set of systems that contain the query q_i. For example let

$q_i=(1,2,3,4)$, then $C(q_i)=\{$system 1, system 2$\}$ for the data base in Table 3.7. The minimum size of the set F occurs when the set C of each query in F is as distinct as possible.

Before we discuss a method for obtaining forbidden queries we need the notion of "maximal independent set" from Graph Theory which we shall explain below.

Let G=(V,E) be a graph G where V is the set of vertices in the graph and E is the set of edges in the graph. A subset S of the vertices of G is said to be "an independent set" of G if any pair of distinct vertices in S are not adjacent to each other. A "maximal independent set" is an independent set that is not contained in any other independent set.

It follows from this definition that the vertices not in a maximal independent set are adjacent to at least one of the vertices in the maximal independent set. A "maximum independent set" is the largest of the maximal independent sets.

Let V be the set of vertices in G corresponding to the set of all possible queries in a data base. The set S contains all possible systems of a data base. Two vertices are adjacent if and only if they belong to the same system in S. Such a graph will be called a "query graph". Consider the example of a data base with N=5 records allowing queries of size k=3. In Table 3.8 the sets of possible queries and

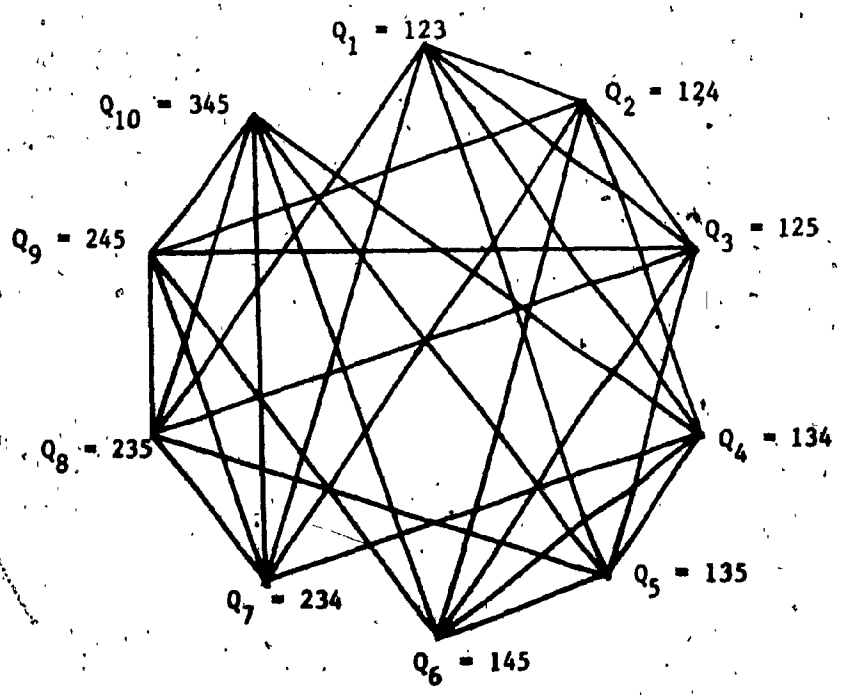
systems are listed.

TABLE 3.8

POSSIBLE QUERIES	POSSIBLE SYSTEMS
$Q_1=(1,2,3)$	$S_1=(1,2,3,4)$
$Q_2=(1,2,4)$	$S_2=(1,2,3,5)$
$Q_3=(1,2,5)$	$S_3=(1,2,4,5)$
$Q_4=(1,3,4)$	$S_4=(1,3,4,5)$
$Q_5=(1,3,5)$	$S_5=(2,3,4,5)$
$Q_6=(1,4,5)$	
$Q_7=(2,3,4)$	
$Q_8=(2,3,5)$	
$Q_9=(2,4,5)$	
$Q_{10}=(3,4,5)$	

Figure 3.3 is the query graph for a data base with N=5 and k=3.

FIGURE 3.3



By finding a maximum independent set of the graph in Figure 3.3, we will determine the largest number of queries that have distinct $C(q_i)$'s. A maximum independent set for the graph in Figure 3.3 is MIS=$\{Q_1, Q_6\}$ and $C(Q_1)=\{S_1, S_2\}$ and $C(Q_6)=\{S_3, S_4\}$.

Therefore, by including Q_1 and Q_6 in the forbidden query set, the systems $C=C(Q_1)\cup C(Q_6)=\{S_1, S_2, S_3, S_4\}$ become unsolvable. After this step we see that system S_5 has not been cancelled. By a cancelled system we mean any system of which one of its queries appears in the set F. The set F is complete when all the systems are cancelled.

The next step in this procedure is to update the graph and find the maximum independent set of the updated graph. The queries in this set are then included in the set F. This step is repeated until all the systems have been cancelled.

In order to update the graph, we first delete from the set S the systems in $C(q_i)$ for all q_i in F. The graph G' is formed in the same manner as we formed the original graph G. Once this is done we delete from G' and from V (the set of vertices) all vertices that have degree equal to zero.

In our example we initially had the following sets:

$S=\{S_1, S_2, S_3, S_4, S_5\}$

$V=\{Q_1, \ldots, Q_{10}\}$

$E=\{(v,w) \mid v,w \in V; v,w$ to a same system in S$\}$

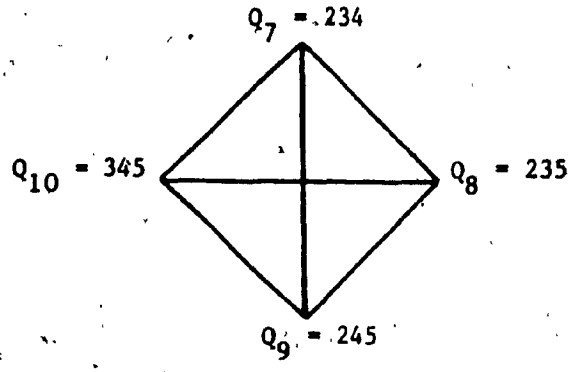
G=(V,E) is the graph in Figure 3.3

We found that a maximum independent set of G was $\{Q_1, Q_6\}$. Updating S we get $S' = \{S_5\}$. We can now determine the set

$$E' = \{(Q_7, Q_8), (Q_7, Q_9), (Q_7, Q_{10}), (Q_8, Q_9), (Q_8, Q_{10}), (Q_9, Q_{10})\}$$

From this we see that vertices Q_1 through Q_6 will have degree equal to zero. Hence the set V becomes $V' = \{Q_7, Q_8, Q_9, Q_{10}\}$. We can mow form the graph $G' = (V', E')$ which is in Figure 3.4.

FIGURE 3.4



The set $\{Q_7\}$ is a maximum independent set of this graph. By deleting all the systems in S' that are cancelled by Q_7 we see that S' is now empty. This is the criteria to stop. Therefore the set F will be

$$F = \{Q_1, Q_6, Q_7\}.$$

Algorithm 3.3 describes this procedure that determines the forbidden query set F by finding maximum independent sets of the associated query graph and the updated graphs.

ALGORITHM 3.3

1) (initialization step)

F= empty set.

V= $\{Q_1,\ldots,Q_m\}$ ($m=\binom{N}{k}$)

S= $\{S_1,\ldots,S_s\}$ ($s=\binom{N}{k+1}$)

E= $\{(v,w) \mid v,w \in V$, and v,w belongs to the same system$\}$

2) Form graph G=(V,E).

3) Find MIS (maximum independent set) of G.

4) F=F$\cup$MIS.

5) Update the sets S, V and E. (as described above)

6) If S empty stop else repeat from step (2).

Analysis of Algorithm 3.3

Analysing this algorithm we see that the major cost will occur in step (3), i.e. finding the maximum independent set of the graph. In [27,28] the authors have shown that finding such a set is very costly for large graphs. In [27] the authors present an algorithm which finds a maximum independent set of an m-vertex graph in time $O(2^{\frac{m}{3}})$ where $m=\binom{N}{k}$ in our case. In a data base that has N=20 records and allows queries of size k=4, the query graph will have m=4845 vertices. Even for small N, the number of vertices in the query graph will be large, therefore the cost of

Algorithm 3.3 will be very high. Even with such a costly algorithm the size of the forbidden query set obtained is not always minimum as the following example demonstrates.

Consider the data base system with N=6 and k=3. There are $\binom{6}{3}=20$ possible queries and $\binom{6}{4}=15$ possible systems in the data base. Using Algorithm 3.3, we find a maximum independent set, for the original graph, that has 4 vertices. The 4 queries associated with these vertices appear in 3 distinct systems each. Updating S in step (5) we have $S=\{S_6,S_8,S_{11}\}$. These 3 systems are

SYSTEM 6	SYSTEM 8	SYSTEM 11
Q(1,2,5)	Q(1,3,4)	Q(2,3,4)
Q(1,2,6)	Q(1,3,6)	Q(2,3,5)
Q(1,5,6)	Q(1,4,6)	Q(2,4,5)
Q(2,5,6)	Q(3,4,6)	Q(3,4,5)

As we can see, the queries in these systems are distinct. In order to cancel every system we must choose one query from each system. Therefore 3 additional queries will be added to the set F. The set F will contain 7 queries

$$F=\{123,145,246,356,125,134,234\}.$$

It can be shown that with only 6 queries in F all the systems will be cancelled. This is achieved by choosing an independent set that is not maximum on the first pass. If in the original graph we choose an independent set that consists of the queries {123,145,246}, we are left with 6 systems that are not cancelled. The 6 systems are listed below:

SYSTEM 6	SYSTEM 8	SYSTEM 9
Q(1,2,5)	Q(1,3,4)	Q(1,3,5)
Q(1,2,6)	Q(1,3,6)	Q(1,3,6)
Q(1,5,6)	Q(1,4,6)	Q(1,5,6)
Q(2,5,6)	Q(3,4,6)	Q(3,5,6)

SYSTEM 11	SYSTEM 13	SYSTEM 15
Q(2,3,4)	Q(2,3,5)	Q(3,4,5)
Q(2,3,5)	Q(2,3,6)	Q(3,4,6)
Q(2,4,5)	Q(2,5,6)	Q(3,5,6)
Q(3,4,5)	Q(3,5,6)	Q(4,5,6)

By obsevation we see that 3 queries suffice to cancel these 6 systems. These are

Q(1,5,6) cancels systems 6 and 9.

Q(3,4,6) cancels systems 8 and 15.

Q(2,3,5) cancels systems 11 and 13.

Therefore F={123,145,246,156,346,235} cancels all 15 possible systems.

As the above example demonstrates, the method described in Algorithm 3.3 does not always find the best forbidden query set. Hence we look for methods that extract a forbidden query set of minimal size at a moderate cost.

We note that each of the $\binom{N}{k}$ possible queries appear in N-k systems. Therefore by adding a query to the forbidden query set N-k systems are cancelled. If, for every query q_i in F, the $C(q_i)$'s (the set of systems cancelled by q_i) are distinct, then the absolute lower bound of the size of F occurs. In other words this bound occurs when every query in F cancels exactly N-k distinct systems. Therefore the lower

bound of |F|, where F is a forbidden query set , is given by $\frac{S}{N-k}$, where $S=\binom{N}{k+1}$ is the number of systems. However this lower bound rarely occurs, since most of the time we will have overlap of systems when we choose our forbidden query set, as we have seen in the previous examples.

We will now consider cases when k is small and find the optimum forbidden query set by manual computations. When k=2 we are able to determine a procedure that finds the smallest possible forbidden query set. From this procedure we found that the size of the smallest forbidden query set is a function of N (the number of records in the data base). We state this result below:

THEOREM 3.1

The minimum forbidden query set size for k=2 is given by $\frac{N(N-2)}{4}$ when N is even and $\frac{(N-1)^2}{4}$ when N is odd.

We prove this after we describe the procedure that finds the smallest forbidden query set for the case k=2.

ALGORITHM 3.4 (Construction of F when k=2 and N>2)

1) If N is even set $P=\frac{N}{2}$

   If N is odd set $P=\frac{N+1}{2}$

2) Let A={1,...,P}

   B={P+1,...,N}}

   (where the integers {1,...,N} represent the record

identifiers)

3) Let $A' = \{(x,y) \mid x,y \in A, x \neq y\}$

$B' = \{(x,y) \mid x,y \in B, x \neq y\}$

4) $F = \{A' \cup B'\}$

In order to see that this algorithm correctly computes a forbidden query set, note that every system must have k+1=3 queries of which at least one of them should belong to either A' or B'. For example consider the case N=6. The step of the algorithm are traced below:

1) Set P=3

2) Let A={1,2,3}

B={4,5,6}

3) $A' = \{(1,2),(1,3),(2,3)\}$

$B' = \{(4,5),(4,6),(5,6)\}$

4) F={(1,2),(1,3),(2,3),(4,5),(4,6),(5,6)}

With N=6 and k=2 there are $\binom{N}{k+1}=20$ different systems. It is easy to see that these 20 systems are cancelled by the 6 queries in the set F shown in step (4).

Proof of Theorem 3.1

The proof of the theorem directly follows from the Algorithm 3.4. Consider the case when N is even. In step

(2) we have two sets of records each of size $P=\frac{N}{2}$. Step (3) constructs all possible queries from the records in each subset. Therefore since both subsets A and B have P records the number of queries that can be formed from each subset is $\binom{P}{2}$. The set F contains the queries formed from the records in A and the records in B for a total of $2\binom{P}{2}$ queries. So when N is even we have

$$|F| = 2\binom{P}{2} = \frac{2P!}{(P-2)!2!}$$

$$= P(P-1)$$

$$= \frac{N}{2}\left(\frac{N}{2}-1\right)$$

$$= \frac{N^2-2N}{4}$$

$$= \frac{N(N-2)}{4}$$

When N is odd, one subset contains $\binom{P}{2}$ queries, while the other subset contains $\binom{P-1}{2}$ queries where $P=\frac{N+1}{2}$. So the size of F in this case will be

$$|F| = \binom{P}{2} + \binom{P-1}{2}$$

$$= \frac{P!}{(P-2)!2!} + \frac{(P-1)!}{(P-3)!2!}$$

$$= \frac{P(P-1)}{2} + \frac{(P-1)(P-2)}{2}$$

$$= \frac{(P-1)(P+P-2)}{2}$$

$$= \frac{2(P-1)(P-2)}{2}$$

$$= (P-1)^2$$

$$= \frac{(N+1-2)^2}{4}$$

$$= \frac{(N-1)^2}{4}$$

When k=2 the number of possible queries is $\binom{N}{2} = \frac{N(N-1)}{2}$. The above results show that the set F contains approximately 50 percent of all the queries. We have seen that the absolute lower bound on the size of F is $\frac{S}{N-k}$. In other words the ratio (R) of queries in F over the whole set of queries is

$$R = \frac{\frac{S}{N-k}}{\binom{N}{k}}$$

$$= \frac{k!}{(k+1)!}$$

$$= \frac{k!}{(k+1)k!}$$

$$= \frac{1}{k+1}$$

Therefore, when k=2 the absolute lower bound for |F| (this occurs when every query in F each cancel N-k distinct systems) is $\frac{1}{3}$ the total query set, However the optimum size of F (except when N=3 or 4) is larger than this bound as F contains about 50 percent of the queries when N gets large.

When k=3, the absolute lower bound of F is 25 percent of the total number of queries. Again this bound is not reached except in the trivial case when N=4. In Table 3.9, the optimum size of |F| is shown for k=3 and some small values of N.

TABLE 3.9

N	SIZE OF F	PERCENT OF TOTAL QUERY SET
4	1	25.0%
5	3	30.0%
6	6	30.0%
7	12	34.2%

In earlier examples we have considered the cases N=5 and N=6. When N=7, we can form 35 queries and 35 systems. This example is small enough to determine the set F by hand. By considering every possible set F and choosing the smallest, we found that an optimum forbidden query set is

F={123,127,137,145,156,235,246,247,267,346,357,467}.

When N=8 there are 56 queries and 70 systems. We see that the number of queries and systems get very large as N increases and to determine the optimum set F becomes very difficult, if not impossible, by hand.

We have seen earlier that by using maximum independent sets we could construct the set F. This is very costly and does not assure optimum results. Since the query graph initially is a uniform graph, by this we mean that every vertex has the same degree, this makes it possible to construct the set F by using an algorithm that is less costly. This algorithm behaves in the same manner as Algorithm 3.3 except that it determines a maximal independent set instead of a maximum independent set. Due to the

relationship between the queries and the systems, Algorithm 3.5 does not need to set up the query graph.

To each query we will associate a count; this count indicates the number of non-cancelled systems the query will appear. Initially no system is cancelled, therefore the count of each query is N-k since each query appears in N-k systems. We will now describe the algorithm in detail.

ALGORITHM 3.5

This algorithm forms a forbidden query set by successively inserting in this set queries that cancel a maximum number of systems. This approach is "greedy" since it tries to minimize the number of queries in the forbidden query set by choosing queries that cancel a maximum number of systems.

1) Set up a list of nodes with two fields:
   $node_i = (query_i, count_i)$ i=1 to $\binom{N}{k}$
   where $query_i$ is the query identifier and $count_i$ is the number of systems in which $query_i$ appears.

2) Set $count_i = N-k$ for i=1 to $\binom{N}{k}$
   (since initially every query appears in N-k systems)

3) Reapeat
   a) choose one query Q_i such that $count_i$ is MAX
   b) find all systems in which Q_i appears
   c) for each query Q_j in systems found in (b) do

$count_j = count_j - 1$

until $count_i = 0$ for all i

It is easy to see from step (3) that a forbidden query set is indeed obtained. We comment on two other aspects namely the running time (or cost) of the algorithm and the size of the forbidden query set obtained using this algorithm.

We have seen that the proportion of queries that must be included in F is greater or equal to $\frac{1}{k+1}$. Therefore step (3) is repeated cm times, where m is the total number of queries and c is greater or equal to $\frac{1}{k+1}$. For each iteration in step (3)

a) is executed once

b) is executed at most N-k times

c) is executed k+1 times for each system found in (b)

So at most O(k(N-k)) counts will be changed when we add Q_5 to the set F. Therefore the cost of Algorithm 3.5 is O(cmk(N-k)) which is approximatly $O(N^k)$ when k is small.

When k is large, $0 < k < \frac{N}{2}$, the cost is $\frac{cN!}{(N-k)!}$, where c is a constant. However it is easy to see that in all cases the cost is $O(N^k)$. Although the cost of this algorithm is much smaller than that of Algorithm 3.3, for moderately large values of N, the cost is still high; however we have a polynomial time algorithm whose performance remains comparable or better than the performance of Algorithm 3.3,

in the sense that the forbidden query set size is about the same.

Table 3.10 contains the size of the forbidden query set, found by Algorithm 3.5, for different choices of N and k.

TABLE 3.10

N	k=3	k=4	k=5	k=6
4	1			
5	3	1		
6	7	3	1	
7	14	7	4	1
8	23	14	11	4
9	34	32	25	14
10	52	56	53	38
11	72	93	110	87
12	94	143	206	172
13	128	230	334	346
14	165	329	557	574
15	211	441	872	1055
16	270	540	1257	1687
17	328	818	1815	2648
18	397	1084	2607	4208
19	480	1448	3621	6756
20	569	1740	4929	9969

When N gets large we see that the forbidden query set grows quickly. For example, when N=32 and k=4 the forbidden query set contains 13,100 queries, which is approximately 36 percent of the total query set. The set F, obtained by Algorithm 3.5 contains at least one query from each system. Therefore, if we return false answers to the queries in the set F, no simultaneous linear system when solved, will allow a user to infer the true value of a record.

The algorithm below outlines the method by which a query will be answered.

ALGORITHM 3.6

1) Find the true answer to the query q_i

2) Search the set F for query q_i

3) If q_i in F

then response=true answer(1+error)

else response=true answer

where error is a predetermined quantity.

We have simulated a data base that uses such a querying system in order to protect personal information. This data base has 20 records and the query size has been fixed at k=4. Using Algorithm 3.5, we constructed the set F containing 1740 of the $\binom{20}{4}=4845$ possible queries. The records in the data base are listed, along with the value of their data field, in Table 3.11.

The results shown in Table 3.12 were obtained by asking a series of queries and then solving the systems formed by these queries. The error in the responses are determined by the querying system pseudo-randomly, subject to the constraint that the error introduced is less than 12.50 percent. The query asks for the average value of the data fields of the records specified.

TABLE 3.11

REC. ID.	VALUE	REC. ID.	VALUE
N1	106.0	N11	162.0
N2	198.0	N12	163.0
N3	156.0	N13	113.0
N4	111.0	N14	111.0
N5	114.0	N15	150.0
N6	189.0	N16	190.0
N7	178.0	N17	143.0
N8	195.0	N18	171.0
N9	191.0	N19	139.0
N10	136.0	N20	132.0

TABLE 3.12

	QUERY	RESPONSE	RESPONSE ERROR	RECORD	VALUE INFERRED	INF. ERROR
1.	Q(2,4,6,8)	193.48	11.68	N2	218.23	10.22
	Q(2,4,6,10)	158.50	0.00	N4	131.23	18.23
	Q(2,4,8,10)	160.00	0.00	N6	209.23	10.71
	Q(2,6,8,10)	179.50	0.00	N8	215.23	10.38
	Q(4,6,8,10)	157.75	0.00	N10	75.30	44.63
2.	Q(11,13,15,17)	142.00	0.00	N11	124.14	23.37
	Q(11,13,15,19)	141.00	0.00	N13	142.77	26.34
	Q(11,13,17,19)	152.11	9.24	N15	128.32	14.45
	Q(11,15,17,19)	148.50	0.00	N17	172.77	20.82
	Q(13,15,17,19)	153.16	12.41	N19	168.77	21.41
3.	Q(3,7,16,19)	165.75	0.00	N3	124.70	20.07
	Q(3,7,16,20)	149.86	8.62	N7	146.70	15.59
	Q(3,7,19,20)	134.09	11.34	N16	227.33	19.65
	Q(3,16,19,20)	154.25	0.00	N19	164.28	18.18
	Q(7,16,19,20)	159.75	0.00	N20	100.70	23.71
4.	Q(1,4,5,13)	111.00	0.00	N1	108.86	2.70
	Q(1,4,5,14)	110.50	0.00	N4	156.32	40.83
	Q(1,4,13,14)	123.73	12.22	N5	269.96	44.78
	Q(1,5,13,14)	100.38	9.56	N13	115.86	2.53
	Q(4,5,13,14)	112.25	0.00	N14	113.86	2.58

This method, as compared to the proposal in Section 3.1,

returns true answers to a majority of the queries. Even though the statistical information of the data base, available to the user, is more precise, the amount of error in the inferred value is larger in a majority of the cases.

We will now analyze the behavior of the error introduced in such a system in order to get an upper bound on the inferred value errors.

In this system, k keys are permitted in a query. Then the simultaneous linear system is of the form

$$AX=r$$

where $A=(a_{ij})$ $1\leq i\leq k+1$ and $1\leq j\leq k+1$. For this problem a_{ij} is either "1" or "0" and each row of A has k 1's and one "0".

Let $\|A\|=\max \sum_{j=1}^{k+1} |a_{ij}|=k$, for $1\leq i\leq k+1$, be the infinity norm of A. Similarly let $\|X\|=\max|X_i|$, for $1\leq i\leq k+1$, be the infinity norm of the unknown vector X.

Since $|\det(A)|=k>0$, the inverse A^{-1} exist. Defining the norm of A^{-1} in the same manner we can easily show that

$$\|A^{-1}\|= \frac{2k-1}{k}$$

We can now define the condition number of A as

$$C(A)= \|A\|\cdot\|A^{-1}\|$$

$$= k\cdot \frac{2k-1}{k} = (2k-1)$$

Uncertainties are introduced only in the response vector r. Let $r'=r+\delta r$ where δr is the uncertainty vector. By

controlling δr we want to estimate the amount of error in the solution vector X (which is what the user infers by solving the system). Let $X' = X + \delta X$ be the solution vector. Therefore the linear system

$$AX = r \tag{3.10}$$

becomes

$$A(X+\delta X) = r + \delta r \tag{3.11}$$

from equation (3.10) we get $X = A^{-1}r$ and

from equation (3.11) $X + \delta X = A^{-1}(r+\delta r)$

therefore, the uncertainty solution vector is

$$\delta X = A^{-1}(r+\delta r) - X$$

$$\delta X = A^{-1}(r+\delta r) - A^{-1}r$$

$$\delta X = A^{-1}\delta r$$

Taking the infinity norm of both sides we have

$$\|\delta X\| \leq \|A^{-1}\| \cdot \|\delta r\| \tag{3.12}$$

From equations (3.10), (3.11) and (3.12) we see that the relative error in the solution vector is given by

$$\frac{\|\delta X\|}{\|X\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\delta r\|}{\|r\|}$$

$$\frac{\|\delta X\|}{\|X\|} \leq C(A) \cdot \frac{\|\delta r\|}{\|r\|}$$

where $C(A) = 2k-1$

Therefore the relative uncertainty in the solution vector is bounded by the product of the condition number of A and the relative uncertainty in the response vector r. By finding the expected value of the relative uncertainty in the response vector, we will have an upper bound on the expected value of the relative error in the solution vector.

Let the ratio $\frac{\delta r_i}{r_i}$, $i=(i_1,\ldots,i_s)$ be independent and uniform in $(-1,1)$, where $(i_1,\ldots,i_s)$ is the set of indices for which $\delta r_i \neq 0$. For example, in Table 3.12 s=1 in the first system and s=2 in the second system. In other words, s is the number of queries in the system that have false answers. The norm of δr is defined as

$$\|\delta r\| = \max\{|\delta r_i|, i=(i_1,\ldots,i_s)\}.$$

Therefore

$$\frac{\|\delta r\|}{\|r\|} = \frac{\max\{|\delta r_i|, i=(i_1,\ldots,i_s)\}}{\max\{|r_i|, i=1,k+1\}}$$

If $\max(|r_i|)$ is $|r_j|$ and j is in $(i_1,\ldots,i_s)$ then

$$\frac{\|\delta r\|}{\|r\|} = \max\left(\frac{|\delta r_j|}{|r_j|}, j=(i_1,\ldots,i_s)\right)$$

otherwise

$$\frac{\|\delta r\|}{\|r\|} < \max\left(\frac{|\delta r_j|}{|r_j|}, j=(i_1,\ldots,i_s)\right)$$

Thus setting $u_i = \frac{|\delta r_i|}{|r_i|}$ we have

$$\frac{\|\delta r\|}{\|r\|} = \lambda \max(u_i \mid i=(i_1,\ldots,i_s))$$

where $\lambda \leq 1$ and the u_i's are uniform in (0,1), since $\frac{\delta r_i}{r_i}$ is

uniform in (-1,1). Therefore the probability distribution of the error function is given by

$$\Pr[\max(u_i) \leq t] = t^s, \quad 1 \leq i \leq s$$

and its density function is

$$\text{DENSITY} = st^{s-1}, \quad 0 \leq t \leq 1$$

Hence the expected value of the relative error in the response vector is

$$E\left[\frac{\|\delta r\|}{\|r\|}\right] = s \int t^s \, dt = \frac{s}{s+1} \qquad (3.13)$$

We have shown that

$$\frac{\|\delta X\|}{\|X\|} \leq C(A) \cdot \frac{\|\delta r\|}{\|r\|}$$

Using equation (3.13) we write

$$\begin{aligned} E\left[\frac{\|\delta X\|}{\|X\|}\right] &\leq C(A) \cdot E\left[\frac{\|\delta r\|}{\|r\|}\right] \\ &\leq (2k-1) \cdot \left[\frac{s}{s+1}\right] \\ &\leq (2k-1) \cdot \left[1 - \frac{1}{s+1}\right] \end{aligned}$$

where $1 \leq s \leq k+1$

The upper bound on the relative error in the inferred values is

$$(2k-1)\left[1 - \frac{1}{s+1}\right] \cdot \epsilon$$

where ϵ is the maximum allowable error introduced in the response. We multiply by ϵ because, in the analysis, we assumed the u_i's to be uniform in (0,1) while in the method

the u_i's are uniform in $(0,\epsilon)$.

The expression $(2k-1)[1-\frac{1}{s+1}]$ is the expected value of the upper bound of the relative error in the inferred values. Our simulation results show that the actual relative errors in the inferred values lie very close to this bound. This is quite justified and also encouraging for the variance of the upper bound is large thus fixing the actual inferred values to lie close to the expected value (not the true value).

So far we have been discussing an inference control scheme in terms of a small data base. The size of the forbidden query set gets large very fast as the number of records in the data base increases. For example, the Algorithm 3.5 constructs a set F that contains 23,443 queries for a data base with N=35 records and with the size of the queries fixed at k=4. In general we have $c \cdot m$ queries in F, where m is the number of possible queries and $\frac{1}{k+1} \leq c \leq 1$. Therefore, to construct the set F for a data base with N=100 records that allows queries that specify a subset of k=4 records, we would need to include no less than 784,245 queries into the set F.

THEOREM 3.2

The size of the forbidden query set is bounded by

$$\frac{1}{k+1}\binom{N}{k} \leq |F| \leq \binom{N}{k}$$

Proof:

The number of systems in a data base with N records that allows queries involving k records is $S=\binom{N}{k+1}$. The number of queries in such a data base is $m=\binom{N}{k}$. Any query Q_i is contained in exactly N-k systems. Therefore the minimum number of queries needed to cancel all the systems is $\frac{S}{N-k}$. This occurs when every query in F cancels N-k distinct systems. So the lower bound of |F| is

$$\frac{S}{N-k} = \frac{1}{N-k} \cdot \binom{N}{k+1}$$

$$= \frac{1}{N-k} \cdot \frac{N!}{(N-k-1)!(k+1)!}$$

$$= \frac{N!}{(N-k)\cdot(N-k-1)!(k+1)(k)!}$$

$$= \frac{N!}{(N-k)!(k)!(k+1)}$$

$$= \frac{1}{k+1} \cdot \binom{N}{k}$$

Even for relatively small N and k the set F is very large. These forbidden queries will have to be stored and to answer any query, we will have to search the set F to determine if the query is to be answered truthfully or not. Therefore, for medium or large size data bases, this will take too much time and space for it to be really effective. We propose a modified approach that enables us to generalize this scheme for medium and large scale data bases.

The only difference in this modified scheme is that we will construct the set F for a subset of the data base. This approach saves in storage (since we only store the forbidden query set for a subset of the entire data base which has

significantly less queries) and in time.

Algorithm 3.7 outlines this approach.

ALGORITHM 3.7

1) Partition the data base into small groups of t records each (say 20 or 30 records in each group). The criteria for obtaining these partition is independent of the value fields of the records.

2) Construct F for a data base with t records.

3) (This is the step where each query is answered)
   a) Input query $Q(key_1,\ldots,key_k)$
   b) Find the group G_j which is referred to by the most number of keys in the query Q.
   c) For each key not in G_j, replace this key by a key in G_j which has value nearest to the value of the given key.
   d) Call this new query Q' and respond to Q' in the usual way. (as in Algorithm 3.6)
   e) Return the response of Q' to the user.

The following example demonstrates how this algorithm responds to a given query.

Assume the data base has 15 records and allows queries of size k=4. We partition the data base into 3 groups. Therefore, each group contains t=5 records. Let records N1

to N5 be in the first group, N6 to N10 in the second group and N11 to N15 in the third group. The data base is listed in Table 3.13

TABLE 3.13

RECORD ID	VALUE	RECORD ID	VALUE	RECORD ID	VALUE
N1	120	N6	115	N11	75
N2	95	N7	190	N12	68
N3	70	N8	98	N13	147
N4	142	N9	145	N14	93
N5	125	N10	132	N15	123

In constructing F for a data base with 5 records we assume the record identifiers to be (1,2,3,4,5). Therefore

$$F=\{(123),(145),(234)\}.$$

The records in these queries refer to the position of the records in each group of the data base in Table 3.13. For instance the query Q(N6,N9,N10) is a forbidden query. Therefore when a user asks such a query, the response will be perturbed. For queries involving records which are all in the same group the scheme is essentially the same as in Algorithm 3.6. For queries with records belonging to different groups the scheme described in step (3) of Algorithm 3.7 is used.

The query Q(N3,N11,N15) becomes Q'(N11,N12,N15) because the third group has 2 records and the first group has one record that is in the query Q. Therefore the record N3 of group 1 is matched to the best fit record in group 3. Q'(N11,N12,N15) corresponds to the query q(125) since N11,

N12 and N15 are the first, second and fifth record of the third group. We now search F for q(125) and we return the true answer of Q (N11,N12,N15) since q(125) is not in F.

The response to the query Q(N3,N11,N15) will not be true if the values of N3 and N12 are not equal, however, with groups that 20 or 30 records, the best fit approach will keep the responses close to the true values.

Table 3.14 contains the results of this method, implemented for a sample data base with 100 records and k fixed at 4. We partitioned the data base into 5 groups of 20 records each. The set F was constructed using Algorithm 3.5 and contains 1740 queries. The queries ask for the average of the data field of the specified records.

ERROR ANALYSIS

In this scheme, error is introduced into the system not only by perturbing the responses to queries in the forbidden query set but also by masking keys from one group to another. Therefore the system

$$AX=r \qquad (3.14)$$

becomes

$$(A+\delta A)(X+\delta X)=r+\delta r \qquad (3.15)$$

We want to determine the relative error in the solution vector X. Solving for X in equation (3.14) and for $(X+\delta X)$ in equation (3.15) we get

TABLE 3.14

	QUERY	RESPONSE	RESPONSE ERROR	RECORD	VALUE INFERRED	INF. ERROR
1.	Q(1,21,31,41)	126.00	3.07	N1	92.99	12.27
	Q(1,21,31,51)	131.50	1.54	N21	113.99	8.81
	Q(1,21,41,51)	103.74	13.91	N31	204.02	38.79
	Q(1,31,41,51)	126.25	0.20	N41	92.99	16.22
	Q(21,31,41,51)	131.50	0.57	N51	114.99	17.86
2.	Q(1,21,41,51)	103.74	13.91	N1	41.79	60.57
	Q(1,21,41,81)	130.00	1.76	N21	165.12	32.10
	Q(1,21,51,81)	137.00	1.48	N41	90.03	18.89
	Q(1,41,51,81)	118.23	10.09	N51	118.03	15.69
	Q(21,41,51,81)	149.06	9.40	N81	223.06	31.99
3.	Q(20,40,60,80)	143.25	1.38	N20	111.14	15.80
	Q(20,40,60,100)	109.39	13.35	N40	101.14	18.43
	Q(20,40,80,100)	137.50	2.14	N60	124.14	7.36
	Q(20,60,80,100)	143.25	0.17	N80	236.57	23.86
	Q(40,60,80,100)	140.75	0.18	N100	101.14	12.05
4.	Q(1,2,40,49)	136.75	2.43	N1	122.43	15.50
	Q(1,2,40,67)	153.50	1.32	N2	209.43	5.77
	Q(1,2,49,67)	160.93	9.48	N40	92.70	25.24
	Q(1,40,49,67)	131.75	2.53	N49	122.43	15.50
	Q(2,40,49,67)	153.50	1.32	N67	189.43	6.42

$$X=A^{-1}r$$

$$X+\delta X=(A+\delta A)^{-1}(r+\delta r)$$

substracting the first from the second we get

$$\delta X=((A+\delta A)^{-1}-A^{-1})r+(A+\delta A)^{-1}\delta r \qquad (3.16)$$

Using the identity $B^{-1}-A^{-1}=A^{-1}(A-B)B^{-1}$ with $B=A+\delta A$ we can transform equation (3.16) into

$$\begin{aligned}\delta X&=-A^{-1}\delta A(A+\delta A)^{-1}r+(A+\delta A)^{-1}\delta r\\&=-A^{-1}\delta A(A+\delta A)^{-1}(r+\delta r)+A^{-1}\delta r\\&=-A^{-1}\delta A(X+\delta X)+A^{-1}\delta r\end{aligned}$$

From this we get

$$\|\delta X\| \leq \|A^{-1}\| \cdot \|\delta A\| \cdot \|X+\delta X\| + \|A^{-1}\| \cdot \|\delta r\|$$
$$\leq \frac{C(A)}{\|A\|} \cdot (\|\delta A\| \cdot \|X+\delta X\| + \|\delta r\|)$$

therefore

$$\frac{\|\delta X\|}{\|X+\delta X\|} \leq \frac{C(A)}{\|A\|} (\|\delta A\| + \frac{\|\delta r\|}{\|X+\delta X\|}) \qquad (3.17)$$

from equation (3.15) we get

$$\frac{1}{\|X+\delta X\|} \leq \frac{\|A+\delta A\|}{\|r+\delta r\|}$$

If we substitute this result in the right hand side of equation (3.17) we have

$$\frac{\|\delta X\|}{\|X+\delta X\|} \leq \frac{C(A)}{\|A\|} (\|\delta A\| + \frac{\|\delta r\| \cdot \|A+\delta A\|}{\|r+\delta r\|})$$
$$\leq C(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta r\|}{\|r+\delta r\|} (1 + \frac{\|\delta A\|}{\|A\|}) \right)$$

This gives a bound for the uncertainty in the solution vector X, taken relative to X+δX, in terms of the relative uncertainties introduced in A and r. We want the expected value of this bound. As we have seen earlier $\|A\|=k$ and $C(A)=2k-1$; we now need the expected values of $\|\delta A\|$ and $\frac{\|\delta r\|}{\|r+\delta r\|}$.

Here $\|\delta A\|$ is not strictly random in the sense that this uncertainty is introduced using "best match" keys. In each query, j out of k keys are introduced this way. In the matrix A+δA, there exist j entries of the form $(1+\epsilon_i)$ and k-j

entries of the form "1" and one position has a "0".

Let s be the number of queries in the system that have uncertainties. Therefore s=0,k,k+1, where s=0 corresponds to the case $A(X+\delta X)=r+\delta r$.

Assume that the ϵ_i's are uniformly distributed and let S_p, $1\leq p\leq s$, be the sum of the j ϵ_i's in row_p. From this we get the probability distribution function

$$U_j(x) = Pr[S_j \leq x]$$

$$= \frac{1}{j!}\sum_{v=0}^{j}(-1)^v\binom{j}{v}(x-v)_+^j$$

where

$$x_+^j = \begin{cases} 0 & x<0 \\ x^j & x\geq 0 \end{cases}$$

The density function is therefore

$$u_{p+1}(x)=[U_p(x)-U_p(x-1)]$$

For j=2 we have

$$u_2(x) = \begin{cases} x & 0\leq x\leq 1 \\ 2-x & 1\leq x\leq 2 \end{cases}$$

The mean of S_2 is therefore

$$\text{Mean} = \int x u_2(x)\,dx$$

$$= \int x^2\,dx + \int (2x-x^2)\,dx$$

$$= \frac{1}{3} + (3 - \frac{1}{3}(8-1))$$

$$= 1$$

For j=3 the density function is

$$u_3 = \begin{cases} \frac{1}{2}x^2 & 0 \le x \le 1 \\ \frac{1}{2}[-2x^2+6x-3] & 1 \le x \le 2 \\ \frac{1}{2}[x^2-6x+9] & 2 \le x \le 3 \end{cases}$$

and the mean is

$$\text{Mean} = \int x u_3(x)\,dx = \frac{3}{2}$$

By computing the mean for different values of j we see that the mean of $S_j = \frac{j}{2}$, $j \ge 1$. They have the same distribution and let us assume that they are independent.

Set $T = \max(S_1, \ldots, S_s)$

The probability distribution function of T is

$$\Pr[T \le t] = \Pr[S_1 \le t, \ldots, S_s \le t]$$

$$= \prod_{i=1}^{s} \Pr[S_i \le t]$$

$$= [U_j(t)]^s$$

and the density function is $s[U_j(t)]^{s-1}u_j(t)$, so the expected value (mean) of $\|\delta A\|$ is

$$E[\|\delta A\|] = \int st[U_j(t)]^{s-1}u_j(t)\,dt$$

where s=0,k,k+1, s=0 corresponds to the case where all the queries in the system have records belonging to the same group and we have assumed that S_i (i=1,k+1) have the same distribution therefore we can assume that each query has some uncertainty introduced in it. Therefore we need only consider s=k+1. With this the expected value of the norm of δA becomes

$$E[\|\delta A\|] = \int (k+1)t[U_j(t)]^k u_j(t)dt$$

Using integration by parts we get

$$E[\|\delta A\|] = j - \int [U_j]^{k+1} dt$$

where

$$U_j(t) = \frac{1}{j!} \sum_{i=0}^{j} (-1)^j \binom{j}{i} (t-i)_+^j$$

Note that $U_j(0)=0$ and $U_j(j)=1$.

Since it is very difficult to get a close form for j>1, Table 3.15 shows the value of the expected value of the norm of δA using the "cautious adaptive Romberg extrapolation" method to evaluate the integral $\int [U_j]^{k+1} dt$.

TABLE 3.15

k \ j=	1	2	3	4	5	6	7
3	0.80	1.42					
4	0.83	1.48	2.08				
5	0.86	1.52	2.13	2.73			
6	0.88	1.55	2.17	2.78	3.37		
7	0.89	1.58	2.21	2.82	3.42	4.00	
8	0.90	1.60	2.24	2.85	3.46	4.05	4.63

Note that j goes from 1 to k-1, since j is the number of keys in each query that have uncertainty, and s has been fixed at k+1 because we assume that each query has at least 1 key that has uncertainty.

We now need the expected value of

$$\frac{\|\delta r\|}{\|r+\delta r\|}$$

Let $X=\|\delta r\|$ and $Y=\|r+\delta r\|$. We want $E[Z]$ where $Z=\frac{X}{Y}$.

$X=\max\{|\delta r_1|,\ldots,|\delta r_s|\}$ and assume that $\frac{|\delta r_i|}{r_i}$ is uniform in $(0,1)$. So $\|\delta r_i\|=ur_i$ where u is uniform in $(0,1)$. Without loss of generality assume that $r_1<r_2<\ldots<r_s$.

Consider the case s=2.

$$X_1=|\delta r_1|=ur_1 \text{ uniform in } (0,r_1)$$

$$X_2=|\delta r_2|=ur_2 \text{ uniform in } (0,r_2)$$

The probability distribution function is

$$\Pr[\max(X_1,X_2)<t] = \Pr[X_1<t, X_2<t] \quad \text{if } t<r_1$$

$$= \Pr[X_2<t] \quad \text{if } r_1<t<r_2$$

therefore

$$\Pr[\max(X_1,X_2)<t] = \Pr[u<\frac{t}{r_1}]\ \Pr[u<\frac{t}{r_2}] \quad \text{if } t<r_1$$

$$= \Pr[X_2<t] \quad \text{if } r_1<t<r_2$$

so

$$\Pr[\max(X_1,X_2)<t] = \frac{t^2}{r_1r_2} \quad 0\le t\le r_1$$

$$= \frac{t}{r_2} \quad r_1\le t\le r_2$$

and the density function is

$$f(t)=\begin{cases}\dfrac{2t}{r_1 r_2} & 0\le t\le r_1\\ \\ \dfrac{1}{r_2} & r_1\le t\le r_2\end{cases}$$

Let $a_j=\left(\prod_{i=j}^{s} r_i\right)$

i.e. $a_1=r_1r_2\ldots r_s$

$a_2=r_2r_3\ldots r_s$

$a_3=r_3r_4\ldots r_s$

so $a_{j+1}=\dfrac{a_j}{r_j}$

In the general case the density function of $\|\delta r\|$ is given by:

$$f(t)=\begin{cases}\dfrac{st^{s-1}}{a_1} & 0\le t<r_1\\ \\ \dfrac{(s-1)t^{s-2}}{a_2} & r_1\le t<r_2\\ \\ \vdots & \\ \\ \dfrac{1}{r_s} & r_{s-1}\le t\le r_s\end{cases}$$

We now need the density function of $\|r+\delta r\|$. We have

$$\begin{aligned} r_i+\delta r_i &= r_i+v_i r_i\\ &= r_i(1+v_i)\\ &= 2r_i(1-u_i)\end{aligned}$$

where v_i is uniform in $(-1,1)$ and u_i is uniform in $(0,1)$.

Thus $$|r_i+\delta r_i| = r_i+\delta r_i$$
$$= 2r_i(1-u_i)$$

First consider the case s=2.

Let $$Y_1 = 2r_1(1-u_1) = 2r_1 v$$
$$Y_2 = 2r_2(1-u_2) = 2r_2 v$$

where u and v are uniform in (0,1). We want the density function of $Y=(\max(Y_1,Y_2)$. Assume $0<r_1<r_2$, therefore

$$\Pr[Y<t] = \Pr[Y_1<t, Y_2<t] \qquad 0\le t\le 2r_1$$
$$= \Pr[Y_2<t] \qquad 2r_1\le t\le 2r_2$$

so

$$\Pr[Y<t] = \begin{cases} \dfrac{t^2}{2^2 r_1 r_2} & 0\le t\le 2r_1 \\ \dfrac{t}{2r_2} & 2r_1\le t\le 2r_2 \end{cases}$$

From this we get the density function

$$g(t) = \begin{cases} \dfrac{t}{2r_1 r_2} & 0\le t\le 2r_1 \\ \dfrac{1}{2r_2} & 2r_1\le t\le 2r_2 \end{cases}$$

In the general case the density function of $\|r+\delta r\|$ is

$$g_j(t) = \frac{(s-j)\,t^{s-j-1}}{2^{s-j}\cdot a_{j+1}} \qquad 2r_j\le t\le 2r_{j+1}$$

where $0\le j\le s-1$.

Note that the density function f and g are discontinuous at end points of intervals. We want an upper bound for:

$$E\left[\frac{\|\delta r\|}{\|r+\delta r\|}\right] = E\left[\frac{X}{Y}\right]$$

Let $D^2[X] = E[X^2]-(E[X])^2$, therefore

$$E\left[\frac{X}{Y}\right] \leq \frac{E[X]}{E[Y]} + \frac{D[X]}{D[Y]} \qquad (3.18)$$

Let us first compute E[X].

$$E[X] = \int tf(t)dt$$

$$= \sum_{j=0}^{s-1} \int \frac{t(s-j)t^{s-j-1}}{a_{j+1}}dt$$

$$= \sum_{j=0}^{s-1} \frac{(s-j)}{a_{j+1}} \left[\frac{t^{s-j+1}}{s-j+1}\right]_{r_j}^{r_{j+1}}$$

$$= \sum_{j=0}^{s-1} \left(\frac{s-j}{s-j+1}\right) \frac{1}{a_j} [r_{j+1}^{s-j+1} - r_j^{s-j+1}]$$

Now find a close form for E[Y]

$$E[Y] = \int tg(t)dt$$

$$= \sum_{j=0}^{s-1} \int \frac{t(s-j)t^{s-j-1}}{2^{s-j}\cdot a_{j+1}}dt$$

$$= \sum_{j=0}^{s-1} \frac{s-j}{2^{s-j}\cdot a_{j+1}} \left[\frac{t^{s-j+1}}{s-j+1}\right]_{2r_j}^{2r_{j+1}}$$

$$= \sum_{j=0}^{s-1} \left(\frac{s-j}{s-j+1}\right) \left(\frac{1}{2^{s-j}\cdot a_{j+1}}\right)$$

$$\cdot [2^{s-j+1}\cdot r_{j+1}^{s-j+1} - 2^{s-j+1}\cdot r_j^{s-j+1}]$$

so

$$E[Y] = 2E[X]$$

Now compute $E[X^2]$ and $E[Y^2]$.

$$E[X^2] = \int t^2 f(t)\,dt$$

$$= \sum_{j=0}^{s-1} \int t^2 \frac{(s-j)t^{s-j+1}}{a_{j+1}}\,dt$$

$$= \sum_{j=0}^{s-1} \frac{s-j}{a_{j+1}} \left[\frac{t^{s-j+2}}{s-j+2} \right]_{r_j}^{r_{j+1}}$$

$$= \sum_{j=0}^{s-1} \frac{s-j}{s-j+2} \cdot \frac{1}{a_{j+1}} \left[r_{j+1}^{s-j+2} - r_j^{s-j+2}\right]$$

$$E[Y^2] = \int t^2 g(t)\,dt$$

$$= \sum_{j=0}^{s-1} \int t^2 \frac{(s-j)t^{s-j+1}}{2^{s-j} \cdot a_{j+1}}\,dt$$

$$= \sum_{j=0}^{s-1} \frac{s-j}{s-j+2} \cdot \frac{1}{a_{j+1} \cdot 2^{s-j}} \left[s^{s-j+2} \cdot r_{j+1}^{s-j+2} - 2^{s-j+2} \cdot r_j^{s-j+2}\right]$$

$$= 4 \sum_{j=0}^{s-1} \frac{s-j}{s-j+2} \cdot \frac{1}{a_{j+1}} \left[r_{j+1}^{s-j+2} - r_j^{s-j+2}\right]$$

Therefore $E[Y^2] = 4E[X^2]$. Substituting these values into equation (3.18) we have

$$E\left[\frac{X}{Y}\right] \leq \frac{E[X]}{2E[X]} + \frac{D[X]}{D[Y]}$$

where $D[X] = \sqrt{(E[X^2] - (E[X])^2)}$

and $D[Y] = \sqrt{(E[Y^2] - (E[Y])^2)}$

$$= \sqrt{(4E[X^2]-(2E[X])^2)}$$

$$= \sqrt{(4E[X^2]-4(E[X])^2)}$$

$$= 2\sqrt{(E[X^2]-(E[X])^2)}$$

$$= 2D[X]$$

so

$$E[\frac{X}{Y}] \leq \frac{E[X]}{2E[X]} + \frac{D[X]}{2D[X]}$$

$$\leq \frac{1}{2} + \frac{1}{2}$$

$$\leq 1$$

Therefore,

$$E[\frac{\|\delta r\|}{\|r+\delta r\|} \leq 1$$

Using these 2 results we have

$$E[\frac{\|\delta X\|}{\|X+\delta X\|} \leq (2k-1)[\frac{E[\|\delta A\|]}{k} + E[\frac{\|\delta r\|}{\|r+\delta r\|}](1+\frac{E[\|\delta A\|]}{k})]$$

$$\leq (2k-1)[\frac{E[\|\delta A\|]}{k} + 1 + \frac{E[\|\delta A\|]}{k}]$$

$$\leq (2k-1)[1 + \frac{2}{k}(E[\|\delta A\|])]$$

The expected value $E[\|\delta A\|]$ is found in Table 3.15 for different values of j and k. Since we assumed that the relative error in the response is uniform in (0,1), we must multiply the results by ϵ, where ϵ is the maximum relative error introduced in the response. Table 3.16 contains the expected value of the upper bound of the relative error in

the solution vector X, where ϵ=0.10; i.e. the maximum allowed error in the response is 10 percent.

TABLE 3.16

k \ j=	1	2	3	4	5	6	7
3	0.77	0.98					
4	0.99	1.22	1.43				
5	1.21	1.45	1.67	1.88			
6	1.42	1.67	1.90	2.12	2.34		
7	1.63	1.89	2.12	2.35	2.57	2.79	
8	1.84	2.10	2.34	2.57	2.80	3.02	3.24

This result and the result for an expected value of the relative error in the system where A is exact demonstrates that the error in the inferred value can be large. In the test results of Tables 3.12 and 3.14 we see that this is true. In Tables 3.12 and 3.14 the relative errors vary from approximately 2 percent up to approximately 61 percent. This means that for any single inferred value it is almost impossible to guess the range of uncertainty, and hence almost impossible to compromise any single data value by asking queries that allow a user to construct these (k+1,k+1) systems of equations.

Therefore, such a method effectively secures a data base against such threats. By this we mean that if queries with the maximum overlap are allowed, and these queries are exclusively used, then the data base is secure. However, as seen in Chapter 2, a user can still compromise the data base by using queries that overlap in fewer places.

In Chapter 2 we discussed the results of [15] where the function $S(N,k,r,\ell)$ defines the smallest number of queries needed to compromise a data base with N records that allows queries of size k and allows no more than r elements in common to each query. Here "ℓ" is the number of record values known by the user.

The security method described in this section guarantees security for r=k-1. However it is possible to determine the value of one record by asking a sequence of queries that do not belong to the forbidden query set and which overlap in at most r=1 positions.

To demonstrate, this, consider a data base that has N=7 records and query size k=3. A forbidden query set for such a data base is

$$F=\{(123),(127),(136),(145),(167),(236),(245),(246),$$
$$(256),(345),(347),(357),(456)\}$$

The queries in this set cancel all 35 possible systems. If a user wants to determine the value of the data field of record N7 alone (say one record at a time) the following sequence of queries can be asked in order to determine the value of the data field of N7.

Q_1=SUM(N1,N2,N4)

Q_2=SUM(N3,N5,N6)

Q_3=SUM(N1,N3,N7)

Q_4=SUM(N2,N5,N7)

Q_5=SUM(N4,N6,N7)

The value of N7 can computed as

$$\text{VALUE}(N7) = \frac{1}{3}(Q_3+Q_4+Q_5-(Q_1+Q_2))$$

Since the queries Q_1 to Q_5 are not in F, the inferred value of N7 will be exact, hence the data base will be compromised. Therefore the method proposed in this section does not secure the data from attacks of this nature. There are 60 possible query sequences of the above type that will solve for N7. Out of these 60 sequences 12 of them will solve for the exact value of N7 because, as in the above example, none of the five queries are in F. In order to secure the data base against such compromise we will have to add queries to F. Similarly, there are 60 distinct query sequences to solve for each of the remaining 6 records. Therefore, we will have to add even more queries to F to assure security. In order to secure for all possibilities of compromise the set F will contain almost all the queries.

Algorithm 3.5 constructs F such that F contains at least one query from every possible simultaneous system, that has queries with k-1 records in common. Let the number of such systems be X_{k-1}. We have seen above that there exist 60

systems with a maximum of 1 element in common that solves for N7. Therefore 7·60=420 such systems exist that can solve for 1 record at a time. Call this number X_1. Similarly, systems with queries that overlap in j out of k positions can exist. Therefore, the total number of systems (sequence of queries) that can solve for at least one record is

$$X=\sum_{i=1}^{k-1} X_i$$

where some of the $X_i=0$.

In order to ensure security, the set F must contain at least one query from each of the X sequences of queries. As we have seen for N=7 and k=3, there exist 455 possible sequences of queries that can solve for at least one data field value.

$$X_{k-1} = X_2 = \binom{N}{k+1} = 35$$

$$X_{k-2} = X_1 = 7 \cdot 60 = 420$$

$$X = \sum_{i=1}^{k-1} X_i = 35+420 = 455$$

A data base with N=7 and k=3 has 35 possible queries. In order to secure the 455 systems all of these almost all of these queries must be in F.

Therefore for protecting a data base in which the overlap size is not fixed at k-1, one must lie almost at every query. This becomes more obvious if we study the relationship

between a forbidden query set and the records in the data base.

Let $F=\{q_1,\ldots,q_t\}$ be a forbidden query set extracted by Algorithm 3.5. For each record R in the data base we associate $Q(R_j)=\{q_i \mid q_i \text{ covers } R_j\}$. Clearly $\{Q(R_j)\}$ is the set of inverted lists formed on the forbidden query set and $Q(R_j)$ contains precisely all the queries that must be forbidden in order to protect R_j. So we call $Q(R_j)$ the "best cover" at R_j. When the overlap size of a sequence of queries lies between 1 and k-1, a particular record R_j can be solved by any one sequence out of a large possible number of sequences of queries, as has been demonstrated by our example. In an on-line dialogue system it is rather difficult to know what the user is after, i.e. which record the user wants to infer. Because of this inherent difficulty, one must lie on all queries. However in a "batch" mode where the user is forced to ask several queries before the system starts responding, our investigation naturally suggest a realistic strategy.

When several queries, not all of them being disjoint, are input to the system, the system can recognize the record which occurs in "several" queries and then use the total cover for this record to perturb the response for the queries of the user. However a user can spend several days or employ another computer to formulate queries and try to compromise a subset of the data base. Since this subset can be any

arbitrary subset, there is no effective way except to lie on all the queries. What remains important is the strategy of lying which should not distort the statistics too much and should be able to protect any individual data. Hence our investigation in this chapter is probably the only significant strategy of effective inference control in a statistical data base.

CHAPTER 4

PROTECTION OF ATTRIBUTE SPECIFIED DATA BASES

Many studies [6,8,12,14,16,17,18,22,23,24] concerning statistical data bases that permit attribute specified queries have shown that protecting personal information against disclosure to unauthorized users is very difficult.

The ability to isolate an individual record of the data base by correlating the available statistical summaries enables a user to determine personal information about that individual. If we are to protect the data base against such inference we must eliminate the possibility of the user to isolate a record.

Random sampling and partitioning of a data base are quite effective in preventing the user to isolate a record. In random sampling, a query returns a statistical summary pertaining to a random sample of the records in the data base. In a large data base this summary may be statistically correct, however the statistics may not be significant in small or medium size data bases. Therefore, the strategy of random sampling on small or medium size data bases, instead of protecting them against compromise, will destroy the usefulness of the statistical summaries.

In partitioning, the records of a data base are stored as groups of records. A user may ask information about any set of groups, but is never allowed to access subsets of records within the same group. At best a user may isolate a group but never a single record. As seen in Chapter 2, this approach puts two severe practical limitations in dynamic data bases. First, the grouping of records may distort the statistical information of the data base and secondly the forming and reforming of groups during insertion and deletion can lead to costly bookkeeping.

Another method to prevent a user from isolating a record is not to answer any query that applies to fewer than m records, where m is called the response level. In [14,17], the authors show that this security safeguard is not effective. The existence of trackers, discussed in Chapter 2, shows that if we are to restrict a query Q in order to ensure security, we must restrict other queries from which Q can be computed. This results in a chain reaction where even more queries will have to be restricted in order to prevent a user from computing the response to these queries. In [6,18,23,24] the authors show that even if a majority of the queries are unauthorized, compromise is still possible.

In this chapter, we will propose a security method where a range is given to a query instead of the true value. In attribute specified queries there are 2 types of statistics

that are returned: the COUNT and the VALUE. A count query is a query that asks "How many records in the data base statisfy the query?" and a value query is a query that asks "What is P of the records that satisfy the query?", where P is a query predicate such as sum, average, median, etc. In our proposal we will give a range for the count queries and the true value will be given for the value queries. We will attempt to show that without the true count it is not possible to isolate a record.

With such a security method the usefulness of the statistical summaries is not questioned since exact answers are given to all queries that pertain to the data field values. By giving a range [a,b] as a response to count queries we are not misleading the user since the true count lies within that range.

4.1 RANGE RESPONSES TO COUNT QUERIES

In this model the data base has N records. Statistics can be obtained through two types of queries: counts and averages. A query is given in terms of a formula $C(V_1V_2V_3\ldots)$ where V_i is one of the attribute values of $attribute_i$. If we are not interrested in $attribute_j$, we set V_j=*. A record matches the formula C if it agrees with C in every position except the positions that have a * in C. The set of records whose attribute values match the formula C is

called the query set X_c of C.

When the count, or the size of the query set, is requested the query will be COUNT(C), and when the average values of the data fields of the records in X_c is requested the query will be AVE(C).

The responses to these types of queries will be:

$$AVE(C) = \sum_{i \in X_c} \frac{V_i}{N_c}$$

where V_i is the value of the data field of record$_i$ and $N_c = |X_c|$.

and

$$COUNT(C) = [a,b] \text{ where } a \leq |X_c| \leq b$$

The range [a,b] has a fixed size of length S and also the boundaries of the intervals are prefixed as follows:

$$[0,S-1],[S,2S-1],[2S,3S-1],\ldots,[(i-1)S,iS-1],\ldots$$

For any query COUNT(C), if $N_c = |X_c|$ lies within the i-th interval the response to the query is [a,b] where a=(i-1)S, and b=iS-1, where S is the length of the interval, i.e. S=b-a+1. There is no overlap between the intervals, this assures that a query will always return the same range.

All count queries are permitted, since the response [0,S-1] is given for any query that involves fewer than S records. Therefore a user can not isolate an individual by

asking queries with small counts. This has the same effect as not allowing queries that involve fewer than S records. Any query that requests statistics about the data fields of the records must match at least S records. Therefore the query AVE(C) is undefined if $|X_C| \leq S-1$. This restriction is identical to the restriction of the data base models that impose a response level to the queries as in [6,14,17,18,23,24]. There are two major reasons why a query, of the type AVE(C), that requests statistics about the data fields, where $0 \leq |X_C| \leq S$, is undefined. The first is that if a user asks the queries COUNT(C) for which the response is in the interval [0,S-1] and AVE(C) for which the true value is R. If R=0, then the user knows that COUNT(C)=0, and if $R \neq 0$ than the user knows that COUNT(C) lies in the range [1,S-1]. The other reason is that if a user knows, from external sources, that the formula C identifies an individual I uniquely, then by allowing a response to AVE(C), the value of the data field of individual I is given.

Using the above data base model and response strategy, we will now show that, when ranges are given for COUNT queries, we can not isolate any individual record. In other words we will not be able to determine, with certainty, that the formula C uniquely identifies an individual. We will give necessary and sufficent conditions under which the range can be reduced and derive the exact extent by which such reductions can occur. We will also show when we can reduce

the range to a single point; this is the case when the true count of a query is inferred.

The results will show that in the majority of the cases the range can not be reduced, that in a small number of cases the range can be reduced and that very rarely can we reduce the range to a single point.

We will first analyze the case when all the attributes are binary (each attribute has one of two possible values), then we will generalize this in order to get the results for attributes of the n-ary type. The results show that it is very difficult (if not impossible) to isolate an individual record, therefore any information about that individual is not readily available.

Consider the case when each attribute has one of two values: 0 or 1. Then the following relations hold true:

$$\begin{aligned} COUNT(1**) &= COUNT(11*)+COUNT(10*) \\ &= COUNT(1*1)+COUNT(1*0) \\ &= COUNT(110)+COUNT(111) \\ &\quad +COUNT(100)+COUNT(101) \end{aligned} \tag{4.1}$$

Consider the data base in Example 4.1. If a questionner wishes to know how many persons in the data base are married, he asks the query: COUNT(*M*) which has the true response 3.

The relation in equation (4.1) states that the total number of married individuals in the data base is the number

EXAMPLE 4.1

NAME	ATTRIBUTE 1 SEX (M/F)	ATTRIBUTE 2 CIVIL STATUS (SIN/MAR)	ATTRIBUTE 3 UNIV. STATUS (PROF/STUD)
N1	M	S	P
N2	M	M	S
N3	F	M	S
N4	F	S	S
N5	M	S	P
N6	F	S	P
N7	F	M	P

of married professors added to the number of married students. Therefore

$$\begin{aligned} COUNT(*M*) &= COUNT(*MP)+COUNT(*MS) \\ &= 1+2 \\ &= 3 \end{aligned}$$

If we use the intevals [0,S-1],[S,2S-1],..., then in some cases the range can be reduced by at most one when we have binary attributes. The next example shows how this reduction is done.

EXAMPLE 4.2

Let S=5 and COUNT(1*)=[25,29]

COUNT(11)=[15,19]

COUNT(10)=[5,9]

Since COUNT(1*)=COUNT(11)+COUNT(10) we have the relation [25,29]=[15,19]+[5,9], where we interpret that there exist

integers $Z \epsilon [25,29]$, $X \epsilon [15,19]$ and $Y \epsilon [5,9]$ such that $Z=X+Y$ holds. There are several possible solutions.

The minimum value of COUNT(1*) is 25, therefore, the sum of COUNT(11)+COUNT(10) must be at least 25. Assume that the true count of COUNT(11) is 19, its maximum value, then COUNT(10) must be at least 25-19=6. Therefore the range [5,9] is reduced in size to the range [6,9] for COUNT(10).

Consider all the possible cases of true counts for the ranges in Example 4.2 which are listed below.

COUNT(1*)	= COUNT(10)	+ COUNT(11)
25	6	19
25	7	18
25	8	17
25	9	16
26	7	19
26	8	18
26	9	17
27	8	19
27	9	18
28	9	19

From this we see that COUNT(1*)=29, COUNT(11)=15 and COUNT(10)=5 are not feasible solutions. Therefore, the ranges can be reduced to

COUNT(1*)= [25,28]

COUNT(11)= [16,19]

COUNT(10)= [6, 9]

The range of each query has been reduced in size by 1. The next example demonstrates a case when a reduction in size

is not possible.

EXAMPLE 4.3

Let COUNT(1*)= [25,29]

COUNT(11)= [15,19]

COUNT(10)= [10,14]

As we can see all the values in these ranges are possible solutions.

We are interested in knowing how many cases are reducible and how many are non-reducible. Also we would like to know how often each case arises.

Let the query COUNT(1*) have the range [(x-1)S,xS-1]. Table 4.1 contains all the possible pairs of ranges for COUNT(10) and COUNT(11).

TABLE 4.1

COUNT(1*)	COUNT(11)	COUNT(10)
[(x-1)S,xS-1]	[0,S-1]	[(x-1)S,xS-1]
	[0,S-1]	[(x-2)S,(x-1)S-1]
	[S,2S-1]	[(x-2)S,(x-1)S-1]
	[S,2S-1]	[(x-3)S,(x-2)S-1]
	.	.
	.	.
	.	.
	[(x-2)S,(x-1)S-1]	[S,2S-1]
	[(x-2)S,(x-1)S-1]	[0,S-1]
	[(x-1)S,xS-1]	[0,S-1]

There is a possibility of 2x-1 cases. Formally we prove as shown below:

Proof:

Note that $x=\frac{a}{S}+1$, so let $P=\frac{a}{S}=x-1$, where P denotes what interval is given as the response. P=0 denotes the interval [0,S-1], P=1 denotes the interval [S,2S-1], in general P denotes the interval [PS,(P+1)S-1]. Let COUNT(1*)=[a,b], COUNT(11)=$[a_1,b_1]$ and COUNT(10)=$[a_0,b_0]$. Therefore, for $[a,b]=[a_1,b_1]+[a_0,b_0]$ to be a feasible case P_0+P_1 must equal P or P-1.

The number of cases in which this occurs is equal to the number of terms in the expression $(T_1+T_2)^Y$ for Y=P and P-1, i.e. for Y=x-2 and x-1 (since P=x-1).

The number of terms in $(T_1+T_v)^Y$, where v=2 (the number of values an attribute can have), is

$$\binom{v+Y-1}{v-1} = \binom{2+x-2-1}{2-1} \quad \text{for } Y=x-2$$

$$= \binom{2+x-1-1}{2-1} \quad \text{for } Y=x-1$$

Therefore the number of possible pairs of responses to COUNT(10) and COUNT(11), when COUNT(1*)=[(x-1)S,xS-1] is

$$\binom{x}{1} + \binom{x-1}{1} = 2x-1$$

This completes the proof.

In Example 4.2, P=5, P_0=1 and P_1=3. We have seen that the ranges in this example are reducible. In Example 4.3 P=5, P_0=2, P_1=3 and this case is not reducible.

We can reduce the range of the queries COUNT(1*), COUNT(10) and COUNT(11) only when $P_0+P_1=P-1$. Therefore the number of reducible cases for binary attributes is x-1.

Of the 2x-1 possible solutions, we have x-1 of them that can be reduced and x of them that can not. Since each case does not have the same probability of occurrence, the number of times that a reducible or non-reducible case occurs is determined by studying the true counts.

Let all the true counts have the same probability of occuring. By this we mean that if COUNT(1*)=10 then the probability of COUNT(10)=4 and COUNT(11)=6 occuring is equal to the probability of COUNT(10)=1 and COUNT(11)=9 occuring.

Consider the case when COUNT(1*)=[10,14]. The possible combinations of exact counts for COUNT(10) and COUNT(11) is given in Example 4.4.

EXAMPLE 4.4

COUNT(1*)	EXACT COUNT(10)		EXACT COUNT(11)		EXACT COUNT(1*)
(10,14)	0	+	10	=	10
	0	+	11	=	11
	0	+	12	=	12
		.			
		.			
		.			
	13	+	1	=	14
	14	+	0	=	14

We assume that each case has the same chance of occuring. From this we have the following theorem.

THEOREM 4.1

If COUNT(1*)=[(x-1)S,xS-1] then there is $\frac{S}{2}(2xS-S+1)$ possible pairs of exact counts for COUNT(10) and COUNT(11).

Proof:

If COUNT(10) lies between 0 and (x-1)S then COUNT(11) has S possible values, therefore a total of ((x-1)S+1)S. If COUNT(10) lies between (x-1)S+1 and xS-1 then the total number of possibilities for COUNT(11) is $\frac{(S-1)S}{2}$. Together, this gives us a total of $\frac{S}{2}(2xS-s+1)$ possible different combinations of true counts for the queries COUNT(10) and COUNT(11).

For each of the possible (x-1) responses in which the range can be reduced there are $\frac{(S-1)S}{2}$ possible combinations of true counts. This is proved simply by inspection. The size or length of the interval is S. When the range is reduced by one the size of each range is S-1. The largest value in the range of COUNT(10) occurs only with the smallest value in the range of COUNT(11); the second largest value of COUNT(10) occurs only with the smallest and second smallest values of the range of COUNT(11); and so on. Therefore the number of possible combinations of exact counts is:

$$\sum_{i=1}^{S-1} i = \frac{(S-1)S}{2}.$$

So the total number of possible combinations of true counts that would cause a reduction in range size is $\frac{(x-1)(S-1)S}{2}$.

Assuming that each case has the same chance of occuring then the probability that any one query's range will be reduced by one is

$$P_x = \frac{(S-1)(x-1)}{S(2x-1)+1}$$

where x varies from 1 to $\frac{N+1}{S}$.

As we can see P_x is always less than 0.5 and as x gets smaller, P_x gets smaller. Remember that x indicates the size of the true count, therefore for small counts, the probability that the range can be reduced is small, when x=1, $P_x=0$.

We will now generalize this for n-ary attributes. Let V indicate the number of values an attribute can have. For example, consider the attribute MARITAL STATUS, this attribute can have one of four values: (married, single, divorced, widowed). Therefore V=4 for this attribute.

Suppose a data base has 4 attributes and the 4-th attribute has t values. Then the following is true:

$$COUNT(a_1a_2a_3*) = COUNT(a_1a_2a_3v_1) + COUNT(a_1a_2a_3v_2) + \ldots + COUNT(a_1a_2a_3v_t) \quad (4.2)$$

where a_1, a_2, a_3 are values for the first three attributes and v_i, i=1,t are the t different values of attribute 4. Consider Example 4.5, where an attribute has three values 0,1 or 2.

EXAMPLE 4.5

Let COUNT(1*)=[25,29]
COUNT(10)= [0,4]
COUNT(11)= [5,9]
COUNT(12)=[10,14]

We see that COUNT(10) can not be "0" or "1" because COUNT(1*) must be at least "25" and when COUNT(10) is "0" or "1" the maximum sum of COUNT(10)+COUNT(11)+COUNT(12) is "23" or "24" respectively. BY the same logic COUNT(11) and COUNT(12) are also reduced. Therefore the ranges for the above queries are reduced to

COUNT(1*)=[25,27]
COUNT(10)= [2,4]
COUNT(11)= [7,9]
COUNT(12)=[12,14]

We see that the range is reduced by 2 or V-1.

Let COUNT($a_1a_2a_3$*)=[(x-1)S,xS-1], and
COUNT($a_1a_2a_3v_i$)=[(x_i-1)S,x_iS-1]

for i=1,V, where V is the number of possible values for the 4-th attribute. We want to know the number of possible sets

of responses for the right hand side of equation (4.2).

Let $COUNT(a_1a_2a_3v_i)=[(x_i-1)S, x_iS-1]$ and let $I_i=(x_i-1)$. Since I_i indicates the interval to which the count belongs, we will call I_i the range indicator and write $COUNT(a_1a_2a_3v_i)=[I_i]$. Now we rewrite equation (4.2) as

$$[I]=\sum_{i=1}^{V}[I_i]$$

The sum in the right hand side can be $I-1, I-2, \ldots, I-V+1$; therefore

$$\sum_{i=1}^{V} I_i = I \text{ or } I-1 \text{ or } \ldots \text{ or } I-V+1 \qquad (4.3)$$

where $0 \leq I_i \leq I$.

Thus we have the following theorem

THEOREM 4.2

The number of possible set of responses for the right hand side of equation (4.2) is the same as the number of solutions for equation (4.3).

Proof:

A general set of responses to equation (4.2) is

$$[IS,(I+1)S-1]=[I_1S,(I_1+1)S-1]+\ldots$$
$$+[I_VS,(I_V+1)S-1] \qquad (4.4)$$

If the minimum in each range of the right hand side of equation (4.4) are the true counts for the queries on the

right hand side of equation (4.2), then the minimum sum of these counts is

$$\text{Minimum sum} = \sum_{i=1}^{V} I_i S$$

while the maximum sum would be

$$\text{Maximum sum} = \left(\sum_{i=1}^{V} I_i S\right) + V(S-1)$$

Therefore, if all these values were possible, the range of the sum of the responses on the right hand side of equation (4.4) is

$$\left[\sum_{i=1}^{V} I_i S \;,\; \left(\sum_{i=1}^{V} I_i S\right)+V(S-1)\right]$$

This range does not intersect $[IS,(I+1)S-1]$, therefore when $\sum_{i=1}^{V} I_i = I+1$ there are no feasible solutions to equation (4.2).

Let $\sum_{i=1}^{V} I_i = I$, then the range of the right hand side of equation (4.4) is $[IS,(IS+V(S-1))]$ which intersects the range on the left hand side, hence at least one solution exists when $\sum_{i=1}^{V} I_i = I$. Similarly, at least one solution exists when $\sum_{i=1}^{V} I_i = I-V+1$ and no solution exists when $\sum_{i=1}^{V} I_i = I-V$. Therefore at least one feasible solution exists when

$$\sum_{i=1}^{V} I_i = I, I-1, \ldots, I-V+1$$

where $I=x-1$ and $I_i \geq 0$.

By finding the number of solutions to equation (4.3) we will determine the number of possible sets of ranges of the left hand side of equation (4.2).

So we want the number of possible sets of t_i, $i=1,V$, $t_i \geq 0$ such that

$$\sum_{i=1}^{V} t_i = Y$$

Let $S(V,Y)$ be the number of sets of non-negative integers of size V, such that the sum of each set equals Y. This problem can be formulated as follows.

Let $P=(t_1+t_2+\ldots+t_V)^Y$, then $S(V,Y)$ is the number of terms in the polynomial P. Consider the case with V=2 and Y=2; where

$$P=(t_1+t_2)^2=t_1^2+2t_1t_2+t_2^2$$

thus $S(2,2)=3$ and $(t_1,t_2)=$ (0,2) or (1,1) or (2,0). Therefore $\sum_{i=1}^{2} t_i=3$ for all sets of (t_1,t_2).

Using this approach we know that the number of terms in the polynomial $P(t_1+t_2+\ldots+t_V)^Y$ is

$$S(V,Y)=\binom{V+Y-1}{V-1}$$

If we substitute V=2 and Y=2, $S(V,Y)=3$ and from the above example we know this to be true.

For our problem, V is the number of possible values the attribute has and Y varies from I-V+1 up to I. Since I=x-1, Y varies from x-v up to x-1. Therefore, the number of possible sets of ranges in the right hand side of equation (4.2) is

$$S(V,x) = \sum_{Y=x-V}^{x-1} \binom{V+Y-1}{V-1} \qquad (4.5)$$

Using the identity

$$\sum_{k=0}^{N} \binom{r+k}{k} = \binom{r+N+1}{N}$$

we reduce equation (4.5) to

$$S(V,x) = \sum_{Y=0}^{x-1} \binom{V+Y-1}{V-1} - \sum_{Y=0}^{x-V-1} \binom{V+Y-1}{V-1}$$

Therefore

$$S(V,x) = \binom{V+x-1}{x-1} - \binom{x-1}{x-V-1}$$

is the total number of possible sets of ranges for the right hand side of equation (4.2) when the range of COUNT($a_1a_2a_3$*)=[(x-1)S,xS-1] and V is the number of values in attribute 4. Our aim is to determine how many of these cases can be reduced.

The range of the response to the query of the left hand side of equation (4.2) can be reduced only when $\sum_{i=1}^{V} I_i = x-V$. Let R(V,x) represent the number of reducible cases. Therefore

$$R(V,x) = \binom{V+x-V-1}{V-1}$$

$$= \binom{x-1}{V-1}$$

So $R(V,x)=\binom{x-1}{V-1}$ sets of possible ranges can be reduced. We have seen that when V=2,(i.e. the attribute in question is a binary attribute) the number of reducible cases is x-1, this is confirmed here.

As in the binary case, we will assume that each set of counts has the same chance of occuring. With this assumption, we want to know the probability of a reduction occuring for any one query that has count lying in the range [(x-1)S,xS-1] and its attribute having V values. Call this probability P(V,x).

To determine P(V,x), we must know how many times a reducible case will occur and how many times a non reducible case will occur. In other words, how many possible combinations of true counts correspond to one reducible interval. Call this number CR(V,S).

$$CR(V,S) = \sum_{t=0}^{b-a-V+1} \binom{t+V-1}{t}$$

$$= \sum_{t=0}^{s-V} \binom{t+V-1}{t} \qquad (4.6)$$

where a=(x-1)S and b=xS-1.

Using the identity

$$\sum_{k=0}^{N} \binom{r+k}{k} = \binom{r+N+1}{N}$$

We reduce equation (4.6) to

$$CR(V,S) = \binom{S}{S-V} = \binom{S}{V}$$

Therefore there are $\binom{S}{V}$ possible combinations of true counts corresponding to each reducible set of responses. There are $R(V,x) = \binom{x-1}{V-1}$ sets of responses that cause a reduction, so the total number of possible combinations of true counts that will allow a user to reduce the range of a response is

$$TR = \binom{x-1}{V-1} \cdot \binom{S}{V}$$

Let $CN(V,S)$ be the number of possible combinations of true counts that correspond to one non reducible interval.

$$\begin{aligned} CN(V,S) &= \sum_{t=0}^{b-a} \binom{t+V-1}{t} \\ &= \sum_{t=0}^{S-1} \binom{t+V-1}{t} \\ &= \binom{V+S-1}{S-1} \\ &= \binom{V+S-1}{V} \end{aligned}$$

where $[a,b] = [(x-1)S, XS-1]$.

So the total number of possible combinations of true counts that will not allow a user to reduce the range of query response is

$$TN = CN(V,S)[S(V,x)-R(V,x)]$$

$$= \binom{V+S-1}{V}\left[\binom{V+x-1}{x-1}-\binom{x-1}{x-V-1}-\binom{x-1}{V-1}\right]$$

$$= \binom{V+S-1}{S-1}\left[\binom{V+x-1}{x-1}-\left[\binom{x-1}{V}+\binom{x-1}{V-1}\right]\right]$$

$$= \binom{V+S-1}{V}\left[\binom{V+x-1}{V}-\binom{x}{V}\right]$$

The total number of possible combinations of true counts to a query with response [(x-1)S,xS-1] is

$$T = TR+TN$$

$$= \binom{x-1}{V-1}\binom{S}{V}+\binom{V+S-1}{V}\left[\binom{V+x-1}{V}-\binom{x}{V}\right]$$

Therefore, the probability that the response range of a query can be reduced is

$$P(V,x) = \frac{TR}{T} = \frac{TR}{TR+TN}$$

$$= \frac{\binom{x-1}{V-1}\binom{S}{V}}{\binom{x-1}{V-1}\binom{S}{V}+\left[\binom{V+x-1}{V}-\binom{x}{V}\right]\binom{V+S-1}{V}}$$

where V>1 and S>1.

When x<V or S<V no reduction is possible. When reduction is possible, the range is reduced in size by V-1. Reduction to a single point (the size of the range is reduced by S-1) is only possible when V=S.

When V=S we have $\binom{x-1}{V-1}$ reducible cases and $\binom{V+x-1}{V}-\binom{x}{V}$ non reducible cases. There is only one possible combination of true counts that can cause any one of the reducible cases to occur, while $\binom{2V-1}{V}$ combinations of true counts can cause

a non reducible case to occur. For example, let x=5 and V=3 and the initial size of the range is S=3. The probability that this range will be reduced to a single point for any query is P=0.0234.

When a query COUNT(A), where A is a characteristic formula, has a response that lies in the first interval, then the range can not be reduced. In other words when COUNT(A)= [0,S-1], then this interval can not be reduced. The range [0,S-1] is [(x-1)S,xS-1] where x=1. So using the probability fuction P(V,x) we have

$$P(1,V)= \frac{\binom{0}{V-1}\binom{S}{V}}{\binom{0}{V-1}\binom{S}{V}+\left[\binom{V}{V}-\binom{1}{V}\right]\binom{V+S-1}{V}}$$

$$= 0$$

The probability that the range [0,S-1] will be reduced is 0.

Consider the equation

$$COUNT(1^*)=COUNT(10)+COUNT(11)+COUNT(12) \quad (4.7)$$

Here V=3 and let S=5. We know that

$$COUNT(10) \leq COUNT(1^*)$$

$$COUNT(11) \leq COUNT(1^*)$$

$$COUNT(12) \leq COUNT(1^*)$$

This is true since the records that match COUNT(10), COUNT(11) and COUNT(12) are contained within the set of records that match COUNT(1*). Therefore if COUNT(1*)= [0,4],

then the responses to the other queries must also be the range [0,4]. So we have

$$[0,4]=[0,4]+[0,4]+[0,4]$$

which is the only possible set of responses for equation (4.7). By inspection we see that every value in each range is a feasible solution, therefore we can not reduce the range of queries that have small counts.

P(V,x) is the probability that a range can be reduced by asking the queries in equation (4.2). It does not consider the fact that a reduction in the range of one query may affect the probability of reduction of another.

Consider the following example. The data contains records that have 3 attributes. Each attribute can have one of three values. The queries along with the responses listed in Example 4.6 is a subset of the possible queries on such a data base.

Suppose a user asks the query COUNT(***) in order to determine how many records exist in the data base. He wishes to reduce the range as much as possible. To do this he sets up the equation

$$\text{COUNT}(***)=\text{COUNT}(1**)+\text{COUNT}(2**)+\text{COUNT}(3**)$$

The response to these queries are

$$[95,99]=[20,24]+[30,34]+[40,44] \qquad (4.8)$$

EXAMPLE 4.6

QUERY	RESPONSE	QUERY	RESPONSE
COUNT(***)	[95,99]	COUNT(221)	[0,4]
COUNT(1**)	[20,24]	COUNT(222)	[0,4]
COUNT(2**)	[30,34]	COUNT(223)	[0,4]
COUNT(3**)	[40,44]	COUNT(231)	[0,4]
COUNT(11*)	[5,9]	COUNT(232)	[0,4]
COUNT(12*)	[0,4]	COUNT(233)	[0,4]
COUNT(13*)	[5,9]	COUNT(311)	[5,9]
COUNT(21*)	[10,14]	COUNT(312)	[0,4]
COUNT(22*)	[10,14]	COUNT(313)	[0,4]
COUNT(23*)	[5,9]	COUNT(321)	[0,4]
COUNT(31*)	[10,14]	COUNT(322)	[0,4]
COUNT(32*)	[10,14]	COUNT(323)	[0,4]
COUNT(33*)	[15,19]	COUNT(331)	[5,9]
COUNT(211)	[0,4]	COUNT(332)	[0,4]
COUNT(212)	[0,4]	COUNT(333)	[0,4]
COUNT(213)	[0,4]		

Looking at equation (4.8), we see that every value in the ranges is a feasible solution; therefore no reduction is possible. However, if a user is able to reduce COUNT(1**), COUNT(2**) or COUNT(3**), then he may be able to reduce COUNT(***). First consider COUNT(1**)

$$\text{COUNT}(1^{**}) = \text{COUNT}(11^{*}) + \text{COUNT}(12^{*}) + \text{COUNT}(13^{*})$$

$$[20,24] = [5,9] + [0,4] + [5,9]$$

By summing up the maximum possible counts of the right hand side we see that COUNT(1**) is no larger than 22, therefore we have

$$\text{COUNT}(1^{**}) = [20,22] \qquad (4.9)$$

Now consider COUNT(2**), and using the same procedure a user will attempt to reduce its range.

COUNT(2**)=COUNT(21*)+COUNT(22*)+COUNT(223)

[30,34] = [10,14] + [10,14] + [5,9] (4.10)

By inspection, we see that no reduction is possible; therefore we attempt to reduce COUNT(21*), COUNT(22*) and COUNT(23*). This will (hopefully) enable us to reduce COUNT(2**). So we have

COUNT(21*)=COUNT(211)+COUNT(212)+COUNT(213)

[10,14] = [0,4] + [0,4] + [0,4]

which reduces COUNT(21*) to [10,12];

and

COUNT(22*)=COUNT(221)+COUNT(222)+COUNT(223)

[10,14] = [0,4] + [0,4] + [0,4]

which reduces COUNT(22*) to [10,12];

and

COUNT(23*)=COUNT(231)+COUNT(232)+COUNT(233)

[5,9] = [0,4] + [0,4] + [0,4]

here no reduction is possible.

Substituting the above two reductions in equation (4.10) we have

COUNT(2**)=COUNT(21*)+COUNT(22*)+COUNT(23*)

$$[30,34] = [10,12] + [10,12] + [5,9]$$

We see that a reduction is now possible for COUNT(2**) and we get

$$\text{COUNT}(2^{**}) = [30,33] \qquad (4.11)$$

Finally consider COUNT(3**). Using the same procedure that we used for COUNT(2**) we get the following result:

$$\text{COUNT}(3^{**}) = [40,43] \qquad (4.12)$$

Using the reductions of equations (4.9), (4.11) and (4.12) in equation (4.8) we get

$$\text{COUNT}(***) = \text{COUNT}(1^{**}) + \text{COUNT}(2^{**}) + \text{COUNT}(3^{**})$$

$$[95,99] = [20,22] + [30,33] + [40,43]$$

The largest possible sum of the right hand side is 98, therefore the range for the response to the query COUNT(***) is reduced to [95,98]. To get this reduction a user will need all the queries in Example 4.6: 31 different queries.

$P(V_i x)$ is the probability of direct reduction from equation (4.8). However, with a lot of work, a user can reduce the range size by first reducing other ranges.

Similarly, the range [0,4] can be reduced by setting up an equation of type equation (4.8) in which the query with the range [0,4] is in the right hand side. Consider the following example of a data base that has 3 binary

attributes. The responses to a subset of the queries are defined in Example 4.7.

EXAMPLE 4.7

QUERY	RESPONSE	QUERY	RESPONSE
COUNT(*1*)	[5,9]	COUNT(21*)	[0,4]
COUNT(1**)	[5,9]	COUNT(111)	[0,4]
COUNT(11*)	[0,4]	COUNT(112)	[0,4]
COUNT(12*)	[5,9]		

Suppose we ask the query COUNT(11*), from Example 4.7 we know that the true count lies in the range [0,4].

$$\text{COUNT}(11*)=\text{COUNT}(111)+\text{COUNT}(112)$$

$$[0,4] = [0,4] + [0,4]$$

Trying to reduce this range in the usual way we see that no reduction is possible since every value in the ranges are possible true counts. However, the query COUNT(11*) appears in two equations of the type equation (4.8), namely

$$\text{COUNT}(1**)=\text{COUNT}(11*)+\text{COUNT}(12*)$$

$$[5,9] = [0,4] + [5,9]$$

and

$$\text{COUNT}(*1*)=\text{COUNT}(11*)+\text{COUNT}(21*)$$

$$[5,9] = [0,4] + [0,4]$$

In the first equation reduction is not possible, but in the second one we see that the range of COUNT(11*) can be reduced. The sum of COUNT(11*)+COUNT(21*) must be at least 5

and COUNT(21*) has a maximun value of 4, therefore COUNT(11*) must be at least 1. So the range of count(11*) is reduced from [0,4] to [1,4]. In Section 4.2 we will see how many ranges can be reduced directly and indirectly.

For queries requesting the average of the data fields, we will return the exact average. If the query involves fewer than S records than the response is undefined. This restriction is similar to the one imposed on the data base models in [6,14,17,18,23,24]. In order to compromise a data base with such a restriction, the authors of [6,18,23,24] use the concept of trackers, which are discussed in Chapter 2.

Trackers use the fact that the exact counts are known in order to deduce the counts of queries involving fewer than S records. Using ranges, instead of exact values, for count queries, the threat of compromise posed by trackers is eliminated.

Consider the following example on the data base in Example 4.8.

EXAMPLE 4.8

NAME	SEX	STATUS	DONATION
N1	M	STUDENT	$100.00
N2	M	STUDENT	200.00
N3	F	TEACHER	150.00
N4	M	STUDENT	50.00
N5	M	TEACHER	250.00
N6	F	STUDENT	300.00
N7	M	STUDENT	150.00
N8	M	STUDENT	30.00

A user knows individual I is a male teacher. He wants to know I's donation. He must first determine if (M•TEACHER) uniquely identifies I or not. If exact counts are given, and the response level is S=5, then using trackers he gets

$$\text{COUNT}(M \bullet \text{TEACHER}) = \text{COUNT}(M) - \text{COUNT}(M \bullet \overline{\text{TEACHER}})$$
$$= 6 - 5$$
$$= 1$$

Therefore, with exact counts the user is able to isolate individual I. To determine I's donation he asks the queries

$$\text{AVE}(M) = \$130.00$$

and

$$\text{AVE}(M \bullet \overline{\text{TEACHER}}) = \$106.00$$

amd computes I's donation by

$$\text{AVE}(M \bullet \text{TEACHER}) = 6(130.00) - 5(106.00)$$
$$= \$780.00 - 530.00$$
$$= \$250.00$$

Therefore the data base has been compromised.

If ranges are used for the counts we would have the following situation.

$$\text{COUNT}(M \bullet \text{TEACHER}) = \text{COUNT}(M) - \text{COUNT}(M \bullet \overline{\text{TEACHER}})$$
$$[0,4] = [5,9] - [5,9] \qquad (4.13)$$

Equation (4.13) is similar to equation (4.2) and we see that these ranges can not be reduced, therefore the user does not know if the formula (M•TEACHER) uniquely identifies individual I or not. Hence by giving ranges as the responses to count queries, we stop the user from isolating any one record.

Now suppose that the user has additional information (collected from external sources) that enables him to know that the formula (M•TEACHER) uniquely identifies individual I. With this information, he attempts to determine I's donation by using the two queries

$$AVE(M)=\$130.00$$

and

$$AVE(M\bullet\overline{TEACHER})=\$106.00$$

since the query AVE(M•TEACHER) is undefined. So we get

$$AVE(M\bullet TEACHER)= \$130.00\bullet(5,9)-\$106.00\bullet(5,9)$$

Taking every possible combination for the counts of (M) and (M•$\overline{\text{TEACHER}}$) he determines that I's contribution is either \$322.00, \$298.00, \$274.00 or \$250.00. He has no way of knowing which of these values is the exact contribution of individual I.

Therefore, even if a user knows information about individual I, and even if he has extra information about the whole data base (in this case the user knows that no other

individual in the data base has characteristics (M•TEACHER)), he can not determine with certainty the value of the data fields, even when the queries requesting statistics about the data fields return exact answers.

Thus the tracker is useless when the exact counts are not known.

In Section 4.2 we will implement sample data bases that use these range queries. We will analyze how many of the ranges can be reduced and by how much.

The results show that when the number of values per attribute increases the number of reductions decreases. The analysis of P(V,x) shows that this is true. Also the amount of work needed to reduce a query increases with the number of values per attribute. In equation (4.2) we see that there are V (the number of values per attribute) queries on the right hand side. Therefore as V increases so does the number of queries needed to reduce the range.

4.2 IMPLEMENTATION OF RANGE COUNTS AND RESULTS

In chapter 2 we mentioned that most of the proposed inference control mechanisms studied in previous research yielded negative results. Mechanisms such as setting controls on query set size and the amounts of overlap of query sets, distorting the data or the query responses and

giving responses by sampling from the data base are not effective or difficult (if not impossible) to implement. Thus most of the past research in this area has been to study efficient attacks rather than effective safeguards.

In our approach in Section 4.1 we showed that if a user can not determine the exact count of a query than no available method of attack, including trackers, can compromise the data base when average values are given as responses to queries which request information about data fields.

In this section we give results obtained from simulation of some sample data bases. Since no information can be inferred when the range is not reduced to a single point, then by knowing how often such reductions occur we can evaluate the degree of security of the data base.

The sample data bases were generated using a random number generator. The algorithm used has 3 parameters: N, the number of records in the data base; t, the number of attributes (characteristics) in each record; and V_i, i=1,t, the number of values attribute i can have. For example, using N=200, t=5, V=(2,2,2,2,2) the algorithm generates a data base that has 200 records, each records having 5 binary attributes. The simulation will generate data bases with these parameters such that two data bases with the same set of parameters will have different distributions of records

for a given set of values of attributes.

The querying system will accept any query of the form $q(a_1,a_2,a_3,\ldots,a_t)$ where a_i is one of the possible values for $atribute_i$. If we do not care about $atribute_i$ we set a_i=*. This will cause the querying system to ignore $attribute_i$. Then the querying system will return the average value of the specified data field or a range in which lies the true count of the records that match the query. A record matches a query, if for every a_i in the query, except a_i=*, a_i has the same value of $attribute_i$ in the record. The size of the range [a,b] was fixed at S=5, i,e. b-a+1=5.

In order to get a comprehensive set of results, we designed a program that generates all the possible queries, obtains a response for each query, and then determines if each query can be reduced or not, and by how much.

An s-query is a query where s out of the t attributes are specified, hence an s-query has (t-s) *'s.

We distinguish between direct and indirect reducability. We say that an s-query is directly reducible if by asking V_i (s+1)-queries we are able to reduce the range of the s-query. Consider a data base with records that have 3 binary attributes, i.e. V=(2,2,2). A 2-query is directly reducible if we can reduce its range by asking 2 3-queries. For example

COUNT(11*) = COUNT(111)+COUNT(112)

If we can reduce the range of COUNT(11*) by this equation, using the method described in the previous section, then COUNT(11*) is directly reducible.

We say that a query is indirectly reducible if we can not reduce its range directly but the range can be reduced by using previously reduced ranges. Consider Example 4.6 of the previous section.

Clearly indirect reduction involves much more work than direct reduction.

The following algorithm outlines the procedure used to determine if an S-query with range [a,b] is directly reducible or not.

Let t be the number of attributes per record and V_i, i=1,t be the number of possible values $attribute_i$ can have. Let p_i be the value of $attribute_i$ specified in the query. Therefore p_i is one of the V_i possible values of $attribute_i$ or p_i=* when the query does not specify a value for $attribute_i$.

ALGORITHM 4.1

1) $Q=COUNT(p_1,p_2,\ldots,p_t)=[a,b]$.

2) For each p_i=*,i=1,t do

   a) For j=1 to V_i do

      Form the query Q_j=Q except in position p_i=j.

b) Determine responses $[a_j, b_j]$ for each Q_j
V_i V_i

c) Let $L = \sum_{j=1} a_j$ and $U = \sum_{j=1} b_j$

d) If $a<L$ set $a=L$

If $b>U$ set $b=U$

EXAMPLE 4.9

To illustrate how this algorithm works, consider a data base that has 4 attributes with $V_1=2$, $V_2=2$, $V_3=3$ and $V_4=4$ (i.e. $attribute_1$ can have one of two values, $attribute_2$ can have one of two values, and so on).

The following are some of the possible queries along with their responses.

QUERY	RESPONSE
COUNT(1*3*)	[25,29]
COUNT(1*31)	[10,14]
COUNT(1*32)	[5,9]
COUNT(1*33)	[5,9]
COUNT(1*34)	[0,4]
COUNT(113*)	[5,9]
COUNT(123*)	[15,19]

Consider the query Q=COUNT(1*3*) which has as response the range [25,29]. Using Algorithm 4.1 we see that this range can be reduced. In step (1) [a,b]=[25,29]. Step (2) determines which attributes were not specified, i.e. which p_i=*. The query Q has a * in position 2 and position 4. For p_2=*, step (2) in the algorithm does the following:

a) For J=1 to V_2=2 do

form the queries Q_1=COUNT(113*)

Q_2=COUNT(123*)

Note that Q_1 and Q_2 are identical to Q except in the second position.

b) Determine the response for Q_1 and Q_2.

$[a_1,b_1]$=[5,9]

$[a_2,b_2]$=[15,19]

c) $L=a_1+a_2=20$

$U=b_1+b_2=28$

d) 29>28 so set b=28.

The range of query Q is thus reduced to [a,b]=[25,28].

For p_4=*, the algorithm forms the queries

Q_1=COUNT(1*31)

Q_2=COUNT(1*32)

Q_3=COUNT(1*33)

Q_4=COUNT(1*34)

and determines that L=20 and U=36, therefore no reduction is possible with attribute 4.

We say that the query COUNT(1*3*) is directly reducible by 1 on attribute 2.

For indirect reductions, we consider two cases. The first case results directly from diect reduction. If an s-query, Q, is directly reduced on attribute$_i$, then the (s-1)-queries (same as query Q, but with a value specified for attribute$_i$) are also reduced.

In Example 4.9, we saw that the query COUNT(1*3*) was reduced using the second attribute, therefore the queries COUNT(113*) and COUNT(123*) are also reduced.

The second case of indirect reduction is when we use ranges $[a_j,b_j]$ in step 2(b) of Algorithm 4.1 that have already been reduced, directly or indirectly. The results that we will describe below consider both direct and indirect reductions.

To get the total number of reductions, we applied Algorithm 4.1 to every possible query. This gave us the number of ranges that were directly reduced. We then update the ranges that were reduced directly and indirectly (case 1) and applied Algorithm 4.1 again to every query. We repeated this until no more reduction was possible. We will now describe the results obtained on randomly generated data bases.

The first data base generated contains 200 records, each record having 4 attributes

our notation, N=200, t=4 and V=(2,2,3,4). With such a data base, a user can generate 180 distinct queries. Using the method described above to determine if a range can be reduced or not we found that 34 ranges can be directly reduced by 1 and one range was directly reduced by 2. Using these direct reductions we were able to indirectly reduce more ranges. Both direct and indirect reduction enable 80 ranges to be reduced by 1 and 9 ranges by 2. Therefore 91 responses are not reducible. As shown in section 4.1, since no COUNT had its range reduced to a single point, V-compromisability does not occur.

The second data base generated has the same input parameters as the first; i.e. N=200, t=4 and V=(2,2,3,4). Again 180 queries were generated. Of the 180 ranges, 35 were directly reduced by 1, 3 directly by 2 and 1 was directly reduced by 3. Using these reduction we were able to indirectly reduce more ranges to get a total (direct and indirect) number of reductions of 84 ranges reduced by 1, 15 by 2, and 7 ranges were reduced by 4. Therefore in this data base we were able to determine the exact count of 7 queries. However no compromise was possible. We will now show how these 7 ranges were reduced to a single point and why this did not compromise the data base. Consider the set of queries along with their responses contained in Table 4.2.

These were the responses given by the querying system system responding to queries on the second data base.

TABLE 4.2

QUERY	RESPONSE	QUERY	RESPONSE
COUNT(****)	[200,204]	COUNT(*2*3)	[25,29]
COUNT(***1)	[30,34]	COUNT(11*3)	[0,4]
COUNT(***2)	[60,64]	COUNT(12*3)	[5,9]
COUNT(***3)	[30,34]	COUNT(1*13)	[0,4]
COUNT(***4)	[65,69]	COUNT(1*23)	[0,4]
COUNT(1**3)	[5,9]	COUNT(1*33)	[5,9]
COUNT(2**3)	[20,24]	COUNT(21*3)	[5,9]
COUNT(**13)	[0,4]	COUNT(22*3)	[15,19]
COUNT(**23)	[10,14]	COUNT(2*13)	[0,4]
COUNT(**33)	[15,19]	COUNT(2*23)	[5,9]
COUNT(*1*3)	[5,9]	COUNT(2*33)	[10,14]

The queries COUNT(****), COUNT(***1), COUNT(***2), COUNT(***3), COUNT(***4), COUNT(1**3) and COUNT(2**3) were reduced to the single points 200, 34, 64, 33, 69, 9 and 24 respectively.

The equation

COUNT(****)=COUNT(***1)+COUNT(***2)+COUNT(***3)+COUNT(***4)

causes the following reductions

COUNT(****)=[200,201]

COUNT(***1)= [33,34]

COUNT(***2)= [63,64]

COUNT(***3)= [33,34]

COUNT(***4)= [68,69]

Using these reductions we form the equation

COUNT(***3)=COUNT(1**3)+COUNT(2**3)—

[33,34] = [5,9] + [20,24]

and from this we see that the only possible values in these ranges are

COUNT(***3)= 33

COUNT(1**3)= 9

COUNT(2**3)= 24

With COUNT(***3)=33 we can determine the exact counts for COUNT(****), COUNT(***1), COUNT(***2) and COUNT(***4) which are 200, 34, 64 and 69 respectively.

Using these exact counts we try to reduce other ranges such as

COUNT(1**3)=COUNT(11*3)+COUNT(12*3)

9 = [0,4] + [5,9]

COUNT(1**3)=COUNT(1*13)+COUNT(1*23)+COUNT(1*33)

9 = [0,4] + [0,4] + [5,9]

We see that even though we know the exact count of COUNT(1**3) we can not reduce the other counts.

In this case we were able to determine the exact counts of 7 queries but we were not able to isolate any one individual; i.e. determine that COUNT(Q)=1 for some query Q.

Table 4.3 contains the number of direct reductions that occurred in each range of the first 2 data bases.

The next 2 data bases contain each N=1000 records, with each record having t=4 attributes and the number of values each attribute has is V=(2,2,3,4). The first data in this case has 37 ranges that are directly reduced by 1, 5 that are directly reduced by 2 and 1 range that is reduced by 3. The total number (direct and indirect) of ranges that are reduced is 77 reduced by 1, 20 reduced by 2 and 7 reduced by 3. Therefore 76 ranges are not reducible either directly or indirectly.

The second data base in this case has 38 and 5 ranges that are directly reducible by 1 and 2 respectively. The total number of ranges that were reduced is 91 reduced by 1 and 26 ranges reduced by 2. The number of responses that are not reducible is 63. Table 4.4 contains the number of direct reductions per range for these 2 data bases.

The next 3 data bases generated contain each N=200 records, each record having t=3 attributes and V=(4,3,6). The first data base had 1 range directly reduced by 2 and 4 ranges in total (directly and indirectly) were reduced by 2. In the second set of data,in this case, no reduction was possible. The third data base with N=200, t=3 and V=(4,3,6) had 2 ranges that were directly reduced by2 and 1 that was directly reduced by 3. In total there were 7 range reduced (directly and indirectly) by 2 and 5 ranges by 3. Table 4.5 shows the number of direct reductions that occurred in each range for these 3 sets of data.

The next 2 sample data bases generated had N=1000 records, t=3 and V=(4,3,6). The first data base had 4 ranges directly reduced by 2 and 12 were indirectly reduced by 2. Therefore 124 responses were not reducible. The second data base in this case had 1 range directly reduced by 2 and 3 more indirectly reduced by 2, leaving 136 ranges not reducible. Table 4.6 contains the number of direct reductions occurring in each range. Note that the data bases whose results are in Tables 4.5 and 4.6 have 140 possible distinct queries.

The data base, whose results are in Table 4.7, contains N=1000 records, each record having 6 attributes. The number of values each attribute can have is V=(2,2,3,4,3,6). In such a data base 5040 distinct queries exist. The number of queries that were reducible is 740, 98 and 2 reduced by 1, 2 and 3 respectively. Therefore 4200 queries were not directly reducible. The table contains the number of reductions occurring in each range.

The next data base generated has N=1000 records, t=6 and V=(3,3,3,4,3,6). Here we have a possibility of 8960 distinct queries of which 220 were directly reduced by 2 and 1 range was directly reduced by 1. Table 4.8 contains the number of ranges directly reduced in each interval.

Table 4.9 contains the number of direct reductions that occurred in 4 different data bases. Each data base has N=200

records that contain each t=5 attributes. The first data base has V=(2,2,2,2,2), i.e. 5 binary attributes. The second data base has V=(3,3,3,3,3), the third one has V=(4,4,4,4,4) and the fourth one has V=(5,5,5,5,5).

Table 4.10 contains the results of 4 data bases containing each N=1000 records. The other characteristics of these data bases are the same as the characteristics of the data bases whose results are in Table 4.9.

The results in Tables 4.9 and 4.10 show that as the number of values per attribute increases the amount of reduction that occurs decreases.

From the results obtained from the randomly generated data bases we can conclude the following:

1) The number of records in the data base has little influence (in most cases) on the number of reducible query ranges.

2) As the number of values (V) an attribute can have increases, the number of reducible cases decreases rapidly.

3) As the number of attributes increase, the number of possible reductions increases, however the number of existing queries increases even faster. Therefore, the ratio of the number of reducible query ranges over the total number of existing queries decreases.

4) The majority of the reductions are done with the attribute that has fewest values; therefore the size of the reduction, in most cases, is small.

5) Approximatly 90 percent of reductions occured on the attributes with the fewest possible values.

6) Even when exact counts can be determined, for some queries, this does not compromise the data base.

In Tables 4.3 through 4.10 the columns indicated by NR contain the number of responses that occurred in each range. The columns indicated by NRDR contain the number of queries that were directly reduced in that range.

TABLE 4.3

N=200 t=4 V=(2,2,3,4) 180 distinct queries

RANGE	SET 1 NR	SET 1 NRDR	SET 2 NR	SET 2 NRDR
[0,4]	90	0	57	0
[5,9]	21	9	37	6
[10,14]	9	2	26	11
[15,19]	10	3	16	7
[20,24]	10	4	9	0
[25,29]	10	4	5	3
[30,34]	1	0	6	3
[35,39]	3	2	4	1
[40,44]	4	2	0	0
[45,49]	2	1	6	3
[50,54]	4	1	1	1
[55,59]	1	1	2	0
[60,64]	1	0	3	0
[65,69]	1	0	1	0
[70,74]	3	0	0	0
[75,79]	2	1	1	0
[80,84]	1	0	1	1
[85,89]	0	0	0	0
[90,94]	0	0	0	0
[95,99]	1	0	0	0
[100,204]	6	5	5	3
TOTAL	180	35	180	39

TABLE 4.4

N=1000 t=4 V=(2,2,3,4) 180 distinct queries

RANGE	SET 1 NR	SET 1 NRDR	SET 2 NR	SET 2 NRDR
[0,4]	8	0	2	0
[5,9]	14	1	8	0
[10,14]	16	4	13	2
[15,19]	13	2	11	3
[20,24]	8	1	11	0
[25,29]	7	2	16	3
[30,34]	11	1	7	0
[35,39]	9	0	7	1
[40,44]	9	3	11	3
[45,49]	2	2	6	2
[50,54]	11	6	4	2
[55,74]	14	2	25	7
[75,99]	11	4	12	4
[100,124]	11	1	9	2
[125,149]	8	1	11	3
[150,199]	8	4	8	1
[200,299]	11	5	10	6
[300,399]	4	1	3	1
[400,499]	0	0	2	1
[500,1004]	5	3	4	2
TOTAL	180	43	180	43

determine the range within which a true value should lie. Therefore the uncertainty in the response cannot be removed and a data value cannot be compromised with certainty. However there is one disadvantage in this approach that is particularly applicable to small data bases; namely, this strategy lies at almost all queries.

In Chapter 4 we have proposed an inference control mechanism for data bases using attribute based queries. This strategy returns a range of lenght S for COUNT queries and the exact average for queries requesting information about data fields. We showed that by giving ranges for count queries it is impossible to isolate an individual record. By giving an average value instead of a SUM we are able to give exact answers and still assure that security will be upheld. The main advantage of this scheme is that the summaries obtained from the system are exact.

Finally we remark that our methods are easy to implement and the complexity of the strategy will not slow down the response time. By comparison to other methods such as controlling overlaps in query sets where implementation is almost impossible and random sampling which is not suitable to small or medium sized data bases, our methods provide effective security, good statistics and are easy to implement.

REFERENCES

1) Haq, M.I. On safeguarding statistical disclosure by Giving Approximate Answers to Queries. E. Morlet and D. Ribbens (Eds.), Int'l Computing Symp., North-Holland Pub., (1977).

2) DeMillo, R.A., Dobkin, D., and Lipton, R.J. Even databases that lie can be compromised. IEEE Trans. Soft. Eng. SE-4,1(Jan 1978), pp. 73-75.

3) DeMillo, R.A., Dobkin, D. Recent progress in secure computation. Tech. report, School of Info. and Computer Sc., Georgia Institute of Technology, Atlanta, Georgia.

4) Reiss, S.P. Security in databases: A combinatorial study. Journal of ACM, Vol. 26,1(Jan 1979), pp. 45-57.

5) DeMillo, R.A., Dobkin, D., and Lipton, R.J. Combinatorial inference. In Foundations of Secure Computation, R.A. DeMillo et al., Eds. Academic Press, New York, 1978, pp. 27-35.

6) Denning, D.E., Denning, P.J., and Schwartz, M.D. The tracker: A threat to statistical database security. ACM Trans. Database Syst. 4,1(March 1979), pp. 76-96.

7) Haq, M.I. Insuring individual's privacy from statistical data base users. Proc. AFIPS 1975 NCC, Vol. 44, AFIPS

Press, Montvale, N.J., pp.941-946.

8) Fellegi, I.P., and Phillips, J.L. Statistical confidentiality: Some theory and applications to data dessemination. Ann. Econ. Soc. Meas. Vol. 3,2(April 1974), pp. 399-409.

9) Conway, R., and Strip, D. Selective partial access to a database. Proc. 1976 ACM Ann. Conf., pp. 85-89.

10) Fellegi, I.P. On the question of statistical confidentiality. Journal of the Americain Statistical Association. Vol. 67,337(March 1972), pp. 7-18.

11) Turn, R., and Shapiro, N.Z. Privacy and security in databank systems: Measures of effectiveness, cost, and protector-intruder interactions. Fall Joint Computer Conference, 1972.

12) Hansen, H.H. Insuring confidentiality of individual records in data storage and retrieval for statistical purposes. Proc. AFIPS 1971 FJCC, Vol. 39, AFIPS Press, Arlington, Va., pp. 579-585.

13) Schwartz, M.D., Denning, D.E., and Denning, P.J. Securing data bases under linear queries. Proc. IFIP Congress, North-Holland Pub., (1977), pp. 395-398.

14) Chin, F.Y. Security in statistical databases for queries with small counts. ACM Trans. Database Syst.

Vol. 3,1(March 1978), pp. 92-104.

15) Dobkin, D., Jones, A.K., and Lipton, R.J. Secure Databases: Protection against user influence. ACM Trans. Database Syst., Vol. 4,1(March 1979), pp. 97-106.

16) Denning, D.E. Are statistical data bases secure? Proc. AFIPS 1978 NCC, Vol. 47, AFIPS Press, Arlington, Va., pp. 525-530.

17) Kam, J.B., and Ullman, J.D. A model of statistical databases and their security. ACM Trans. Database Syst. Vol. 2,1(March 1977), pp. 1-10.

18) Hoffman, L.J., and Miller, W.F. Getting a personal dossier form a statistical data bank. Datamation Vol. 16,5(May 1970), pp. 74-75.

19) Denning, D.E., and Denning, P.J. Data security. Computing Surveys, Vol. 11,3(Sept 1979), pp. 227-249.

20) Eswaran, K.P. Faithful representation of a family of sets by a set of intervals. SIAM J. Comput., Vol. 4,1(March 1975), pp. 56-68.

21) Lipton, R.J., and Snyder, L. A linear time algorithm for deciding subject security. Journal of ACM, Vol. 24,3(July 1977), pp. 455-464.

22) Denning, D.E. Secure statistical databases with random sample queries. ACM Trans. Database Syst., Vol. 5,3(Sept

1980), pp. 291-315.

23) Denning, D.E., and Schlorer, J. A fast procedure for finding a tracker in a statistical database. ACM Trans. Database Syst., Vol. 5,1(March 1980), pp. 88-102.

24) Schlorer, J. Identification and retrieval of personal records from a statistical data bank. Methods of Information in Medicine, Vol 14,1(Jan 1975), pp. 7-13.

25) Schlorer, J. Confidentiality of statistical records: A threat monitoring scheme for on line dialogue. Methods of Information in Medicine, Vol. 15,1(Jan 1976), pp. 36-42.

26) "Records, Computers, and the Rights of the Citizens." Report of the Secretary's Advisory Committee on automated data systems, U.S. Department of Health, Education and Welfare. Copyright 1973 by the Massachusetts Institute of Technology.

27) Tarjan, R.E., and Trojanowski, A.E. Finding a maximum independent set. SIAM J. Comput., Vol. 6,3(Sept 1977), pp. 537-546.

28) Tsukiyama, S., Ide, M., Ariyoshi, H., and Shirakawa, I. A new algorithm for generating all the maximal independent sets. SIAM J. Comput., Vol. 6,3(Sept 1977), pp. 505-517.