

## Essay 24

# Inference Problems in Multilevel Secure Database Management Systems

Sushil Jajodia and Catherine Meadows

---

An inference channel in a database is a means by which one can infer data classified at a high level from data classified at a low level. The inference problem is the problem of detecting and removing inference channels. It is clear that inference problems are of vital interest to the designers and users of secure databases. Database management systems are intended to provide the means for efficient storage and retrieval of information. Their very power means that if they are not properly designed to prevent illegal inferences, they not only will not prevent such inferences, but will greatly assist users in forming them. Yet so far inference problems in multilevel databases have not been studied very deeply. This is partly due to the difficulty of the problem, and probably also due to the fact that one cannot implement any means of controlling inferences until one has solved the more fundamental problem of determining how one stores and retrieves multilevel data.

This essay surveys the state of the art of the study of inference problems in multilevel databases. We describe particular strategies that have been developed for certain inference problems, as well as more general models of the inference problems and the tools that have been developed for handling them. We do not describe work on preventing inferences in statistical databases, which we consider a specialized problem not necessarily relevant to the inference problem in multilevel databases. However, we do note that the work on statistical databases shows that one can be successful in preventing inferences if one carefully limits the scope of the problem one is studying.

Before beginning our survey, we should point out that all the models and techniques discussed here, and indeed all attempts to deal with inference control in database systems, have one limitation in common. An inference of sensitive data from nonsensitive data can only be repre-

sented within a database if the nonsensitive data itself is stored in the database. We have no way of controlling what data is learned outside of the database, and our abilities to predict it will be limited. Thus even the best model can give us only an approximate idea of how safe a database is from illegal inferences. This fact should always be kept in mind when dealing with inference problems.

The remainder of this essay is structured as follows. We begin with a description of some specific inference channels that have been discovered in the design of multilevel databases, and some techniques that have been developed to close them. Then we move to a more general consideration of the inference problem, and we describe some of the formal models and characterizations of the general inference problem that have been developed. We also describe some of the tools that have been proposed and developed for dealing with inference problems in databases. We discuss aggregation problems, which constitute a special kind of inference problem that is usually mentioned in connection with inference problems in databases.

### **Specific inference problems**

In this section we describe a number of inference channels that have been discovered in the course of database security research, and some techniques that have been developed to deal with them. These channels by no means constitute an exhaustive list of all the possible channels that can arise in a multilevel database; however, they are often easy to recognize and avoid. In some cases, specialized techniques have been developed to handle them. Thus, they deserve special mention.

**Inference from queries based on sensitive data.** In this section, we assume that a user makes a sequence of queries  $q_1, q_2, \dots, q_n$  against the database. The user then utilizes the responses to these queries to derive an inference which has a higher classification than that of the retrieved data. We illustrate this situation by giving an example.

*Example 1* [MEAD88a]. Suppose that data is classified at the relation level. We have two relations, an Unclassified relation called EP, with attributes EMPLOYEE-NAME and PROJECT-NAME, and a Secret relation called PT, with attributes PROJECT-NAME and PROJECT-TYPE, where EMPLOYEE-NAME is the key of the first relation and PROJECT-NAME is the key of the second. (The existence of the relation scheme PT is Unclassified.) Suppose an unclassified user makes the following SQL query:

```
SELECT EP.EMPLOYEE-NAME
FROM   EP, PT
WHERE  EP.PROJECT-NAME = PT.PROJECT-NAME
```

If this query is evaluated by taking the natural join of the two relations EP and PT along PROJECT-NAME, and then projecting along EMPLOYEE-NAME, we have an inference channel, even though only the Unclassified data (employee names) is being returned to the user. Although the output of this query is Unclassified, it reveals Secret information.

The following SQL query posed by an uncleared user represents a similar problem:

```
SELECT EP.EMPLOYEE-NAME
FROM   EP, PT
WHERE  EP.PROJECT-NAME = PT.PROJECT-NAME
AND    PT.PROJECT-TYPE = 'SDI'
```

If we examine the above SQL queries carefully, we quickly observe that even though the data returned to the user has a low classification, the data that is required to evaluate the query has a higher classification. Thus, the inference channels arise from the fact that these queries are conditioned on data that are supposed to be invisible to the user [DENN86a].

Inferences of this type are easy to eliminate. The system can either modify the user query such that the query involves only the authorized data or simply abort the query. If the user is cleared to see all data involved in the query, then the result can be returned to him, but it must be labeled at the least upper bound of all labels involved in the query.

**Statistical databases.** The problem of statistical database security is the problem of answering queries about statistics on data, such as mean, median, standard deviation, and so on, without releasing the data itself. A typical application of statistical database security would be that used by the US Census Bureau, in which aggregate statistics on groups of individuals are made public (for example, the average income of people in a particular geographic area), but in which information about particular individuals is kept secret. The threat against statistical database security is that an attacker may be able to find out statistical information about individuals by posing queries on aggregate statistics over a period of time and performing arithmetic operations on the answers received, using his own information about the size and nature of the sets of individuals involved. For example, suppose that an attacker wishes to find out the salary of A1. He can do this by asking for the average salaries of A1, A2, and A3; of A2, A3, A4, and A5; and of A4 and A5. Or he can ask for the average salary of some set of individuals of which he knows that A1 is the only member.

The security of statistical databases is an area that has received much study and really deserves an essay of its own. Rather than attempt to

cover the subject here, we refer the reader to several of the survey papers that already exist [ADAM89, DENN83].

**Inference from data combined with metadata.** Many of the inference channels that arise in multilevel secure DBMSs are due to combining data retrieved from the database with the metadata used for its storage and management. This is particularly true for integrity constraints. A user at a low security class can use his or her knowledge of the low security class data and of the constraint (if it is made available to the user) to infer information about high security class data also affected by the constraint. In this section we will consider some kinds of metadata used in databases, and show how it can be used to assist in making inferences. We will concentrate on the relational model, since this is the one that has been most closely studied from the point of view of security. However, we note that many of these issues arise when other models are used, although in somewhat different form.

*Key integrity.* One of the basic integrity requirements of the relational model is key integrity, which requires that every tuple in a relation must have a unique key. This constraint does not cause a problem when data is classified at the relation or column level, since in that case all keys in a relation are at the same security class. But consider a low security class user who wants to enter a tuple in a relation in which data is classified at either the tuple or the element level. If a tuple with the same key at a higher security class already exists, then to maintain key integrity, the DBMS must either delete the existing tuple or inform the user that a tuple with that key already exists. In either case, we have a problem. In the first case, the actions of a low user can cause data inserted by a high user to be deleted, which is unacceptable.<sup>1</sup>In the second case, we have an inference channel: The existence of high data is allowed to affect the existence of low data.

To illustrate, consider the following instance (where “Name” is the key for the relation):

Label	Name	Destination	Engine
S	Wombat	Persian Gulf	Nuclear

---

<sup>1</sup>Although it is not a secrecy violation, it could lead to serious integrity problems and also cause denial of service.

Suppose an unclassified user wants to insert the tuple (Wombat, Norfolk, Nuclear). If we choose to preserve key integrity, we must either delete the secret tuple or reject this insertion. We have an integrity problem if we delete the secret tuple (since it is possible that the entry “Norfolk” in the unclassified tuple is merely a cover story for the real, classified entry “Persian Gulf”). If we reject the insertion, then the low user can derive an inference.

It turns out that this problem can be eliminated using polyinstantiation, in which case both tuples are allowed to exist. Polyinstantiation and its effects are discussed in more detail in Essay 21.

*Functional and multivalued dependencies.* A more general situation than the one discussed in the previous section has been considered by Su and Ozsoyoglu [SU86, SU87, SU90]. They study inference channels that arise because of the functional and multivalued dependencies that are constraints over the attributes of a relation. The following example illustrates how inference channels can arise if certain functional dependencies are known to low users.

*Example 2* [SU87]. Assume that a company database consists of the relation scheme EMP-SAL, which has three attributes: NAME, RANK, and SALARY. The attributes NAME and RANK are considered nonsensitive in the database, while the attribute SALARY is considered sensitive. (Thus, the labeling is at the column level.) Suppose every employee is aware of the constraint that all employees having identical ranks have the same salaries. Given this scenario, an employee who is not permitted to have access to sensitive data can easily determine employee salaries, which are sensitive.

If we examine the above example carefully, we see that it contains an inference channel because the functional dependency  $RANK \rightarrow SALARY$  is not properly reflected in the classification levels of attributes RANK and SALARY. If the rank of an employee is known to a user, then the employee’s salary is also known to that user. The way to avoid the problem in cases such as this is to raise the classification of the attribute RANK from nonsensitive to sensitive. If attributes are assigned security labels in a manner consistent with the functional dependencies, then these inference threats can be eliminated. This process is formalized by Su and Ozsoyoglu [SU86, SU87, SU90].

Su and Ozsoyoglu give several algorithms for raising the classification labels of attributes based on functional and multivalued dependencies among them. One of their algorithms takes as input a list of attributes, the proposed classification labels of the attributes, and a set of functional dependencies that cause inferences. The algorithm produces as output another list of attributes together with their classification labels

(which may be different from the assignment that was input), such that the list is free of inference channels arising from functional dependencies.

*Value constraints.* A value constraint is a constraint on data values that can involve one or more items of data. If a constraint is defined over data at different security levels, availability of the constraint may lead to inference channels.

*Example 3* [MEAD88a]. Suppose that an attribute A is Unclassified while attribute B is Secret. Suppose the database enforces the constraint  $A + B \leq 20$ , which is made available to Unclassified users. The value of B does not affect the value of A directly, but it does determine the set of possible values A can take. Thus we have an inference channel. The usual solution to this problem is to allow such constraints to be defined only over a single security level. If a constraint is defined over several levels, then it is necessary to partition it into several single-level constraints. Thus, for example, the constraint  $A + B \leq 20$  can be partitioned into  $A \leq 10$  and  $B \leq 10$ .

*Classification constraints.* A classification constraint is a rule describing the criteria according to which data is classified. It is also possible to infer sensitive data from the classification constraints themselves, if they are known. Consider the following example.

*Example 4* [SICH83]. Suppose the following integrity constraints apply to a database containing the fact that Mediocrates is an Athenian:

Every man is an Athenian, a Boeotian, a Corinthian, or a Dorian.  
All Athenians and Corinthians are peaceable.  
All Boeotians and Dorians are violent.

Mediocrates does not wish it to be known that he is peaceable. Rhinologus, a public nuisance, tries to find out about Mediocrates from a system that refuses to answer whenever the answer, together with the integrity constraints, would imply the secret:

Rhinologus: Is Mediocrates an Athenian?  
System: I will not tell you.  
Rhinologus: Is he a Boeotian?  
System: No.  
Rhinologus: Is he a Corinthian?  
System: No.  
Rhinologus: Is he a Dorian, then?  
System: I will not tell you.

Rhinologus, knowing that it is his disposition that Mediocrates is trying to conceal, notes that if he were not an Athenian, then the system could have answered “no” to the first question without revealing the secret. Thus Mediocrates must be a peaceable Athenian.

Sicherman, de Jonge, and van de Riet [SICH83] consider the problem of protecting secrets in cases when classification constraints are known and when they are not known. They define a number of conditions for refusing to answer a query, and describe a set of strategies based on these conditions. They then develop a formal model of a secure database with inference rules and prove theorems that show what are the safe responses, depending on whether or not the classification constraints are known.

The results Sicherman, de Jonge, and van de Riet present are rather pessimistic. In particular, they show that, if the classification constraints are known, then the only one of their strategies that is safe requires that no queries about a value in a tuple be answered if that value is secret. This is a result of the rigidity of their strategies: Each of their strategies applies a set of criteria consistently over time. Thus it is not possible to adopt a more liberal policy initially and switch to a more restrictive one as the number of options for the secret value decreases, as is done for some statistical databases. However, their result is important because of its extreme generality: It applies to all databases and all inference systems, and thus can be used as a basis for further study of this problem.

## **General characterizations of the inference problem**

In the previous section, we discussed some specific inference channels that could be easily characterized and had relatively straightforward solutions. But pinning down inference channels in general is not always so easy. As a case in point, consider the following two stories from Stanislaw Ulam’s *Adventures of a Mathematician* [ULAM76]. The stories concern the US government’s attempt to keep the existence and the location of the Manhattan Project secret during World War II:

Finally I learned that we were going to New Mexico, to a place not far from Santa Fe. Never having heard about New Mexico, I went to the library and borrowed the Federal Writers’ Project Guide to New Mexico. At the back of the book, on the slip of paper on which borrowers signed their names, I read the names of Joan Hinton, David Frisch, Joseph McKibben, and all the other people who had been mysteriously disappearing to hush-hush war jobs without saying where. I had uncovered their destination

in a simple and unexpected fashion. It is next to impossible to maintain absolute secrecy and security in war time.

This reminds me of another story. Since I knew Stebbins well, about a month after arriving at Los Alamos, I wrote to him. I did not say where I was but I mentioned that in January or February I had seen the star Canopus on the horizon. Later it occurred to me that as an astronomer he could easily have deduced my latitude since this star of the Southern skies is not visible above the 38th parallel.

There are several important things to note about these stories. First of all, in figuring out the destination of his mysteriously vanishing colleagues, Ulam did not use standard logical deduction. Instead, he looked for the best explanation for the fact that they had all checked out a guidebook for the same place. Second, although Ulam was able to deduce the destination of his friends from the guidebook, he did not know enough to look for the guidebook until he already knew some sensitive information: the location of the project he himself would be working on. If he hadn't known that information, he might have had to look at every guidebook in the library to find out the location of the Manhattan Project. Third, Ulam also needed some relatively nonsensitive information to make his deduction: namely, the fact that some of his colleagues had been leaving to work on what were apparently secret government projects. Finally, it is not always possible to predict what information an individual will know that can be used to deduce facts from other, apparently nonsensitive facts; this is the point of the story about Canopus and the 38th parallel.

Probably the earliest formal characterization of the inference problem in databases is that of Goguen and Meseguer [GOGU84]. Consider a database in which each data item is given an access class, and suppose that the set of access classes is partially ordered. Define the relation  $\rightarrow$  as follows: Given data items  $x$  and  $y$ , we say  $x \rightarrow y$  if it is possible to infer  $y$  from  $x$ . The relation  $\rightarrow$  is reflexive and transitive. A set  $S$  is said to be inferentially closed if whenever  $x$  is in  $S$  and  $x \rightarrow y$  holds, then  $y$  belongs to  $S$  as well. Now, for an access class  $L$ , let  $E(L)$  denote the set consisting of all possible responses that are classified at access class less than or equal to  $L$ . There is an inference channel if  $E(L)$  is not inferentially closed.

Goguen and Meseguer do not set forth any one candidate for the relation  $\rightarrow$ . They merely require that it be reflexive and transitive, and say that it will probably be generated according to some set of rules of inference: for example, first-order logic, statistical inference, nonmonotonic logic, knowledge-based inference, and so on. They do note, however, that for most inference systems of interest, determining that  $A \rightarrow b$

(where  $A$  is a set of facts and  $b$  is a fact) is at best semidecidable — that is, there is an algorithm that will give the answer in a finite amount of time if  $A \rightarrow b$ , but otherwise may never halt. Complexity-theoretic properties of inference relations in multilevel databases are considered further by Thuraisingham [THUR90].

Goguen and Meseguer, besides not fixing on any inference system or set of inference systems, leave several other questions unexplored. For example, they do not consider the effect on the inference problem of information that may exist outside the database (as in the Canopus example cited above). Nor do they attempt to include any measure of the difficulty of computing an inference. Some of these problems, however, have been discussed in the work that has followed.

In a refinement of Goguen and Meseguer's definition, Denning and Morgenstern [DENN86] derive the inference relation from classical information theory. Given two data items  $x$  and  $y$ , let  $H(y)$  denote the uncertainty of  $y$ , and let  $H_x(y)$  denote the uncertainty of  $y$  given  $x$  (where uncertainty is defined in the usual information-theoretic way). Then, the reduction in uncertainty of  $y$  given  $x$  is defined as follows:

$$INFER(x \rightarrow y) = \frac{H(y) - H_x(y)}{H(y)}$$

The value of  $INFER(x \rightarrow y)$  is between 0 and 1. If the value is 0, then it is impossible to infer any information about  $y$  from  $x$ . If the value is between 0 and 1, then  $y$  becomes somewhat more likely given  $x$ . If the value is 1, then  $y$  can be inferred given  $x$ .

The  $INFER$  relation can be used in the following way, as is done by Morgenstern [MORG88]. Choose an  $\epsilon > 0$  and define the *sphere of influence* of a set (or core) of data to be the set of all data items  $y$  such that there exists an  $x$  in the core such that  $INFER(x \rightarrow y) > \epsilon$ . A system is safe from inference if, for each security class  $C$ , the sphere of influence of all data items with security level  $\leq C$  does not contain any data elements of a higher or incomparable class.

Note that, unlike the inference relations discussed by Goguen and Meseguer, the sphere of influence relation is nontransitive. For example, knowing the street on which someone lives may reduce our uncertainty about that person, knowing that an individual works on a project may reduce our uncertainty about the nature of the project, but knowing the name of a street inhabited by someone working on a project reduces our uncertainty about the project somewhat less.

This formulation is especially nice since it shows that inference is not an absolute problem, but a relative one. It gives us a way to quantify the bandwidth of the illegal information flow. On the other hand, Denning and Morgenstern [DENN86, p. 5] point out its serious drawbacks:

1. In most cases it is difficult, if not impossible, to determine the value of  $H_x(y)$ .
2. It does not take into account the computational complexity that is required to draw the inference.

To illustrate the second point, they give the following example from cryptography: With few exceptions, the original text can be inferred from the encrypted text by trying all possible keys (so  $H_x(y) = 1$  in this case). However, it is hardly practical to do so.

A discussion of the kinds of inference functions that would be appropriate for determining the security of a multilevel database is given by Garvey, Lunt, and Stickel [GARV91]. The authors discuss three kinds of inference channels:

- deductive channels, in which high data may be formally derived from low data;
- abductive channels, in which a deductive proof could be completed if certain low-level axioms were assumed; and
- probabilistic channels, which occur when there is low data that can be used to decrease the uncertainty about the nature of high data, and it is likely that the low data will be known by a low user.

Ulam's second story, about the sighting of Canopus, is an example of abductive reasoning applied in the manner proposed by Garvey et al. (although a little too late). Ulam wrote his friend that he had seen Canopus on the horizon and then realized that, not only could the low-level fact, together with another low-level fact, be used to deduce the high-level fact, but that the second low-level fact was likely to be known to the low-level recipient of his letter.

Techniques and tools that make use of abductive reasoning and probabilistic reasoning provide a good way of characterizing cases in which high-level information can be derived using low-level information from both inside and outside a database. Garvey et al. [GARV91] discuss how these techniques might be used to make sure that a database is free from inference channels. For each high-level fact in the database, one attempts to discover what low-level facts could be used either to derive the high-level fact or to make it more likely to be derived. One also measures the likelihood that these low-level facts are known. Garvey et al. also point out that automated tools used for abductive proofs have features that measure the difficulty of a proof, and that such a feature can provide a guide to the difficulty a low-level user would have in performing the inference.

## Tools and techniques for dealing with inference channels

In this section, we discuss the various techniques and tools that have been proposed for locating inference channels and preventing their exploitation once they have been found. Basically two kinds of techniques have been proposed for locating and eliminating inference channels. One is to use semantic data modeling techniques to detect inference channels in the database design, and then to redesign the database so that these channels no longer exist. The other is to evaluate database transactions (involving either reads or updates, or both) to determine whether they lead to illegal inferences. If they do, the query is either disallowed or reclassified at the higher level.

One of the earliest examples of the semantic data modeling approach is found in Hinke's work on the ASD Views project [HINK88a]. This work describes the construction of a semantic relationship graph to express the possible inferences in a database. Data items are represented as nodes connected by edges that represent the relationships between them. A relationship may or may not be classified. A possible inference channel is said to exist if there are two paths from node A to node B such that it is possible to be cleared to see all edges in one path and not all edges in another. If such an inference channel is found, one analyzes to see whether it is a genuine channel. If it is, one raises the classification of one or more edges, if possible. This process continues until all inference channels are closed.

This technique is very simple. No assumptions are made about the nature of the relationships between data items except that they either exist or they do not. For example, if A implies B, then an inference path is assumed to exist, not only from A to B, but from B to A. Applying this technique is relatively straightforward. The main danger appears to be that, if data is very highly interrelated, then so many false channels will be discovered that the database designer wastes most of his or her time trying to eliminate them.

An implementation of a model similar to Morgenstern's is described by Buczkowski [BUCZ90]. In Buczkowski's system, a PINFER function is developed that is similar to the INFER function described by Denning and Morgenstern. For certain pairs of facts  $x$  and  $y$ , one asks an expert to determine  $PINFER(x, y)$ , the probability that one can infer  $y$  from  $x$ . One then uses fuzzy logic to estimate other probabilities — for example, the probability that one can infer  $z$  from  $x$  — where these probabilities have not been explicitly provided by the expert. These probabilities are computed using knowledge about the degree of dependency between various facts.

Smith [SMIT90, SMIT90a] proposed a scheme using a semantic data model that refines Hinke's by allowing the user to express different

kinds of relationships. In Smith's system a number of possible types of data items and relationships between them are identified. These include abstract class, attribute, identificate (an identificate is something that can serve to identify a data item, or nearly so), association, external identifier, generalization, and disjoint subclasses. Any of these can be classified. Thus, if one wishes to hide the relationship between employee and salary, one merely classifies that relationship, and leaves both employee and salary unclassified.

This technique also has its dangers, although it is less likely to give false inference channels than Hinke's tool. Since types of relationships are enumerated and explicitly identified in the database, there is the danger that not all possible types of relationships have been identified. Moreover, it is not necessarily the case that classifying a relationship will make it impossible to infer by some other means. This is especially true if the database is dynamic. For example, Lunt [LUNT89a] points out that, even if one classifies the association between employees and salaries, it is still possible that it can be deduced for some cases whenever employee-salary pairs are entered into or deleted from the system, or if both employees and salaries are presented in the same order. Thus, this approach should be thought of more as a tool for developing and describing a security policy than as a way of telling the database designer exactly what data should be classified.

All of the tools described above are used to assist in database design and classification. Other researchers have considered tools to evaluate queries. Mazumdar, Stemple, and Sheard [MAZU88] propose a system that evaluates the security of transactions by using a theorem prover to determine whether one of a set of predefined secrets can be derived from the integrity constraints of the database, the precondition of the transaction, and the type of data input into the transaction. Since their techniques do not depend on any specific data that is stored in the database, they can be applied to a transaction when it is compiled. Thus they can be used to determine, for example, what the level of a certain type of canned transaction should be. Mazumdar et al. also point out that their techniques can be used to test the security of the database and point out ways in which it can be redesigned to make it more secure.

A somewhat different approach is followed by the Lock Data Views (LDV) project [STAC90]. Classification constraints are defined on sets of data according to the level of the information that may be inferred from the data. When a query is presented to the system, the result is upgraded to the appropriate level according to the classification constraints and then released. It is possible to include an inference control mechanism in this system so that the classification constraints will be based on the results of its analysis of a query. It is also possible to use a history mechanism to guard against gathering together information that

can be used to infer sensitive information over time. Thus if A and B together imply C where C is sensitive and A and B by themselves are not, and a low-level user has already seen A, one can prevent that user from seeing B, or prevent any low-level user from seeing B, according to the security policy.

Another approach, described by Thuraisingham [THUR87], uses a version of query modification [STON74]. A query is first evaluated to determine whether it will lead to illegal inferences, and then, if it will, is transformed into a query that will not lead to such inferences. Such an approach can be helpful to a user since, not only does the system prevent the user from making illegal queries, it also assists him or her in making legal queries. However, such a system must be carefully implemented, since there is the danger that a user may be able to form inferences from the ways in which legal queries differ from illegal queries.

Searching for inferences at query processing time has some obvious potential drawbacks. As has been pointed out [HAIG90], it may be necessary to keep some kind of history of past accesses and to use that history when evaluating the security level of a query, so that low-level users cannot infer high-level information by building up a store of low-level information through repeated queries. In certain circumstances, this can make the system vulnerable to a denial-of-service attack: A low-level user could prevent other low-level users from accessing data they need by accessing other data, that, together with the needed data, would lead to an inference channel.

Nonetheless, such an approach may have its use in certain applications. For one thing, in many cases it may be prohibitively expensive to search an entire database for illegal information flows, but somewhat easier to evaluate a particular query. Moreover, the addition of new data in updates may cause new illegal inferences that are best checked for at the time of the update; a query-time inference checker might be useful in detecting such inferences. But it is clear that more work needs to be done to understand the relative advantages and disadvantages of the two approaches.

## **Aggregation problems**

Inference and aggregation are usually discussed together. Many of the tools discussed in the previous section are described by their builders not as inference detecting tools, but as inference and aggregation tools. But aggregation problems, although they are motivated by inference problems, are actually somewhat different in nature, and thus we devote a separate section to them.

We say that an aggregation problem or aggregation policy exists when the aggregate of two or more data items is classified at a level higher than the least upper bound of the classification of the individual items.

The most commonly cited example is the SGA (Secretive Government Agency) phone book [SCHA83]: The entire phone book is classified but individual numbers are not.

Aggregation policies are usually assumed to arise because sensitive information can be deduced from the aggregate that cannot be deduced from any individual member. Thus, they are generally discussed in connection with inference problems. However, they differ in that in the case of an aggregation problem, a particular labeling policy has been recommended. Even so, once such an aggregation problem has been identified, there is still a wide variation in how it can be handled, depending on the situation. Some examples include [MEAD90]:

1. Use the aggregation policy as a guide for downgrading. That is, begin by classifying all members of the aggregate at the level of the aggregate, and then downgrade as many as is consistent with the aggregation policy.
2. Use the aggregation policy as a guide for relaxing security requirements. In one example, the members of the aggregate were made available only to individuals who were cleared to the level of the aggregate, but they were allowed to follow less strict policies for handling individual aggregate members. Thus, an Unclassified member of a Confidential aggregate could be stored on an Unclassified PC.
3. Release individual members of an aggregate to individuals cleared at the lower level, but do not release more than a certain fixed number to any one individual. This was the policy followed in the SGA phone book example. Any individual could be given as many as  $N$  phone numbers, where  $N$  was some fixed number, but no more.

Another possible way of handling an aggregation problem can be used when inferences may be formed by watching the ways in which data changes over time. In this case, one could prevent inferences by limiting not the amount of data an individual sees, but the amount of time during which he has access to the data. This is essentially the “snapshot” approach discussed by Meadows and Jajodia [MEAD88a].

Aggregation policies are generally the result of an uneasy compromise between the need to protect sensitive data and the need to release useful data. This makes them difficult to account for in a formal security model and difficult to implement in a secure database without violating the general security policy. Moreover, the fact that different aggregation problems are dealt with using different security policies means that it is hard to come up with a mechanism that can be used to deal with all cases.

Nevertheless, a number of researchers have proposed various mechanisms to implement aggregation policies. These usually involve keeping some sort of history of each user's access, and granting or denying access to a member of the aggregate based on that history:

- In the SeaView system [LUNT89a], data is stored high and selectively downgraded according to the requester's past access history.
- In the LDV system [STAC90], data is stored low and access to it is selectively restricted based on its access by low users.
- In the Brewer-Nash model [BREW89] and its generalization by Meadows [MEAD90a], data is stored at different levels and access is granted to levels based on the past access history of the user or of a set of users. In Meadows' model, histories may also be kept of devices and other environments to which the data may be exported.

A problem closely related to aggregation and often confused with it is one commonly known as the "data association problem." This occurs when two data items may not be sensitive, but their association is. The most commonly cited example is the one in which names and salaries are considered nonsensitive, but the association between a name and a salary is. One way of dealing with this problem is to treat it as an aggregation problem; that is, to give a user access to names or to salaries, but not to both. However, as Lunt has pointed out [LUNT89a], what is really sensitive in this case is not the combination of a list of names and a list of salaries, but the association between individual names and individual salaries. If there is a way of keeping the association hidden, then there is no reason to prevent a user from learning names and salaries. Of course, as Lunt also points out, hiding such an association is by no means trivial. As a matter of fact, much of the work that has been done in statistical database security has been devoted to preventing inferences about such associations. However, the recognition that a particular aggregation problem is really a data association problem in disguise allows us to explore other techniques for dealing with it that may be more convenient to implement.

## **Conclusion**

In this essay, we have identified various approaches to inference problems in databases. We have described some of the specific inference channels that can arise, and have outlined the various approaches to eliminating them. We have also described some general models of the inference problem in databases, as well as some tools and methodologies that implement these models. We have also presented some of the various approaches to aggregation problems, which are related to but not identical to inference problems.

Inference problems are an important but still relatively unexplored aspect of database security. However, the works surveyed in this essay show a number of promising ways of attacking these problems and a number of interesting angles from which they can be approached. A complete and general solution to the inference problem is impossible. However, in the future, given a good understanding of one's problem area and the ways in which inferences are made, it should be possible to construct a database that is both usable and reasonably secure against inference.