SECURITY OF DISTRIBUTED DATA SYSTEMS

by

STEVEN D. FINCH

BGS, University of Nebraska Omaha, 1976

A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

Approved by:

Major Professor

Table of Contents

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Background

One of the most significant aspects of the future of information processing systems is the increasingly complex problem of securing and controlling information in a distributed database system. With the growing proliferation of comparatively easy-to-use microcomputers, there has been a dramatic increase in the conditions that nuture undercontrol and heighten exposure to security risks. It is common knowledge that corporate losses resulting from the accidental and intentional misuse of computers are growing. Current estimates of losses from computer fraud in financial institutions alone range from $500 million to $1 billion. Such losses should shoot over $7 billion by 1986 [LEVE 82].

Of greater concern is the growing potential for single instances of massive losses as corporate resources become accessible to a rapidly growing number of dispersed office computer users. According to a recent report, the average computer larcenist nets $450,000 to $620,000 per incident, while the average bank robber nets around $3,200 [LEVE 82].

There is obviously a pressing need to implement more stringent corporate-wide computer security measures.

1.2 Report Overview

As databases become larger and proliferate throughout industry and government, we will see a steady growth in the amount of sensitive

information stored in a computer system. If we expect to allow sharing of that database, we must provide a secure system to prevent misuse, loss or manipulation. Unless a user is assured that his sensitive data is protected from unauthorized access, he will not allow it to be stored in the system. Furthermore, society will not tolerate violation of a citizen's privacy because of non-secured databases.

This report analyzes distributed database systems and discusses possible vulnerabilities in the areas of access control, inference access control, integrity of data and communications security. Emphasis is placed on those controls that can be implemented in software. Hardware, physical, environmental and procedural controls are mentioned when they are used to supplement software controls.

CHAPTER II

ACCESS CONTROL

2.1 Introduction

About 60 percent of the reported abuses to computer systems are thefts or embezzlements by a trusted employee who misuses his access to the computer. Technology has progressed to the point of providing reasonable protection from the outside penetrator. This chapter will therefore address the aspects of access control of data to authorized system users. In other words, if user "A" wishes to obtain a data element owned by user "B", permission must have been granted by user "B", even if the data is resident in the same database. Access controls must regulate the reading, changing and deletion of data and programs. These controls must also prevent the accidental or malicious disclosure, modification or destruction of data by errant or deceptive programs.

2.2 Password Access Systems

The most common method used historically in control of access to a computer system and its resources is by using passwords. There are three basic password schemes for user identity--by something he knows or memorizes, by something he carries or by a personal physical characteristic. Prevalent schemes include single or fixed passwords, changeable passwords, random passwords, functional passwords (categorize a user according to his security classification) and extended handshake [HOFF 73]. The method by which the operating system handles the comparison of the user-supplied password with the authorized password on

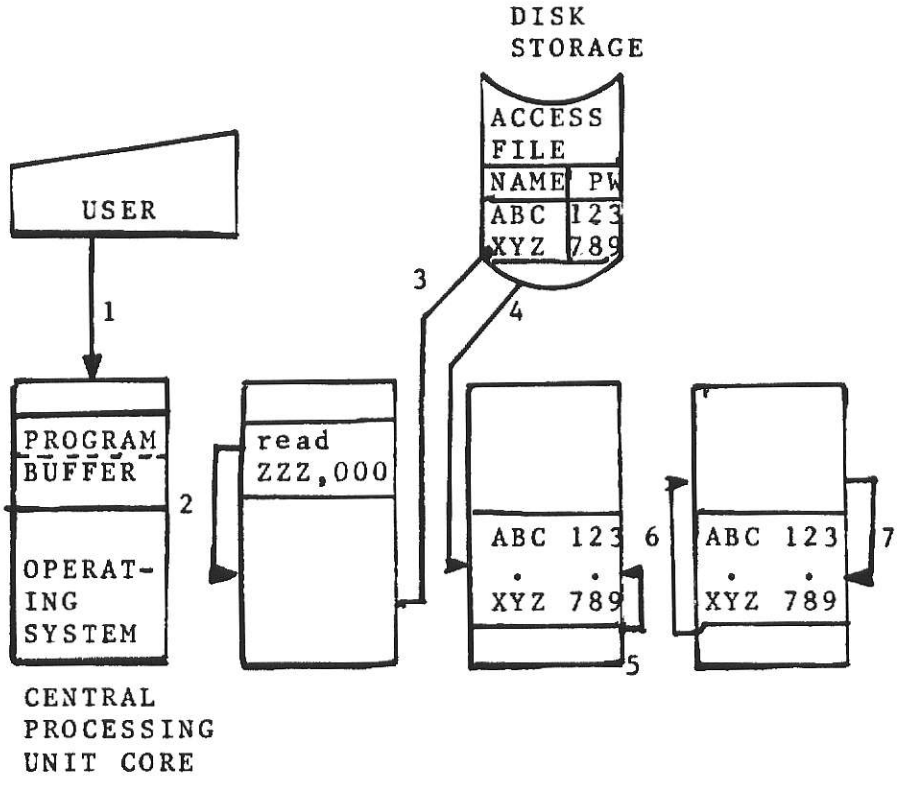
the authorization file is critical. Many systems write the authorization file into the user-allocated memory and then search this list for a matching password to a particular data set (data file). However, because the information remaining on the shared area (the user area) has not been overwritten, the current user now has access to the other users' passwords. This problem has been traditionally called implied sharing (Figure 2.1). Another type of implied sharing exists when registers that are accessible to user programs are used for the password comparison. Systems using this method do not expose the entire file to compromise; but if the registers are not overwritten or cleared, subsequent processes could read sensitive passwords. Many contemporary operating systems have resolved the implied sharing problem by performing the password search and comparison in memory accessible only by the system supervisor and/or by refreshing the memory used, by overwriting it with a set pattern of characters (ones or zeros or alternate ones and zeros).

There are many effective user identification means now commercially available to replace passwords in sensitive systems. These identification systems using handprint/fingerprint and voiceprint provide a higher level of security but at a cost prohibitive for most systems. Passwords are, therefore, by no means an outdated method of access control in many systems. They will provide the system reasonable security if properly handled externally by systems users and internally by the hardware and software.

In database management systems, it is sometimes necessary to protect selected data elements. Therefore, if we were to recount the passwords used to access that data element, we would need four passwords

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.



1. An unprivileged user requests a given amount of core. 2. The user requests to read file ZZZ and enters an arbitrary password 000. This file does not exist but would be the last file in the sort sequence. 3. The operating system must check the access file to see if he user has access to file ZZZ. 4. The system copies the access file into the users buffer area. 5. The operating systen searches for file ZZZ. 6. Failing to locate file ZZZ the system notifies the user and control is returned to the user. 7. The user reads his buffer obtaining file names and passwords.

FIGURE 2.1 Implied Sharing of Privileged Data

(Figure 2.2). If we now expand this database management system to a distributed system, we could envision a doubling, tripling, or quadrupling of the number of passwords needed. This would defeat the availability goals of our DDBMS and probably encourage the systems users to devise ways to thwart the security system.

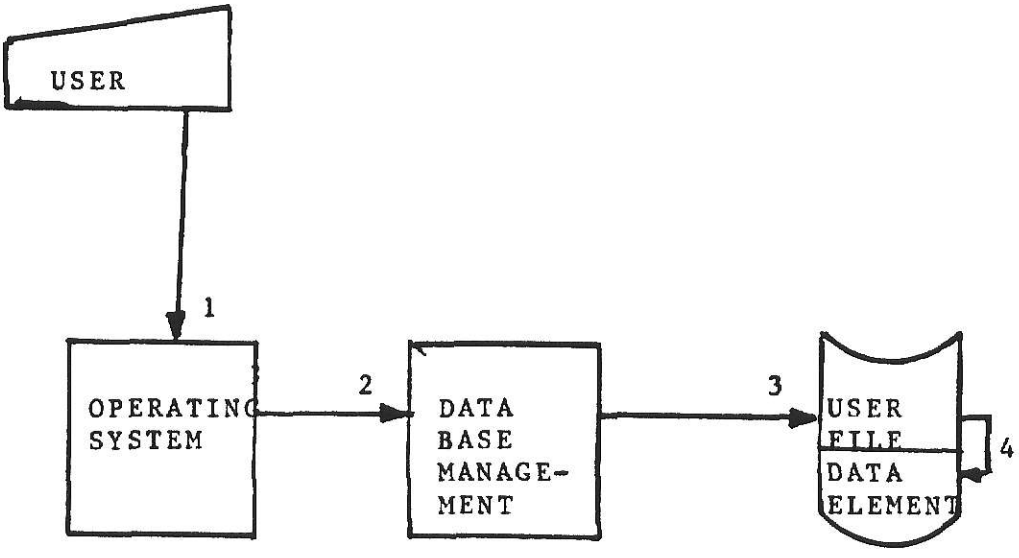
This gross proliferation of passwords would also greatly increase the probability that external disclosure would occur. Computer systems auditors and security evaluators have determined that one of the greatest vulnerabilities to penetration by an outsider is the availability of passwords that are not properly secured [ABBO 79]. Some of the common misuses of passwords include:

1. Passwords taped to the terminal
2. Passwords left in an unattended desk drawer
3. Passwords written on a paper in the purse or billfold
4. Passwords taped under the desk drawer
5. Passwords written inside the terminal users guide

These noted problems refer to systems where the user supplied only the password to enter the system. If we expand that by only three passwords, it is readily apparent that prevention of disclosure is difficult. If the passwords are randomly generated (i.e., PJ54C6U), one can see the difficulty in remembering three or more and applying them to the correct situation. The obvious solution is to develop an alternate scheme for user recognition and granting of access.

2.3 Transaction Processing Controls

The commands issued by the user of a transaction processing system are calls on a small library of transaction processes that perform specific operations, such as querying and updating on a database. In



1. At the system entry access point.
2. At the DBMS access point.
3. At the user file request.
4. At the data element level.

FIGURE 2.2 DBMS Access Control Points

such a system, the only authorized processes are the certified transactions. Therefore, it is possible to enforce the rules of access at the interface between man and machine. A user can identify a set of records by a characteristic formula 'C' which is a logical expression using the relational operators (=, >, <, etc.) and the boolean operators (AND, OR, NOT). These operators join terms which are indicators of values or compositions of relations. An example is:

C = "Army Officer and Graduate Student or (Status = Full time).

The transaction control program looks up a formula R specifying restrictions that apply to that given user. It then proceeds as if the user had actually presented the formula and R [DENN 79]. The concept of adding user restrictions, or privileges if you will, to the user request is common in database management systems.

When the system allows owners of records to revoke privileges that may have been passed around among users, it must be designed to revoke also any privilege that emanated from the revoked privilege. It is not difficult then to imagine the amount of communication that may be required in a distributed system.

In a distributed database system, a very critical but basic decision must be made concerning where the evaluation for access is made. If the evaluation is made at the node where the user entered the system, the transaction plus its appended authorization may be transmitted on the communications system. (For example, user A requests the number of items shipped to location ABC. This data is stored at a non-local node; the transaction might look like this: 'list,no, where item = widget and loc = ABC'). The local system would send this transaction to the node containing the data with the user I.D. and

authorization appended (i.e., 'A,OK'). So the entire transaction might appear as 'list,no, where item = widget and loc = ABC: A,OK;'. This would allow the perpetrator with a piggyback terminal (Figure 2.3) to monitor the authorization codes. This is potentially a very dangerous situation, but could be resolved by verifying access at the local node and then transmitting the transaction with an appended intermediate code that could be changed dynamically. This transaction would then be reevaluated at the node containing the data elements to be acted upon. The most casual reader will soon recognize the communications and synchronization required to maintain this type of access system. An alternate method would be to strip off the transactions that could be performed at the local node and grant authorization as appropriate. The remaining transactions would be communicated to the nodes containing the requested data for authorization. While this seems the simpler way to handle the problem, the piggyback terminal operator will soon realize that the transaction with an appended authorized I.D. will gain access to selected data elements. Then penetration is a simple matter of imitative deception. This method of handling access control could also make enforcement of inference access more difficult. This problem will be discussed further in Chapter III.

Unfortunately, transaction processing controls are traditionally implemented on a general purpose computer and are vulnerable to manipulation via the operating system. Most security flaws in existing systems are the consequences of design shortcuts taken to increase the efficiency of the operating system.

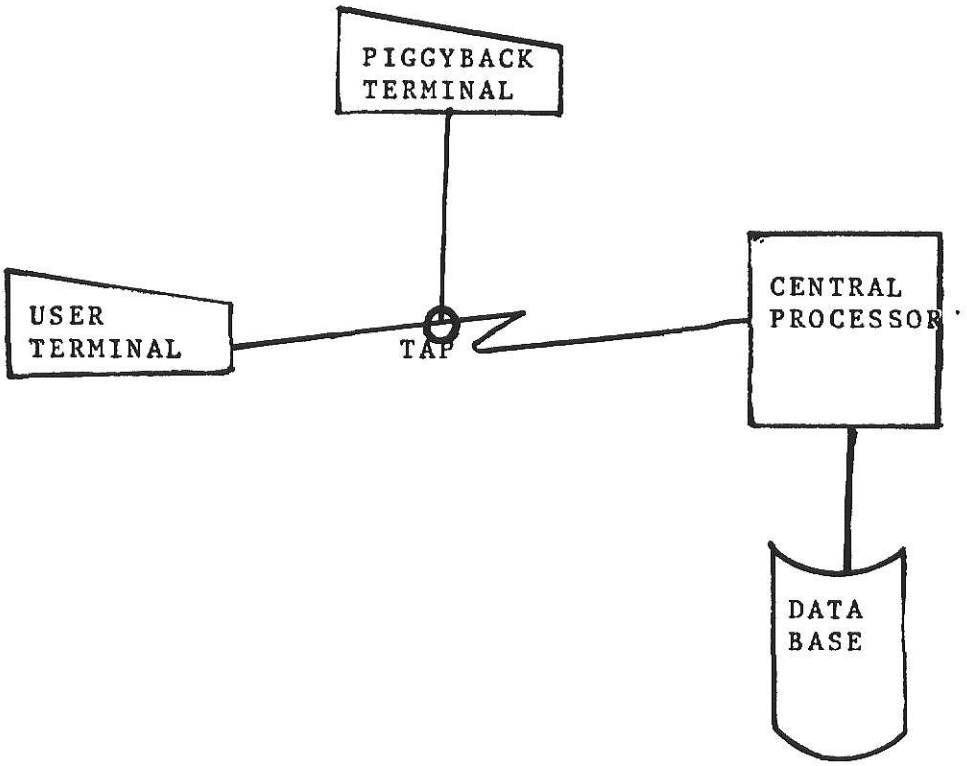


FIGURE 2.3 The Piggyback Terminal

2.4 SYSTEM R*

SYSTEM R* is the distributed version of IBM's SYSTEM R. Access control is via a modification of the transaction processing controls discussed in the previous section. Authorization rights must be refined so that different users have different access rights on the same database. IBM's designers quickly dismissed the use of passwords to accomplish this.

The mechanism used goes beyond the traditional password approaches. Its highlights are enumerated here:

1. Passwords are not used to control access rights to specific objects.

2. The system figures out when processing a user's access request whether the user is authorized to access the specified object.

3. Users access rights to specific objects can be dynamically modified.

4. To each object defined in the database is associated the identification of users who have specific access rights on this object.

5. Users can be allowed to grant and revoke access authorization on objects they are authorized to access.

6. The mechanism of access rights allocation is not based upon a rigid hierarchical method of access rights grant, rather control is distributed among authorized users.

The advantages of this mechanism include its flexibility, a high level of privacy, and ease of use. Another attractive aspect is the complete distribution of control over access rights. There does not need to be a centralized database administrator holding all power and granting access rights to specific users.

The authorization mechanism basically performs three kinds of operations:

1. Checking the ability of a SUBJECT to access an OBJECT
2. Granting access rights on an OBJECT to a SUBJECT
3. Revoking access rights on an OBJECT from a SUBJECT

One of the basic OBJECTS in SYSTEM R* are relations which are sets of n-tuples; a relation is conceptually represented in the system as a table. There is a special relation kept by the system, called the authorization table, which records the state of user's access rights of the relations. For each relation currently existing in the database, there exists at least one entry in the authorization table.

Privileges which may be exercised by each subject fall into three basic categories:

1. Privileges on relations: These are the normal database query authorizations (i.e., SELECT, INSERT, DELETE, UPDATE, EXPAND, INDEX).

2. Privileges on a program: These are defined as privileges granted to a program at compile time (i.e, RUN).

3. Special privileges: This encompasses special authorizations called resource authority and database administration (DBA) authority.

    a. Resource authority: This is required to create tables and acquire segments (logical storage space) because these operations use up space in the database.

    b. DBA authority: It is the highest level of authorization and gives its owner the capability to use any privilege on any relation and to run programs; DBA authority also includes resource authority. The only power restriction imposed to DBA consists in the interdiction for a DBA to grant or revoke privileges he uniquely holds due to DBA authority.

Privileges on relations and programs may be owned with or without grant option; a privilege obtained with grant option allows its holder to transmit this privilege to other subjects. Transmission of a privilege is performed by a GRANT statement. The GRANT statement has the form: GRANT <privileges> ON <object_name> to <subject_list> (WITH GRANT OPTION). The grant statement runs successfully if and only if the

grantor holds all the privileges he wants to deliver, with grant option. However, he may hold them either personally or he may belong to a set of users holding these privileges with the grant option. If the privilege is transmitted to the grantee without the grant option, the grantee is only allowed to use the privilege; he may not grant it to others. Any subject who has granted a privilege may later withdraw it by issuing a revoke command. The privileges held on the object will then be withdrawn unless the revokee had also acquired the privilege from another grantor. Conversely, a user is not allowed to revoke a privilege from a subject to whom he never granted it. The format of the REVOKE statement is:

REVOKE <privilege> ON <object_name> FROM <subject_list>.

SYSTEM R* provides the users with the capability for a more secure database because of the ease in granting and revoking privileges to objects. All objects in SYSTEM R* are protected when created and initially only the creator has access rights. Like other transaction processing systems, SYSTEM R* is subject to the vulnerabilities of the communications system and host operating system.

2.5 The Database Machine

The use of a backend processor or database machine properly implemented may greatly increase database security. There are a number of advantages resulting from the physical separation of the DBMS and the rest of the system. When the DBMS and the rest of the system are separate entities, it is possible to specialize each for their given task. The DBMS need not have a general purpose operating system; one

which is tailored to serve just the database management function will suffice. The host computer's general purpose operating system at each node need not support database management functions; it must only provide a mechanism to channel data to and from the DBMS. Thus, both components and the interface between them are simpler than would otherwise be necessary and so the difficulty of their construction is reduced somewhat.

The physical separation of the DBMS from the rest of the system should enhance database security. Since the DBMS and the rest of the system are connected by a single, well-structured link, assaults on the database via the host computer's operating system are virtually eliminated.

2.6 LADDER

LADDER (Language Access to Distributed Data with Error Recovery) is a distributed database management system developed at SRI for the Advanced Research Project Agency of the Department of Defense. The goal of the system is to provide the user easy access to information stored on multiple computers, under various database management systems. A graphical overview of LADDER is presented at Figure 2.4.

The system employs a natural language front-end called INLAND (Interactive Natural Language Access to Navy Data). While the primary function of this component is to provide the user with the ability to ask questions about the database in a natural language (i.e., English with an expansion to include Navy terminology), it also provides additional security within the system by isolating the user from

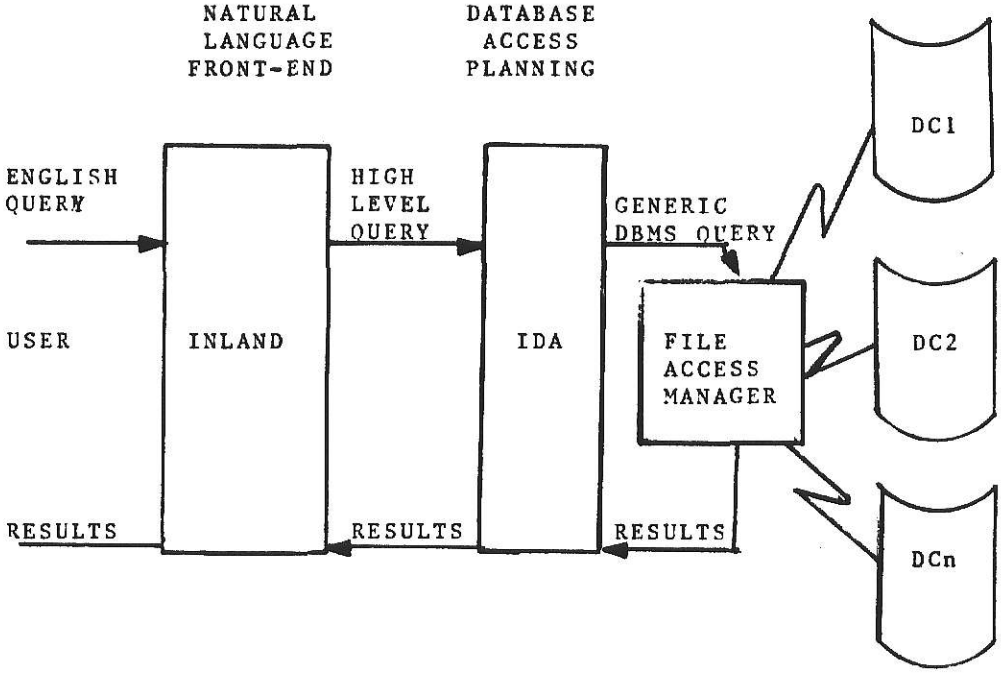


FIGURE 2.4 The LADDER System

database information such as location, and file and record structures. This is accomplished by the simple conversion of the user's query into a high-level query to the next component of the system called IDA.

IDA (Intelligent Data Access) translates the query received into a query program to be issued to the database management systems. To accomplish this, IDA must have access to a data dictionary or a description of the database structure. One query to IDA may generate several subqueries that may require access to multiple databases. IDA is not given the precise location of each file. In fact, IDA is led to believe that all the files exist on one computer. It, therefore, uses generic filenames and the next component must select the appropriate computer.

The generic DBMS query is received by the FAM (File Access Manager). This component is responsible for the connection of various computers finding the most recent versions of pertinent files, issuing the actual DBMS's queries and recovering from errors. Because of the structure of FAM, a system could be implemented by-passing both INLAND and IDA. While this would make the system cost much less, the two levels of transparency provided would also be lost and security lessened.

2.7 Conclusions

All methods of access control in a distributed database system must necessarily result in the transmission of the authorization in some form over the communications system. This transmission is subject to monitoring by unauthorized persons, and systems penetration could

result. There are ways to protect that transmission through encryption of the transaction or communication link. The methods of encryption will not be discussed in this paper. Therefore, the interested reader should consult one of the following references for more information [DOOL 78, McGL 78, ROTH 77].

Passwords are an effective means of controlling access at the man-machine interface point, but beyond that point they are less effective due to the inadequacies of current operating systems and the probability of compromise due to human nature.

Transaction processing controls or variations of this basic concept provide the best possibilities for reasonable security. Access controls can be implemented in software, hardware or firmware and be controlled so as to not allow manipulation by unauthorized users. Additionally, a greater level of transparency is available with transaction processing type controls.

Backend processors or database machines provide additional security because of the dedication of those systems to the task of controlling the database, and the isolation of the database from the general purpose operating system.

CHAPTER III

SECURITY IN STATISTICAL DATABASES

3.1 Introduction

The problem of enhancing the security of statistical databases has attracted much attention in recent years. This problem can be stated as follows. A statistical database is a collection of records that contain information on some number n of individuals. Each record contains confidential category and data fields; at least two values exist for each such field. The category fields are used to select and identify records, while the data fields contain the information. Given a statistical database containing information about a population of individuals, this database should provide users with statistical data such as totals, median, counts, etc. for groups of individuals without compromising the data on any individual. Compromise can occur by asking for the average of an element where there is only one response set qualifying or by asking two or more related queries resulting in a response set size of one when the queries are compared. The technique is to form queries so that responses will provide information on one individual without a direct query for that information. A compromise is positive if the questioner deduces the value of a given category or data field of an individual and negative if the questioner determines that the individual is not in the selected category or data fields. This problem is complicated by the possibility that the user could ask a large number of questions of the database by submitting a series of related queries. To compound this problem further, if the user has access to some of the data via public or other means (i.e., sex, age,

marital status, etc.) or confidential (salary, etc.) information about certain individuals, he can use this knowledge to frame queries to obtain confidential information on others.

This chapter will give an overview of both problems associated with and the prevention of those problems. It will attempt to present some additional issues that are incurred due to the distribution of a statistical database.

3.2 The Problems

A database is said to be compromisable if any protected element of data for an individual can be determined exactly by one or a series of queries. If the answers are distorted, this exact determination is not possible. However, a user can obtain statistical estimates of the data. The larger the number of queries, the more precise the estimates. Under this definition, disclosure cannot be prevented. It can, however, be controlled.

A few examples will serve to illustrate some of the difficulties. The database (Figure 3.1) is assumed to contain personal information such as age, employer, education, and salary and we are allowed to request totals, counts and averages.

Example 1. If we know Mr. Keys' age, education and employer, the following query could be used: List the average salary of all persons where age is 39, employer is the ABC company and education is a Masters in Computer Science at Kansas State. If Mr. Keys is the only person who meets all of the criteria of the query, the exact salary will be returned.

ABC CO. PERSONNEL FILE

Name	Age	Employer	Div	Education	Salary
KEY	39	ABC CO.	FIN	MS COMPSCI KSU	$34,000
JONES	39	ABC CO.	FIN	MS COMPSCI KSU	$36,000
SMITH	32	ABC CO.	ACCT	BS BUS IOWA ST	$22,000
CORD	34	ABC CO	FIN	BS BUS KU	$25,000
DODD	33	ABC CO	ACCT	MBA U-TEXAS	$31,000

Figure 3.1
Sample Database

Example 2. Many authors have suggested restricting the size of a response set (i.e., require that the response be composed of information on 2 or more persons). Then the following two queries would result in a compromise.

a. List the total salary for all persons where age is in the range 39-50 and employer is the ABC company and education is a Masters degree in Computer Science at Kansas State.

b. List the total salary for all persons where age is in the range 40-50 and employer is the ABC company and education is a Masters degree in Computer Science at Kansas State.

Now subtract the results of query b from query a and you have Mr. Keys' salary.

Example 3. If the query in Example 1 resulted in a response set of size two, anyone knowing the salary of one could mathematically obtain the other (i.e., if Mr. Jones wants to know the salary of his associate Mr. Keys, knowing they are the same age and have the same education, the query in Example 1 and some simple math will provide the correct amount of Mr. Keys' salary.)

Example 4. If we ask the following queries:

a. List the number of individuals where age is in the range 39-50 and education is Graduate Degree in Computer Science from Kansas State and salary is in the range of $30,000 to $35,000.

b. List the number of individuals where age is in the range 40-50 and education is Graduate Degree in Computer Science from Kansas State and salary is in the range of $30,000 to $35,000.

If the answers to these questions differ, we have determined that Mr. Keys makes between $30,000 and $35,000 per year. If more precision is desired, you could perform the queries again and reduce the salary range.

In all of these examples, we have been able to obtain salary information on Mr. Keys without using his name. There are, of course, many other possibilities to compromise the data through inference. One method is by use of a technique called trackers.

3.3 Trackers

Statistical databases can be easily compromised even if some queries are not answerable because their query sets are too small. The questioner divides his preknowledge of a given individual into parts, which are then reassembled into a special characteristic formula called a tracker. From the responses of a few answerable queries involving the tracker, the questioner may determine whether or not the individual has a characteristic previously unknown [SCHL 75].

Trackers were further defined by [DENN 79] into three types:

1. Individual
2. General
3. Double