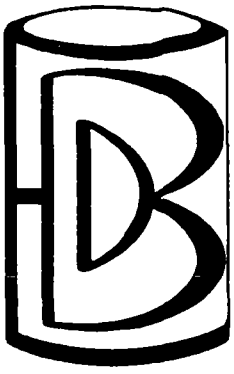


Data Base Engineering

Vol. 1 No. 2 **June 1977**

Contents

A Quarterly Bulletin published by the IEEE Computer Society Technical Committee on Data Base Engineering

Data Base Engineering Bulletin

A Quarterly Publication of the IEEE Computer Society
Technical Committee on Data Base Engineering

Data Base Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Data Base Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, data base architecture, data base machines, intelligent front ends, mass storage for very large data bases, distributed data base problems and techniques, data base software design and implementation, data base utilities, etc.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews and summaries, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Data Base Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in Data Base Engineering Techical Committee is open to IEEE Computer Society members, student members, and associate members. (Application form in this issue.)

FROM THE CHAIRMAN

I should like to say a few words in the Bulletin before my tenure as the Chairman of the TC on Data Base Engineering expires in June.

The idea of starting some data base activities within IEEE was due to Steve Yau. In fact, it was Steve who then as the President of the IEEE Computer Society established this Technical Committee on Data Base Engineering in the fall of 1975. I was asked to draw up a charter and solicit members. Included herein is the list of founding members of the TC.

Mr. Charles Bachman
Honeywell Information Systems

Professor P. Bruce Berra
Syracuse University

Professor Robert Carlson
Northwestern University

Professor Yaohan Chu
University of Maryland

Dr. E. F. Codd
IBM Research Laboratory

Dr. Murray Edelberg
Sperry Research Center

Professor Michael Hammer
Massachusetts Inst. of Tech.

Professor Martin Hellman
Stanford University

Professor Keki Irani
University of Michigan

Professor Douglas S. Kerr
Ohio State University

Professor T. L. Kunii
University of Tokyo

Professor David Lefkowitz
The University of Pennsylvania

Dr. Vincent Lum
IBM Research Laboratory

Professor Stuart E. Madnick
Massachusetts Inst. of Tech.

Mr. Frank Manola
Naval Research Laboratory

Dr. E. J. McCauley
Aeronutronic Ford Corp.

Dr. Harold Schwenk
BGS Systems, Inc.

Professor Diane Smith
University of Utah

Professor S. B. Yao
Purdue University

Professor Eugene Wong
University of California

One of the common concerns among the founding members was the possibility of unnecessary rivalry with other similar organizations. It was agreed at the outset that the TC should work with others in sponsoring technical activities, that the TC should emphasize its members' strong points in data base research and development (e.g., the engineering aspects), and that the TC should specialize in new and emerging data base areas. Although the TC is not yet two years old and has only a membership of about one hundred, we have been striving toward such goals. Let me report to you some of the important activities which have taken place.

(1) TC becomes a permanent sponsor of the International Conferences on Very Large Data Bases. The second such conference was held in Brussels last year and the third one will be held in Tokyo in October. The fourth one will likely be held in Berlin next year. Other sponsors include

ACM's SIGMOD, SIGIR and SIGBDP, making this the first truly joint data base activity between IEEE Computer Society and ACM (see the VLDB announcement in this issue of the Bulletin).

(2) TC becomes one of the first organizations to encourage the study of common issues affecting operating and data base systems. A joint workshop with TC on Operating Systems was held in Chicago in March this year. Based on the quality of the presentation and discussion and the number of the attendees, I must say that it was successful (see a report on the workshop in the issue).

(3) TC attempts to promote both hardware and software work in data bases. For the COMPCON Spring 76, two sessions on "Where are Data Base Systems Heading?" and "Can Data Base Machines Be Built?" were organized. For the COMPSAC 77, there will be many data base sessions. In addition, a special collection of selected data base papers from COMPSAC 77 will appear in the March issue of Transactions on Software Engineering.

(4) TC tries to get more members interested in data base machine work. It is an area that engineering-oriented members can pursue with some confidence. Not only was there an article on this topic in the last issue of the Bulletin, but I also include a note of my view on this topic in this issue.

(5) TC inaugurated its quarterly news letter, the Data Base Engineering Bulletin, in March of this year.

In closing, I may say that the TC is doing well. Vincent Lum, who will be our new TC chairman has great plans for us. I'd like to take this opportunity to welcome him aboard and wish him every success. I must thank Steve Yau for getting us started, Jane Liu for inaugurating the Bulletin, and others for organizing the workshop and conferences. As an "old" TC member, I shall continue to serve TC for data base advancement.

TC/DBE MEMBERSHIP APPLICATION/RENEWAL FORM

To become a member of the TC/DBE and be on the mailing list for the Data Base Engineering Bulletin, please return this form or a copy of it to:

IEEE TC/DBE
Department of Computer Science
University of Illinois
Urbana, IL 61801

NAME _____
(please print)

INSTITUTION _____

ADDRESS _____

_____

Areas of Interest: _____

Level of Participation:
( ) Read newsletter only.
( ) Work on workshops and Symposia.
( ) Help with newletter.

REVIEW OF THE WORKSHOP ON OPERATING AND DATABASE MANAGEMENT SYSTEMS

March 21-22, 1977
Northwestern University
Evanston, Illinois 60201

by

C. Robert Carlson
Northwestern University

Abstracts of the papers presented at this Workshop have been published in the March, 1977 issue of Data Base Engineering, Vol. 1, No. 1. This brief review does not do justice to each of the papers but seeks to identify a few of the major topics around which discussion centered.

Most current systems exhibit conceptual structures in which the execution of database management operations requires the invocation of operating system provided service functions (e.g., input-output, file access, secondary storage management, protection). Discussion focused on the following two areas:

(1). What service functions need to be provided by the operating system on a conventional computer architecture. As an example, the relationship between System R and its operating system was explored.

(2). In a distributed computing environment, there are disadvantages to employing a system structure in which the DBMS is dependent on a centralized OS. Alternative system structures were discussed in which either the DBMS and OS are independent or the OS is dependent on the DBMS for many of these same service functions.

Most current systems possess unnecessary duplication of function, often resulting in uncoordinated strategies being employed. Security and storage management are two examples. After surveying existing OS and DBMS security mechanisms, several security techniques were described in detail. These included trigger subsystems, kernel design of software, referencing monitoring, cryptography, probabilistic modeling and defining safe transaction sets. Finally, several papers presented performance evaluation techniques and raised performance issues related to the OS/DBMS interface.

The Workshop concluded with a (large) round-table discussion in which numerous OS/DBMS interface problems were discussed and the research progress in several of these areas summarized.

DATA BASE COMPUTER - WHY AND HOW

David K. Hsiao

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

- **What is a database computer?**

The database computer is made of specialized computer hardware which supports basic database management functions found in most contemporary software database management systems.

- **Why do we need hardware to perform basic database management functions when most such functions are available in software?**

For reliability - Database management software grows in complexity and size. This growth is prompted by the increase in user database management requirements and the recent change of data processing mode from an off-line, batched, single-user environment to an on-line, concurrent and multi-user environment. Large and complex software systems tend to be failure-prone. Furthermore, practical verification methods for software systems are still not in sight. On the other hand, methods for verifying hardware functionality, design and production have long been available. Advanced technology has also overcome some of the problems of the logic complexity and capacity requirements, making the construction of relatively large and complex computers viable. By incorporating basic database management functions into hardware, not only can we provide more reliable basic functions, but we can also improve the software reliability since the software requirements will be less complex and the system software will be smaller in size.

For performance - Conventional computer systems were not designed for database management. These von Neumann-type computers are good for the preparation and execution of programs for numerical computations and for simple data processing. Database management activity on the other hand is concerned with the storage, retrievel and management of large databases and requires quick search and good update operation for concurrent access. For example, it is well known that the execution of an IMS data management call on an IBM 370 computer takes an average of 130 cpu instructions for preparations of eventual I/O. Thus a typical conventional computer spends much of the time interpreting data management calls instead of executing them. Consequently, the response time for a call is degraded. By relegating the database management functions to specialized hardware (i.e., the database computer), the response time can be improved. Furthermore, the von Neumann-type computer can concentrate on its traditional role of program preparation and execution with freed-up cpu cycles. Both the general-purpose conventional computers and special-purpose database computers can then yield high performance.

- **If hardware specialization is such a viable solution to unreliable and low performance database management software, why are no database computers available?**

Until recently the main roadblocks to the realization of database computers have been the lack of adequate database research and hardware technology.

First, there is the immaturity of the database research. Database practitioners were overwhelmed by the continuous demand for larger and more complex software systems; they have not been given enough time and support to consider hardware realization. Furthermore, database research became 'respectable' only recently with the publication of the two definitive works - Codd's 1970 paper on the relational database model and Codasyl's 1969 and 1971 DBTG reports on the network data model. Only in the last three or four years, have database researchers begun to consider seriously hardware realization as a viable solution.

Second, there is also the inadequacy of hardware technology. Database management functions require large on-line storage and rapid real-time search. Since associative memories can provide quick real-time search, considerable research was devoted to the study of associative memory and processor. However, early reliance on monolithic associative memories and processors has resulted in disappointment due to high cost and the limited capacity. More recently, efforts were made to rely on logic-per-track design in providing content-addressability. By incorporating logic capabilities in the read/write mechanisms of a fixed-head disk, we can achieve a limited form of associativity with reasonably large capacity. Although this approach is promising, it has several drawbacks. The fixed-head disk design has a capacity limitation of about 10^8 bytes. In order to overcome this limitation, the cheaper moving-head disk device would have to be used. In this case, it is desirable to separate structural information of the database (e.g., indices) from the database itself in order to minimize the number of accesses to slow moving-head disk devices. By storing the structural information in a faster (possibly electronic) storage medium, we can make frequent accesses to this information for the purpose of limiting the search space in the database stored in the moving-head disk device.

- Are the roadblocks toward hardware realization of database computers being removed?

There are several promising indications: (1) several emerging technologies such as magnetic bubbles, charged-coupled devices (CCDs) and electron beam addressed memories (EBAMs) are now available. They are likely to replace fixed-head disk technology. Because these new technologies can be configured for different access modes and access times, they are far more versatile and effective for a variety of data management tasks. For example EBAMs with memory capacities of 10^7 - 10^8 bytes and block access times of 5 μsec can be configured for storing and processing structural information of the database. For smaller structural information storage and management, both bubbles and CCDs are good candidates. Although none of technologies is currently being configured for database management hardware, they are being constructed for other special-purpose tasks - bubble memories for voice recording, CCDs for buffering and EBAMs for paging. In other words, these technologies are entering the production stage.

(2) New use of existing technology has given impetus to moving-head disks. In on-line storage technology, there is no immediate competition

to the moving-head disk in terms of cost and capacity. Thus, the moving-head disk is likely to remain the mainstay of on-line storage for years to come. However, there are novel ways to modify moving-head disks for high performance without incurring high cost. An example of performance improvement is to convert a disk to a device with content-addressability by cylinders. Utilizing microprocessors and conventional hardware, the conversion is fairly stratightforward. The relatively large access time of the moving-head disk is compensated for by the more intelligent search and processing capability of the new device. Furthermore, by relegating the structural information of a database to a faster device using EBMs, CCDs or bubble memories, we can minimize the number of accesses to the database itself. Consequently, the number of accesses to and the amount of processing by the modified moving-head disks can be greatly reduced.

(3) Some good research experience has accumulated in the past years from known projects on database machines. Although one of the projects is terminated and the second one just recognizes that they use the wrong technology, some good lessons have been learned. First, a database machine should not be designed to support only one data model, making it difficult to support other data models. Second, its design should not be based on a "dying" technology such as the fixed-head disk, making it difficult to migrate to other technologies. Third, it must be designed with a complete, although basic, set of database management functions in mind. The lack of security and clustering mechanisms, for example, in some of the earlier machines makes later add-ons of these mechanisms difficult.

• What are the requirements of a modern database computer?

The following requirements are indeed critical:

(1) Large storage capacity: The on-line storage capacity for the database store should be in the range of 10^9 - 10^{10} bytes. The on-line storage capacity of the structural information device should be in the range of 10^7 - 10^9 bytes.

(2) Intelligent search and update: Real-time search and efficient update can only be achieved with content-addressability. However, the degree of content-addressability may vary with the nature of the information that the database machine handles. For larger database stores, we can achieve content-addressing in milliseconds (the revolution time of a disk). However, the content-addressable unit must be large (say, cylinder size) and cost-effective For more frequently used structural information, we must have a system of faster, albeit smaller, content-addressable memories. To this end, the emerging electronic memories and microprocessors are possible solutions.

(3) Innovative architectural approach: Because the database computer is a special-purpose machine, it requires a non-traditional approach to its design and configuration. Three factors must be considered for the design and configuration:

(a) Functional specialization - A database computer must perform a series of specialized database functions which will be performed by different components. Unless one has a good understanding of these functions and their interactions, one will not be able to design a highly

parallel and pipeline-oriented computer for throughput improvement. Thus, thorough knowledge of database functions and algorithms for implementing these functions are necessary prerequisites of the specialization.

(b) Technology utilization - It is not possible to rely on the limited number of current technologies for building database computers. The solution to proper balance of architectural cost and performance lies in the utilization of new and the modification of old technologies. Thus, designers of database computers must also be knowledgeable of technology availability and developments.

(c) New Facilities - Many add-on facilities of existing database systems have not worked well. Examples of these facilities are security mechanisms for access control and clustering methods for performance enhancement. For a new database computer to be viable, the design of the computer must be complete in the sense that all the new facilities should be considered in the design at the outset and be integrated with the rest of the design.

(4) Multiple data model support: The new computer must compete favorably with conventional database management systems running on conventional general-purpose computers. In order to compete favorably, the new computer must also have new system capabilities in addition to its architectural and technological advances. Since there are at least three outstanding types of database management systems in the field (namely, hierarchical, e.g., IBM's IMS; network, e.g., Honeywell's IDS; and relational, e.g., IBM Research Lab's System R), the new computer must have built-in hardware data structures and facilities which will support these types of database management functions well. That is, the new computer must satisfy the following three criteria:

(a) The user's existing database application programs can be run on the new computer (via a software interface) without the need of any conversion.

(b) The software requirements of the new computer's interface must be minimal.

(c) The storage and processing requirements of the user's existing database applications should be cost-effective when carried out by the new computer.

(d) There should be new facilities that the database computer can provide and that the conventional computers cannot.

(5) No distant solution and technology: The need for a high-performance and low-cost solution to database computers is now. Thus, any database computer design should not rely on distant technology. Furthermore, it should provide a viable design which is realizeable in this or the next decade. Prototype construction and testing should not be a distant goal.

THE REFERENCE MONITOR TECHNIQUE* FOR SECURITY IN DATA MANAGEMENT SYSTEMS

Gillian Kirkby and Michael Grohn

I. P. Sharp Associates Limited
Suite 600, 265 Carling Avenue
Ottawa K1S 2E1 CANADA

I. Introduction

In the last few years, the subject of data security has been widely researched, due to the requirements of modern computer systems. Many systems have the responsibility for manipulating both classified and unclassified data in a multi-user, shared resource environment. Present computer systems are unable to provide adequate protection in such circumstances, since they cannot be proven correct. In the past, it has been left to a "tiger team" to find flaws in a system and patch them, but this unsatisfactory approach does not guarantee that the system is inviolable.

The work discussed in this paper is a theoretical solution to the problem of providing adequate protection to classified military information in systems where all users may access only a subset of the total data. The work is being carried out in consultation with the Mitre Corporation. Mitre has performed much research in the field of protection of automatic systems for military applications.

The technique utilized in our work is that of a reference monitor to act as a mediator between the system users and the data base. A reference monitor is a hardware and software interface which is guaranteed to maintain the security of the data base. The purpose of the hardware is twofold: it must ensure that the software is isolated and thus "tamperproof"; it must provide the only means of accessing the multi-level data base.

Our initial task was to identify the security policy relevant to the military environment and to build a mathematical model of a secure data management system (DMS) embodying these principles. Having established a satisfactory model, the subsequent state involved developing a structured functional design for the DMS; in particular specifying the software portion of the reference monitor, which is termed the DMS security kernel. The final work was the certification of this security kernel to guarantee its correspondence with the model.

The mathematical model [1] was developed out of existing security models, notably that of Bell and LaPadula [2]. The terminology was adopted and expanded to cope with new concepts and the application of the model to general relationally organized systems. The relational approach to data management was adopted to provide an implementation independent, theoretically based design: it will be discussed more fully in Section III.

* This work was performed as part of contract F19628-76-C-0025 jointly sponsored by the United States Air Force Directorate of Computer Systems Engineering, Deputy for Command and Management Systems, Electronic Systems Division, Air Force Systems Command and by the Canadian Federal Department of Industry, Trade, and Commerce.

One of the main differences between our mathematical model and that of Bell and LaPadula was the incorporation of an integrity mechanism as well as a security mechanism. The result of this was that the sensitivity of data base objects is measured by what is known as a protection level, the result of combining security and integrity levels. Section II explains the adoption of protection levels.

The design itself specifies a relational data organization with adequate software to provide a harmonious, protected data sharing environment. The abstract subjects and objects, which the model identifies, are interpreted as processes, acting on behalf of system users, and data, in relational form. All activity between these is determined by a special set of functions specified in terms of their effects. Of these functions, those that involve security considerations compose the DMS security kernel and it must be proven that they act in accordance with the access authorization policy presented in the model.

II. The Mathematical Model

The mathematical model consists of a set of symbols representing variables and functions, where a function is a rule associating an output value with an input argument. The goal of the model is to represent concisely the essential entities and actions inherent in secure data management, where secure means conforming to a security policy.

2.1 The Bell-LaPadula Model

The model is based on the Bell-LaPadula (B-L) model [2] of a secure computer system, and embodies the U.S. Department of Defense security policy. In the (B-L) model each process (user) and data object is assigned a security level consisting of a classification (e.g. SECRET, TOP-SECRET) and a set of categories (e.g. NATO, NUCLEAR). One level is said to dominate another if the classification of the first is higher than or equal to that of the second, and if its set of categories include those of the second. The security policy requires: (i) the level of a process to dominate that of the data it observes, (ii) the level of any modified data to dominate the level of any observed data , (iii) explicit (discretionary authorization for every data access, (iv) and the level of an active data object to remain constant.

2.2 Extensions

Security levels were considered adequate for the authorizations of data observation, but improved data modification authorization required the establishment of dual integrity levels, and the definition of a composite form of level dominance. The result was increased access control without changing rules (i) and (ii) (2.1). More specifically, a protection level is defined to consist of: a security classification, a security category set, an integrity classification, and an integrity category set. One protection level is said to dominate another if: its security classification is equal to or higher than that of the other; its security category set includes the other's categories; its integrity classification is equal to or lower than that of the other; and its integrity category set is included in the other's categories. Alternately, a level is said to be higher if it dominates another, and lower if it is dominated by another.

The hierarchial directory structure (in B-L) was replaced by a flat-file organization of directories, where one directory contained identifiers of ALL data at a certain level. This made data access more convenient, and allowed space quota management to be performed on a user basis, rather than on an object and subordinate object basis. A data object was decomposed into three components: an access permission matrix, a set of values, and a component defining the format of the set of values.

III The Design

3.1 An Overview

The mathematical model established the basic features of the data base; namely the protected data objects and the directories in which object identifiers are recorded. At the design stage, the means of controlling sign-on to the data base was addressed and directories of active users (sign-on lists) were defined in the data base.

Access and manipulation of these entities must obey the rules of protection developed in the model. To this end, a set of primitive functions was developed. It is these primitives which form the DMS security kernel. From these functions it is possible to build subroutines which handle all reasonably conceivable data base manipulations. If each primitive is quaranteed to enforce the protection policy, then the subroutines must necessarily conform to this policy.

The choice of a relational data base organization meant that data objects are established as relations and that each relation has a specific protection level. Data base users sign-on to the data base at a particular level and may observe data at all lower protection levels. Relational operations may then be applied to this information, in order to respond to queries etc.

It was decided, therefore, that each data base user would be provided with a working area to be used as a computerized "scratch pad." This area would be one into which data objects could be copied and relational manipulation performed without affecting other data base users. Because this working area, denoted W, is associated with an individual user, all data copied into it is considered to exist at that user's current protection level. To transfer data from the data base into the working area the user must have discretionary access authorization to observe that data and have a current level which dominates the protection level of the data. A user's W area is private and he is virtually at liberty to do whatever he wishes to the data in it; however, he may not transfer this data back into the data base at will, since each such transfer constitutes a modify operation.

3.2 The Data Base

Three classes of entities exist within the data base: directories; sign-on lists; data objects. Non-discretionary protection levels are associated with all three groups, though only data objects have a discretionary protection mechanism.

A data object is established in the data base by a specific DMS user. That user is considered to be the owner of the information and his unique user identifier is incorporated as part of the object identifier. The information input may be either a standard relation or one of the special types to be discussed subsequently. Object type is indicated in the identifier, which is the quadruple:

OWNER ID; OBJECT NAME; OBJECT TYPE; PROTECTION LEVEL.

It is this identification which is recorded in the directory at the appropriate protection level. This is termed the definition entry for the object, as provision is also made for registering object identifiers in directories at dominated protection levels. Thus directories are also relations, consisting of identification tuples for objects.

The sign-on lists are utilized to provide current information on which data base users are active. A user signing on at a particular protection level will be recorded in the sign-on list corresponding to that level. Sign-on lists are somewhat similar to directories except that the tuples are more volatile and are deleted when the user signs off.

One special type of relation, recognizable to the system, is the message string. A string is a single tupled relation, where each domain constitutes a field with the message. The other specially identified relational type is the program, which is considered to be a multi-tupled, two domain relation. The format of this type of relation establishes a relationship between a set of instructions and an order of execution. The instructions consist of relational operations to be applied to specified relations in the system. The invoker of a program must have access to both the relation storing the program and the operand relations which it uses.

In Figure 1, examples of a standard relation, a view and a string are given. It may be noted that the formats for all these types are themselves formatted identically.

In order to support the discretionary protection of these relations and, additionally, to provide some useful data management information, a number of minor component entities are associated with each data object in the data base. A permission matrix contains the information on which users have which discretionary access rights to the relation. One of the tasks of the primitives is to ensure that a user attempting an access does have the appropriate authorization as well as being at a suitable protection level.

3.3 The DMS Kernel

So far, little has been said about the security related primitives of the DMS kernel. These are the functions which access the data base and are responsible for maintaining the protection policy. They are specified in terms of the effects which they produce on the system state. To maintain security, kernel functions must execute in isolation from the system users and, indeed, users will not always be aware of their outcome. A case in point is that of appending information to an object at a protection level dominating the user's current level. This is a legitimate and useful opera-

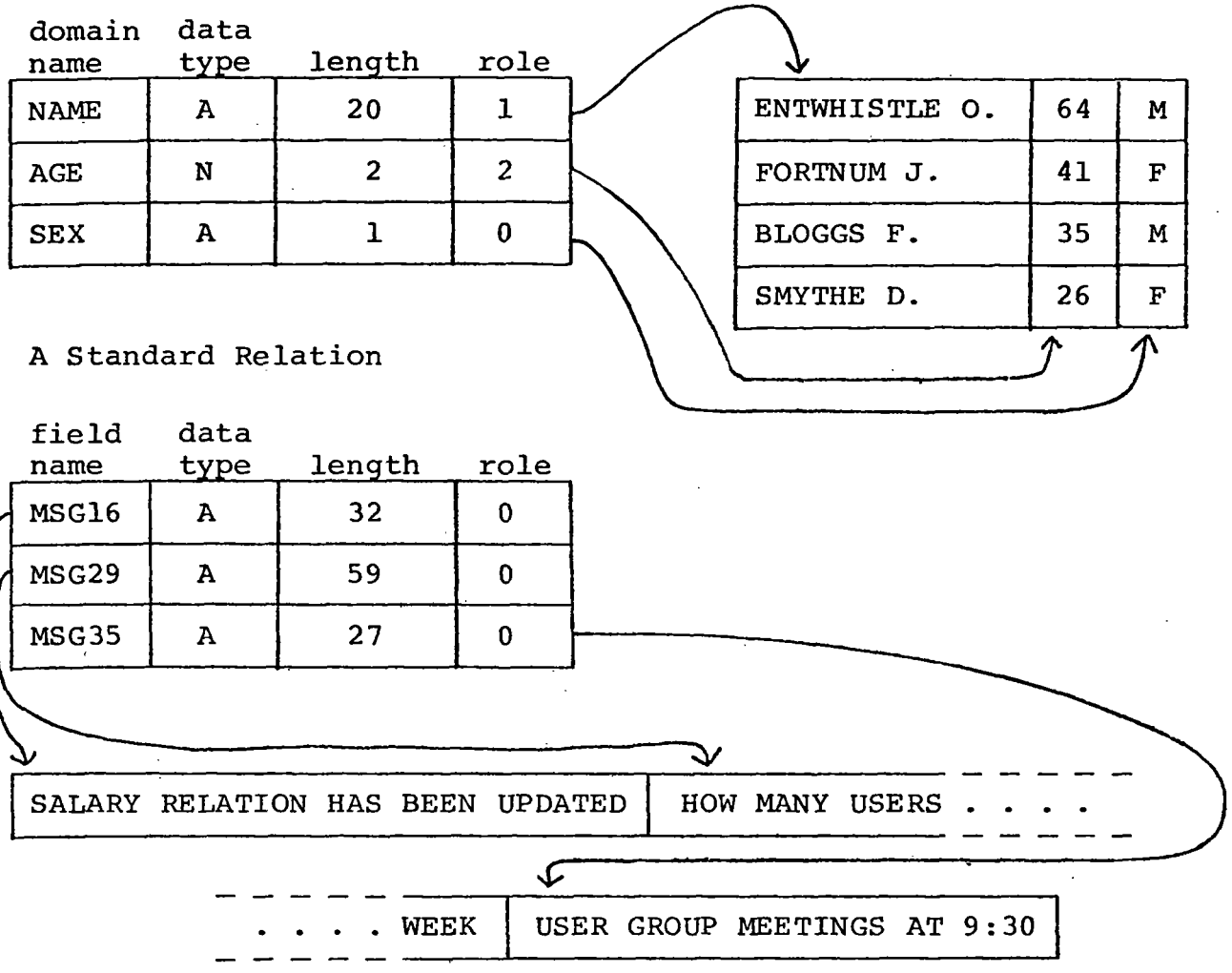


Figure 1

Logical Representations of The Permissible Relational Types

tion, but the user may not be informed of its success, since to do so could cause an information flow from a high to a low level.

A decision was made to consider an entity space associated with the kernel, where intermediate results could be stored and manipulation occur without the user's knowledge. The acknowledgement of this kernel working area, denoted by K, was a matter of choice, adopted to facilitate the definition of the DMS kernel primitives. A K area will exist for each user, but it will be more closely associated with the data base, D, than with the W area. Figure 2 is a logical view of the DMS as it would appear to a user.

Each K area contains an accumulator and some temporary storage space. Certain kernel primitives transfer entities from the data base into the accumulator, other primitives manipulate the data there and yet another is responsible for transference of information into W. All of these primitives scrupulously maintain the principles of protection as established in the model. All data copied from D into K is, at least, maintained at its original protection level; however, as was mentioned in 3.1, any information transfered into W assumes the level of W, namely the user's current level.

A number of additional entities reside in K, among them the user's identification, current level and the level of the information presently in the accumulator. This latter level may never be less than the user's current level.

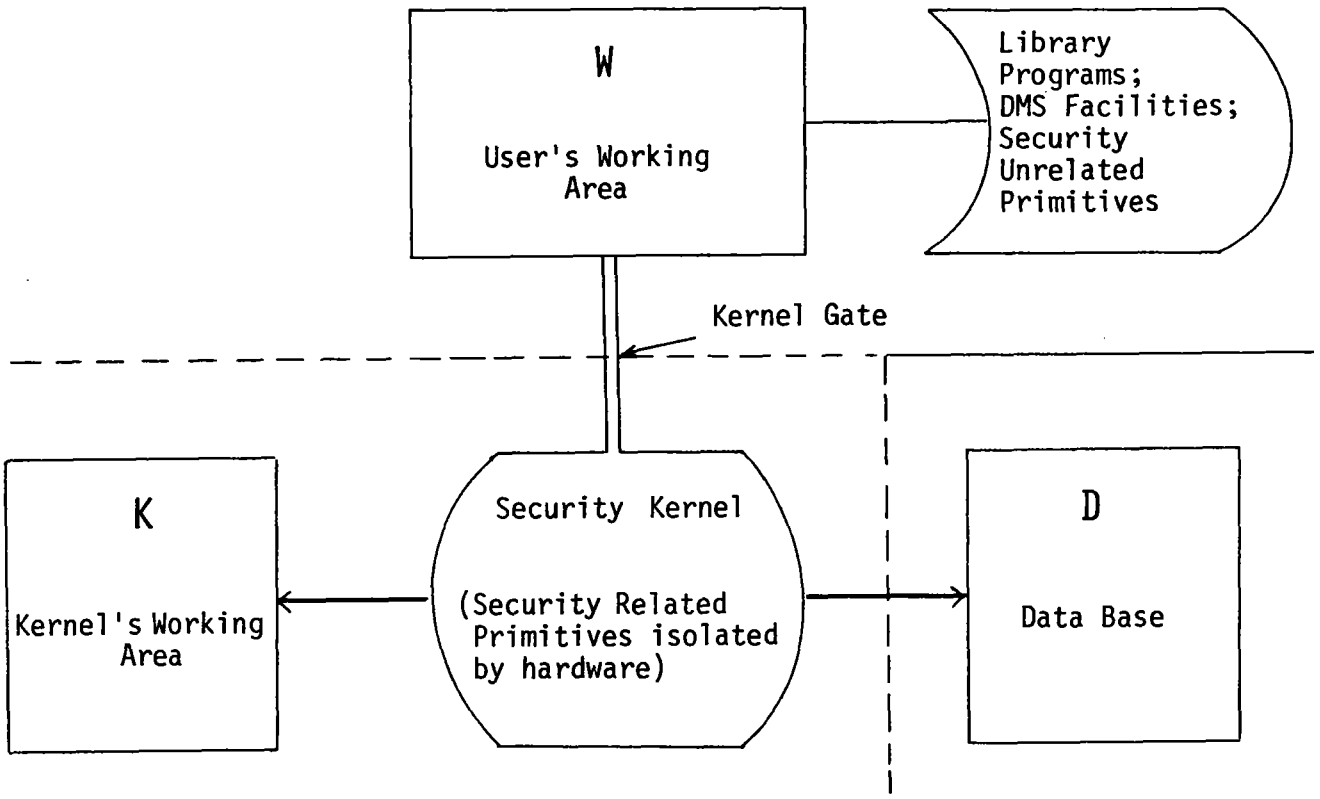
Data base resource management is organized on a per user per session basis and is determined by a quota count contained in K. System deadlock is circumvented by maintaining a table of currently reserved objects for each user, within the K area.

IV. Specifying The Design

The specification language is structured to serve the following purposes: to specify concisely the design of the DMS security kernel primitives, to describe the external (interface) characteristics of the security kernel, to facilitate validation of the security of primitive functions, and to communicate efficiently the concepts involved. The language is derived from Parnas [3] techniques, and includes Pascal-like [4] data type mechanisms. Symbols for mathematical operations are arbitrarily taken from APL [5]. The parnas-like constructs of the language are: variables, logical conditions, V-functions, assignment statements, exceptions, and O-functions. Types are established for O-function parameters, V-function paramenters and V-function values.

Figure 3 demonstrates how the primitive for reserving an object (RES) is specified. There is a similar 'O' function specification for each primitive function [6], including those unrelated to security, which operate only in the W area. The purpose of an 'O' function is to operate on specific variables to produce a change of state. Each "O' function is made up of two sections: one determining conditions for failure; and one specifying the effect of a successful execution.

The exception conditions are logical expressions, represented using arithmetic, set theoretic, and relational notation. Each expression is checked in a predetermined order and a true result causes the 'O' function to fail at that instant. A return code, such as 'NO', will be returned to the user if this does not constitute write-down.



LOGICAL OVERVIEW OF SYSTEM

Figure 2

(A) 0-function RES(o,n,t)

```
  *Reserve a data base object at the*
  *current level if it's available.*
parameter types
  user o, name n, type t
abbreviation
  id = o,n,t,K_CUR_LEVEL                  *Identifier if object to be reserved*
exception
  NO:  (id,RSRV) ∉ K_OPEN                 *Object has not been opened*
  RS:  ⊢D_R(id)⊣ ≠ ∅                      *Someone has it reserved*
effect
  [1]  D_R(id) ← K_CUR_ID                 *Set object as reserved
  [2]  K_RESERVE ← ⊢K_RESERVE⊣∪(id)       *Append identifier to reserve table*
  [3]  W_CODE ← DN
```

(B) Access Table

Variables Observed	Variables Modified	Variables Observed and Modified
K_CUR_ID K_OPEN K_CUR_LEVEL	W_CODE	D_R(id) K_RESERVE

Note that "⊢⊣" indicates the value of the variable before 0-function invocation.

Figure 3

If all exception conditions prove false then the effect specified will occur. Every effect section consists of a number of assignments which will occur in an arbitrary order. These assignments give new values to certain variables, sometimes modifying the old value, as in the example where the identifier of the newly reserved object is appended to the reserve table in the user's K area (K_RESERVE).

The construct ⊢VAR⊣ indicates the required value is that of VAR prior to invocation of the 'O' function. The prefix 'K_' indicates that the variable resides in the K area and similar prefixes are used for identifying variables in W and D.

V. Conclusions

Our efforts in specifying and validating primitive entities and functions for a secure DMS suggest that secure data management utilizing the reference monitor technique is possible. System portions with different security and operational characteristics (i.e. W,K,D) are implementable using multi-state (rings) computers. The entities and functions found in the security kernel are implementable as a separate state or a back-end processor to the data base. The validation of the kernel functions guarantees the maintenance of data security.

The relational approach to data management is consistent with and convenient for the incorporation of security access control mechanisms. This is because a single data structure (a relation) is the object of level assignment and access control. Then sets of data and relationship among the data sets may be controlled by the same security kernel mechanisms. There are no hidden links or inverted lists to complicate the security considerations. Of course, the security of the relational DMS depends upon the correctness of its implementation using such mechanisms.

The use of working areas simplifies the security considerations since complex data management can be performed in isolation, leaving only data movement and data base housekeeping in need of certification. Additionally, timing difficulties involving concurrent data base accesses would tend to be reduced because of minimal direct data base activity.

The availability of a useful, viable secure DMS will depend on progress towards the construction of secure operating systems and (hardware) security mechanisms for computing machines. An important requirement of a secure OS for our secure DMS is maintenance of isolated user processes and working area. By isolated is meant that there is no information flow from one user to another user which is recognizable by means of the user interface (other than the shared data base, of course).

References

1. M. J. Grohn, A Model of a Protected Data Management System, ESD-TR-289, I. P. Sharp Associates, Ottawa, Canada, June 1976.

2. D. E. Bell and L. J. LaPadula, Secure Computer System: Unified Exposition and Multics Interpretation, ESD-TR-75-306, The MITRE Corporation, Bedford, Massachusetts, July 1976.

3. D. L. Parnas, A Technique for Software Module Specification with Examples, Communications of the ACM, Volume 15, Number 5, May 1972, pp. 330-336.

4. K. Janson, N. Wirth, Pascal User Manual and Report, 2nd. edition, Springer-Verlag, New York, 1976.

5. An Introduction to Sharp APL, I. P. Sharp Associates Limited, Toronto, Canada, 1975.

6. G. Kirkby, M. Grohn, On Specifying the Functional Design for a Protected DMS Tool, I. P. Sharp Associates Limited, Ottawa, Canada, March 1977.

MEETINGS OF INTEREST

◆ ACM - SIGMOD International Conference on Management of Data
August 3-5, 1977
University of Toronto
Toronto, Canada

General Chairman
W. Frank King
IBM Research Lab.
San Jose, CA

Local Arrangements Chairman
Dennis Tsichritzis
University of Toronto
Toronto, Canada

Advanced Registration closes July 15, 1977. Please mail with check payable to SIGMOD CONFERENCE 1977 to the Conference Treasurer:

Sakti P. Ghosh
Computer Science Dept. K55/282
IBM Research Lab
5600 Cottle Road
San Jose, CA 95193, USA

Member of ACM and SIGMOD	$55.00 (USA)
Member of ACM only	$60.00 (USA)
Member of SIGMOD only	$60.00 (USA)
Nonmember	$80.00 (USA)
Student	$20.00 (USA)

◆ VERY LARGE DATA BASE CONFERENCE

The Third International Conf. on Very Large Data Bases (VLDB) will be held in Tokyo during Japan's Information Week (October 1-7, 1977). Previous VLDB conferences were held in Boston, MA, U.S.A., and Brussels, Belgium, and had attracted many users, designers, and researchers of very large databases from all over the world. It is expected that the VLDB conference this year will have even more participants than the previous ones since it will be held in conjunction with Japan's Information Week.

The Information Week in Japan began in 1972. In 1976, there were 78 activities held in 30 cities attracting more than 100,000 participants. The activities included conferences and exhibition of computers. Last year, there were conferences on new information technology, computers and society, investment for information processing, and computer usage in industry. Exhibition included terminals, mini-computers, office computers, and communication equipments.

The VLDB conference is expected to be one of the major activities of Information Week. It is intended to promote an understanding of very large databases, both in terms of complexity of structure as well as physical size. There will be tutorial sessions as well as technical sessions. Tutorial sessions will be given by leading experts in the field and are intended to provide an introduction to the advanced topics to be discussed in the technical sessions. Conference topics include: database design, data base machine architecture, database machines, database system analysis and evaluation, large-scale database applications, database systems, access control, concurrent access, database structures, integrity and recovery user interface, and database languages. A limited number of travel grants for the participants of the conference is available. Applications should be directed to one of the following persons:

Professor Tosiqasu Kunii
General Conference Chairman
Department of Information Science
University of Tokyo
Tokyo, JAPAN

Professor Stuart Madnick
Conference Chairman
Center for Information Systems Research
Massachusetts Institute of Technology
Room E53-333
Cambridge, MA

Dr. Hermann Schmutz
European Coordinator
IBM - Heidelberg Scientific Center
Heidelberg, GERMANY

Reduced rates for pre-registration are available until September 15, 1977. For further information, please contact either of the registration chairmen:

Mr. Yutaka Karasawa
Manager of Academic and
Scientific Programs
IBM Japan Ltd., Dept. 974
2-21, 3-chome, Roppongi, Minato-ku
Tokyo, 106, JAPAN

Mr. James Gabbert
Room E53-330
M.I.T. Sloan School
50 Memorial Drive
Cambridge, MA 02139

The VLDB Conference is sponsored by the following organizations: ACM, SIGBDP, SIGIR, and SIGMOD, IEEE Computer Society TC/DBE, Information Processing Society of Japan (IPSJ), International Federation for Information Processing (IFIP) and Society for Management Information Systems (SMIS).