

DTIC FILE COPY

5

**RADC-TR-90-268
Final Technical Report
November 1990**

AD-A231 878



RESEARCH DIRECTIONS IN DATABASE SECURITY, II

SRI International

Teresa F. Lunt

**DTIC
ELECTE
FEB 14 1991
S B D**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

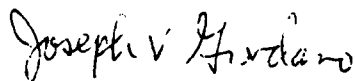
**Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700**

91 2 13 098

This report has been reviewed by the RADC Public Affairs Division (PA) and is releasable to the National Technical Information Services (NTIS) At NTIS it will be releasable to the general public, including foreign nations.

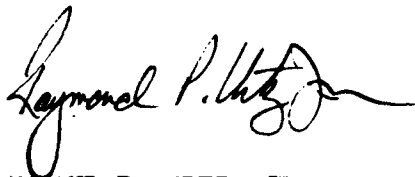
RADC-TR-90-268 has been reviewed and is approved for publication.

APPROVED:



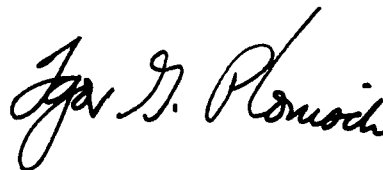
JOSEPH V. GIORDANO
Project Engineer

APPROVED:



RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER:



IGOR G. PLONISCH
Directorate of Plans & Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTC) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

GTRI - Prime
Cont. Git

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE November 1990		3. REPORT TYPE AND DATES COVERED Final Jul 89 - Nov 89	
4. TITLE AND SUBTITLE RESEARCH DIRECTIONS IN DATABASE SECURITY, II			5. FUNDING NUMBERS C - F30602-88-D-0025 PE - 35167G PR - 1068 TA - 01 WU - P3	
6. AUTHOR(S) Teresa A. Lunt			8. PERFORMING ORGANIZATION REPORT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SRI International Menlo Park CA 94025			10. SPONSORING/MONITORING AGENCY REPORT NUMBER RADC-TR-90-268	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Air Development Center (COTD) Griffiss AFB NY 13441-5700			11. SUPPLEMENTARY NOTES RADC Project Engineer: Joseph V. Giordano/COTD/(315)330-2925	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report describes the final results of a workshop on database security held 15-18 May 1989 in Bethlehem New Hampshire. The workshop featured several short presentations on technology areas including trusted applications, security checking with Prolog extensions, inference/aggregation, auditing, Discretionary Access Control and operating system support. Vendor activities and new product developments in multilevel secure database management systems technology were also presented. Finally, a major portion of the workshop was devoted to discussion and presentation of database design alternatives for multilevel systems with emphasis on graphical presentation, inference/aggregation, context and content-dependent classification, and modeling of classification constraints.				
14. SUBJECT TERMS Multilevel Security, Trusted Systems, Database Management Systems			15. NUMBER OF PAGES 200	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Contents

Workshop Summary

Teresa F. Lunt, SRI International

SCTC Technical Note: Organizing Secure Applications "By Name"

Paul Stachour, Honeywell Inc. Secure Computing Technology Center

MAC, DAC and the Need-to-Know

Gary W. Smith, George Mason University

Security Checking with Prolog Extensions

M. B. Thuraisingham, The MITRE Corporation

MLS Database Design, Homework Problem #2

Gary W. Smith, George Mason University

Report on the Homework Problem

Gary W. Smith, George Mason University

Database Design with Row Level MAC and Table Level DAC

Thomas H. Hinke, TRW

MLS Implementation Using the Sybase Secure SQL Server

Melody F. O'Brien and Edward D. Sturms, TRW, Inc.

SCTC Technical Note: Secure Database Homework Problem #2, LOCK Data View

Paul Stachour and Dan Thomsen

Honeywell Inc. Secure Computing Technology Center

Auditing in Secure Database Management Systems

Sushil Jajodia, Shashi K. Gadia, Gautam Bhargava, and Edgar H. Sibley

George Mason University

Secure DBMS Auditor:

Audit in Trusted Database Management System Environments
Marvin Schaefer, Trusted Information Systems

The ROLES Facility

Bill Maimone, Oracle Corporation

DAC Mechanisms in Trusted Database Management Systems

Ronda R. Henning, Harris Corporation

Operating System Support of Multilevel Applications

Catherine Meadows and Judith Froscher, Naval Research Laboratory

**An Interim Report on the Development of Secure Database Pro-
totypes at the National Computer Security Center**

John R. Campbell, National Computer Security Center

Some Remarks on Inference Controllers

T. Y. Lin, California State University, Northridge

Commutative Security Algebra and Aggregation

T. Y. Lin, California State University, Northridge

List of Attendees



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification _____	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Workshop Summary

Teresa F. Lunt
Computer Science Laboratory
SRI International
Menlo Park, CA 94025

Introduction

The second RADC Invitational Database Security Workshop was held May 15-18 in Bethlehem, New Hampshire. Thanks to the unflagging efforts of Joe Giordano, the workshop was funded by the U. S. Air Force, Rome Air Development Center (RADC) ¹. The workshop focused on multilevel security issues for Class B3 or A1 database systems. The workshop was organized by Teresa Lunt of SRI. Twenty people attended from SRI, NRL, TIS, Kanne Associates, George Mason University, NCSC, RADC, Oracle, Tandem, TRW, DEC, Harris, MITRE, and Secure Computing Technology Corporation.

The workshop featured several short presentations on research in progress, but the bulk of the workshop was devoted to discussion. Gary Smith of George Mason University devised an extensive homework problem, which was mailed to attendees about a month before the workshop. Much of the workshop's discussion sessions were devoted to presentation and discussion of the various approaches to the homework problem.

Process Privilege

Paul Stachour of Secure Computing Technology Corporation gave a talk called "On Behalf Of," in which he discussed the pros and cons of database processes that operate "on behalf of" a user versus processes that operate "in the name of" a user. By "on behalf of" a user, Stachour means that the user's actions are carried out by another agent at the user's request, and that agent operates with different, usually greater, privileges than the user's. By "in the name of" a user, Stachour means processes that operate under the user's name with the user's privileges. Most database systems operate "on

¹Subcontract E-21-T27-S1 of prime contract F30602-88-D-0025 with Georgia Institute of Technology

behalf of" users, with a single process serving many users simultaneously, and operate with privileges to the entire database, whereas individual users are authorized for only a subset of the database. Stachour advocates using, to the extent possible, processes that operate "in the name of" the user, for least privilege and accountability. One workshop participant noted that we need a lightweight process mechanism in operating systems so that we can perform database processing "in the name of" users.

Need to Know

In Gary Smith's talk called "MAC, DAC, and the Need to Know," he pointed out some "need-to-know" security policy requirements that cannot be met by conventional mandatory or discretionary mechanisms. According to the *Trusted Computer System Evaluation Criteria* [1], need-to-know requirements are to be addressed by discretionary mechanisms. However, Smith pointed out that such requirements have a mandatory flavor to them, in that the authorizations should not be passed from user to user at the user's discretion. On the other hand, these need-to-know requirements also have a discretionary flavor to them, in that the authorization types do not form a lattice, and information flow security may not be required. For example, the policy "managers are authorized to see salary data only for those below them in the chain of command" is a need-to-know policy that cannot readily be mapped onto levels and compartments, but is also nondiscretionary in that users cannot pass on their authorizations to other users. Thus, Smith advocates special "need-to-know" mechanisms that are different from mandatory and discretionary mechanisms. These need-to-know authorizations should be administratively controlled rather than user-grantable. Categories are neither appropriate nor sufficient to control need-to-know. As another example, Davison pointed out in his talk that release markings such as NOFORN are related in an inverse manner to a lattice (e.g., "US Only" and "US/UK" combine to "US Only"). Thus, a labeling mechanism will not be adequate to handle these markings.

Modeling Issues

Bhavani Thuraisingham of MITRE gave a talk called "Security Checking with Prolog Extensions," in which she used fuzzy logic to develop a fuzzy multi-

level relational data model. In her model, each multilevel tuple is marked with a *chi value*, that indicates the likelihood that the fact represented by the tuple is true. She added fuzzy algebra to the relational operators to compute new *chi* values for derived tuples. She uses the *chi* values to control partial inferences.

Lucien Russell of George Mason University gave a short talk in which he presented a new data model for multilevel database systems. He feels that the relational model is not powerful enough to adequately represent the necessary semantics. In particular, he feels that the relational model cannot distinguish among the various semantics possible for update, such as changing a value, changing how a value is represented, changing a version of the value, changing the classification of a value, and correcting errors. He suggests a new model in which all relations are binary, containing only the association of a data attribute with a key attribute, and in which "surrogates" (system-generated indices) are used to identify the individual associations.

Auditing

In his talk "Support for Auditing in a Secure Database Environment," Sushil Jajodia of George Mason University presented a temporal model that captures timestamps for when database operations occurred and also records the periods of validity for the data values. Marv Schaefer of Trusted Information Systems gave a talk called "Auditing in a Multilevel Database System" in which he described a project whose aim is to develop auditing requirements for multilevel database systems. One requirement they are considering is the ability to determine where particular data has propagated when all copies need to be upgraded.

Operating System Support

Catherine Meadows of the Naval Research Laboratory gave a talk called "Operating System Support of Multilevel Applications," in which she pointed out that operating system security policies have not been particularly relevant to multilevel database systems. We have to work too hard; either we struggle to stay within the limits of the operating system's policy or we struggle to reimplement operating system functions in a new context. She asked what approaches to designing operating systems would make life easier for the

designers of multilevel database systems. She suggested that perhaps an operating system should provide a set of security-supporting functions rather than enforce a particular policy. Then the database system can be free to enforce its own policy using the security primitives provided by the operating system without having to duplicate the functionality of the operating system. Some additional suggestions put forth by the workshop participants were support for an arbitrary number of levels and categories; the ability to replace the operating system's lattice structure with another label-based security structure, possibly not a lattice; an abstract data type for labels; strong typing; and finer granularity of function. Rae Burns of Kanne Associates also discussed the support she would like to see from operating systems. She advocated operating systems that merely provide domain isolation and strong data typing, rather than providing a file system. Her feeling was that a database system would not make use of the operating system's file system, but would manage its own set of objects in a domain separate from the file system.

The workshop participants also discussed how trusted (in the Bell-LaPadula sense [2]) subjects can be used in the database system without invalidating the evaluation of the underlying operating system. The problem is that the use of a trusted subject in combination with the operating system's TCB can introduce new information flows beyond those that can be discovered by performing a flow analysis on the trusted subject alone; such flows can be discovered only by performing a flow analysis on the combination of the trusted subject and operating system TCB, a task that may not be possible for a database system vendor to do. The group felt that a set of conditions could be developed that a trusted subject would be required to meet; these conditions would ensure that no additional flows were introduced by the interaction with the operating system TCB. Such constraints would be developed by the vendors of the trusted operating systems. It would then be the responsibility of the database system vendor to show that the trusted database system does not violate the operating system's trusted subject constraints.

Vendor Developments

John Campbell of the National Computer Security Center presented the results and status of the vendor prototype developments being supported by

NCSC. These prototypes are being developed by Teradata and by Oracle Corporation. They are intended to be fully functional, commercial quality database systems. Teradata's prototypes are database machines, whereas Oracle's are host-based database systems. Teradata is using the "monolithic" architecture described in the draft *Trusted Database Interpretation*, whereas Oracle is using the TCB subsets approach [3]. The currently-funded prototypes are to be C2 and B1, but Campbell said that NCSC has plans to fund prototype developments for B3 and A1 database systems that would also include significant new requirements for data integrity, inference controls, denial of service controls, and data distribution. Both vendors have already delivered C2 prototypes, and both plan to have the C2 enhancements incorporated into their standard products in the summer or fall of this year. Problems common to both systems were that some system functions, such as bulk data import and system recovery, bypassed the security controls; more attention will be given to this area. Both vendors are working with NCSC to develop SQL modifications and extensions to handle selective auditing, groups, roles, and referential integrity; these will be presented to the ANSI SQL standardization committee for consideration for incorporation into ANSI-standard SQL. Campbell noted that Teradata's C2 enhancements did not add any performance penalty (they have not measured Oracle's performance yet), with the exception that if all auditing options are turned on there can be up to a fifty percent slowdown. Campbell plans to investigate storing only summary audit information instead. For the B1 prototypes, separate audit logs will be required for each level and category.

Other Issues

In his talk "Compartmented Mode DBMS and the Need for Dual Labels," Jay Davison of DEC discussed work at MITRE to develop a database system to run on the Compartmented Mode Workstation. Ronda Henning of Harris Corporation also discussed work underway at Harris to develop a database system for the Harris Compartmented Mode Workstation.

One evening was devoted to discussion of recent changes to the TDI; these are reported elsewhere [4].

Flexible Access Controls

Bill Maimone of Oracle Corporation gave a presentation of Oracle's new roles facility. The approach is apparently motivated by the fact that different commercial customers want to enforce very different security policies. Oracle's new approach is to have a flexible mechanism that can be used to enforce a variety of different commercial policies. They wanted to make it easy to add new users and new applications (with new tables) and correctly set the table privileges for them. Their approach groups privileges, rather than users. A collection of privileges is a role. Roles are first-class objects, and have owners. Thus, authorized users can grant and revoke privileges to and from roles. Roles are assigned to users as required for the applications they use, and roles can be further grouped into meta-roles. The roles are designed by the application designer. The simplicity in assigning roles to users is that there is no need for the security officer (or whoever assigns the roles) to know which tables to grant privileges for.

Bob Baldwin of Tandem Computers described what he calls "policy-based access controls." These controls consist of rules that govern which subjects can perform which actions upon which objects. These rules are stored in a rule table. Baldwin also showed how such rules can be used to express the Bell and LaPadula model. The group did not react adversely to his suggestion that Tandem might build a B1 system in this way.

The Homework Problem

Those presenting their approaches to the homework problem included Bill Maimone of Oracle Corporation, Bob Baldwin of Tandem Computers, Ed Sturms of TRW, Tom Hinke of TRW, and Paul Stachour of Secure Computing Technology Corporation.

In the discussion of the homework problem, Gary Smith introduced a graphical notation he developed to represent multilevel data and their classifications along with an indication of what causes them to be classified. Data is not generally classified in isolation; for example, the value 17.3 is not classified in isolation, but its association with its attribute name YIELD, the key value "anti-matter" and the key attribute name WEAPON may be classified. In addition, the *existence* of the classified data may or may not be classified. This may have important ramifications on how to design an application so

as to adequately protect its classified data, and also on the design of multi-level database systems to contain that data. As Smith points out, most of the systems we are designing do not distinguish between when an attribute value is classified and when the association between the attribute value and its key value is classified. For example, if an employee's skills are SECRET whereas the employee name is unclassified, should this mean the skills values themselves are SECRET, or is their association with a particular employee SECRET?

One finding of the workshop participants as a result of doing the homework problem is the difficulty of enforcing value-dependent security constraints. There seemed to be a general consensus that using value-dependent constraints for mandatory security is unsound. For example, if all salaries above \$75K are SECRET but those below \$75K are unclassified, difficulties arise if all employees are to receive a ten percent raise; either some previously visible employees disappear from the unclassified salary table, thereby introducing a covert channel, or the classification rule must be violated. There are different problems with enforcing value-dependent discretionary rules. For example, the rule that managers can see the salaries of only their employees is easily implemented using a view, but systems such as Sybase do not provide any assurance for views. The absence of a view mechanism to enforce discretionary policies by all except the Oracle and SeaView systems also made the following homework problem requirement difficult: project data may be updated only by that project's secretary. Because most of the systems under development use relation-level (versus tuple-level or element-level) discretionary authorizations, most systems can implement this requirement only by creating a different relation for each project, each of which contains only a single tuple! Although this may seem to point to the need for tuple-level access control lists, such a solution would be extremely awkward and space-consuming. The natural solution is to use a view. The workshop participants agreed that this policy requirement of the homework problem was an entirely realistic one, and they expressed concern that without balanced assurance such policies cannot be enforced in systems targeted for the higher evaluation classes.

Conclusions

The workshop participants discussed operating system support for secure database systems; database system process privilege; mandatory, discretionary, and need-to-know requirements; modeling issues, auditing, and vendor developments. Perhaps the most valuable part of the workshop was the discussion of the homework problem. The participants' experience of determining how their secure database systems could support the sample application described in the homework problem brought several key issues to light. The participants discovered that it is important to know what makes a particular data item classified in order to know how to protect that data in a secure database system. They also discovered that for most of the systems under development, the discretionary access controls did not have nearly the flexibility that the application required. Discretionary access controls on views are needed, pointing to the need for balanced assurance.

References

- [1] National Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria*. Technical Report DOD 5200.28-STD, Department of Defense, December 1985.
- [2] D. E. Bell and L. J. LaPadula. *Secure Computer Systems: Unified Exposition and Multics Interpretation*. Technical Report ESD-TR-75-306, The MITRE Corporation, Bedford, Massachusetts, March 1976.
- [3] A first glimpse at the TDI. *Data Security Letter*, (4), P.O. Box 1593, Palo Alto, California, August 1988.
- [4] Balanced assurance dropped from TDI. *Data Security Letter*, (10), P.O. Box 1593, Palo Alto, California, June 1989.

SCTC Technical Note
Organizing Secure Applications "by Name"
2nd RADC Secure Database Conference

LOCK Data Views

Prepared by:

Honeywell Inc.
Secure Computing Technology Center
St. Anthony, Minnesota, 55418

August 16, 1989

Prepared for:

Rome Air Development Center
Workshop on Database Security
Bethlehem, New Hampshire
May 16..18, 1989

PREFACE

This report entitled 'Organizing Secure Applications "by Name"' describes one way to organize a secure application such as a Database Management System. It describes some design tradeoffs considered in the definition of the design for LOCK Data Views (LDV). It gives the reasoning behind the LDV style.

The work was supported by Honeywell from IR&D funds and by the personal time of the author. It is not a part of the Secure Distributed Data Views (SDDV) contract (contract no: F30602-86-C-0003), nor of any other federal government contract. It is related to work done on the SDDV and LOCK contracts by the Secure Computing Technology Center (SCTC), Honeywell Inc.

This report describes activity performed by the LDV team studying some aspects of discretionary security during the period 1 February 1988 through 15 March 1989. The SCTC person on this activity was Paul Stachour.

Controlling Secure Applications using In-the-Name-Of Organization Paul Stachour, Honeywell SCTC

1. Introduction

In designing and building any secure application, we have a need to control the flow of information, which is usually done by controlling the flow of data. The objective of this control is to allow only authorized operations: reading, writing, and use of metadata for control. At the same time, one wants to prevent unauthorized reading, writing, and use of metadata.

In a secure application, one normally considers three kinds of control:

- o Mandatory Access Control (MAC) [Levels]
- o Discretionary Access Control (DAC) [Users]
- o Well-Formed Operations (MGR) [Program Modules as Managers]

The LOCK Data Views (LDV) approach is that all three kinds are useful if they are properly integrated. The security community generally considers and studies MAC very diligently. The two other kinds of control are often ignored. This paper discusses two common varieties of controlling DAC and points out why LDV chooses the less common "In-The-Name-Of" variety. The effect of By-Name on MGR and MAC policies is also discussed for completeness; however, the focus of this paper is on DAC.

2. How to Control DAC

There are two varieties of DAC control in general use. They are referred to in this paper as "On-Behalf-Of" (also known as Behalf, or OBO) and "In-The-Name-Of" (also known as By-Name, Name, or ITNO).

When writing a secure application, one important question to have answered is:

Who really did it to what; and where, when and how did it get done.

The key words to be thinking about in getting an answer to the question are not only "authorized", but "audit". The question is, under what organization is it easiest to ensure that the DAC controls are properly applied and that audit data is not compromised, either during preparation or later.

2.1. What Control Organization?

We reject immediately the idea of no control organization. No control organization would mean no control of writing, thus leading to uncontrolled change. Neither would there be any control of reading, thus leading to uncontrolled disclosure.

2.1.1. On-Behalf-Of Organization

The key concept behind the On-Behalf-Of organization is that the work is done by "something other than a user". The work is done at a user's request, often by sending a message to some other process. This second process runs as the agent of the user. That is, as the agent, it acts On-Behalf-Of that user, though not with the user's authority. It acts with the authority of some pseudo-user. The initial process hopes that the second process is indeed acting on its side, and not performing an illicit activity for which the original process will be held accountable. That second process must explicitly decide which actions can be performed for this request of this user and which actions must be disallowed.

2.1.2. In-The-Name-Of Organization

By contrast, when using the In-The-Name-Of organization, the work is done by the user "directly". All work is done with the authority of the user. Any action attempted is allowed wherever the user is allowed to perform the operation. All other operations are automatically disallowed by the underlying system.

2.2. Secure Applications: System Breakers?

We digress here to point out that secure applications are often, by their very nature, system breakers. That is, they break the system's security policy to enforce one of their own.

Examples of such applications are: mail, meetings, and databases. For example, many mail systems work by being allowed to open mailbox files (or even all files) regardless of the discretionary or even mandatory access requirements for "normal files". The alternative, fully open mailboxes, databases, ... also seems to be a popular, though unacceptably silly, alternative.

Secure applications need organizations which are secure in all of the MAC, DAC, and MGR senses. They should not break the system policy, either in letter or in spirit. If we look at what "secure applications" really do, we see that they need to enforce the MGR property (e.g., no program can write to mailboxes except mail programs) and that the MGR enforcement is generally absent from operating systems (OSs). Applications often simulate the MGR property by placing their objects under control of a special userid. Unfortunately, this means that they then create problems with DAC enforcement, since they have pre-empted the ordinary DAC enforcement mechanism in order to enforce their MGR needs.

2.3. Examples--DAC

2.3.1. Example: In-Name-Of Organization--DAC

Let's look at a production-quality DBMS that uses In-the-Name-Of organization. This is the Multics MRDS (Multics Relational Data Store) DBMS. In MRDS, any user may create a database, and be owner of that database. Each

relation in the database has Multics Access Control Lists (ACLs) to provide discretionary control of data in that relation. Sharing between users is provided by setting ACLs, either directly or through the MRDS administration interface. All MRDS databases are stored in ring 3. This gives MGR separation by the rings, since only trusted MGRs are allowed to place objects in ring 3. This enables the DAC sharing and separation by the OS (Multics). Note that MAC can still be done by the Multics Access Isolation Mechanism (AIM), and also enforced by the OS. Both authorization and audit are easy, since each access to the database is indeed done by a Multics process with the authority of the logged-in user.

2.3.2. Example: On-Behalf-Of Organization--DAC

Let's look at a production-quality DBMS that uses On-Behalf-Of organization. This is the Unix "Uoo" DBMS. Uoo is a fictitious name used to avoid singling out any one product by name, but matches many of the DBMSs available on Unix.

Each database is owned by some "separate" user, which is a pseudo-user representing the DBMS. MGR protection is provided by users all running same set of setuid programs; those programs have the userid of the DBMS pseudo-user. Separation between users, thus, cannot be enforced by ACLs on the database, since the database must be owned by/ have ACLs appropriate to the setuid userid, and not that of the requesting user. Therefore DAC sharing and separation must be enforced by the DBMS, since the requesting-user has no access to the database. And now, since there is no longer an easy way to determine who really referenced/updated the data, audit is difficult if not impossible. There is no MAC, but that's somewhat of acceptable since Unix is not a multilevel OS and we should not strongly criticize an OS for not doing something that is not one of its objectives.

2.3.3. Example: LOCK-Data-Views Organization--DAC

Let's now turn to the LDV DBMS. The database may be owned by the database administrator (DBA) or an end-user. DAC is done by setting ACLs to control the sharing. MGR separation is done by the domain/type mechanism. Both MGR and DAC sharing and separation are enforced by LOCK TCB. Audit and authorization are easy since it is always clear which user is making each request.

2.4. Examples--MAC

2.4.1. Example: In-Name-Of Organization--MAC

The style used in MRDS is a user process at the real level of the request. Note that this means that the user is automatically MAC restricted by AIM to those levels s/he can see.¹ This means that write-down is seldom required,

¹ There is a related question of whether a user process should be allowed to operate beyond the cleared level of its user if that process is properly confined. However, the discus-

and, where it is, it can often be restricted to a small amount of control data within the DBMS. That is, nothing that is written down is directly given to the user. Access to and determination of existence of the metadata, as well of that of the data, is easily restricted since it can be stored at levels invisible to particular users. Audit in this senerio is thus easy.

2.4.2. Example: On-Behalf-Of Organization--MAC

The style that is most likely to be used in in a multilevel Uoo is that of a daemon process. Notice that the daemon process would be cleared to database-high, since it must sometimes process at database high.² The daemon often runs at database-high, since the real level of the user is not known, not available, or the DBMS organization may require such a level. This means that a write-down from a high-level process to a low-level object is required even when not necessary. The daemon has full access to, and and is able to determine the existence of, metadata that the user should not see. With all the data and metadata visible, and with little knowledge of the true user and level, audit becomes difficult.

2.4.3. Example: LOCK-Data-Views Organization--MAC

The style used in LOCK LDV is to have only user, i.e. no daemon, processes. This means that there are certain portions of the DBMS that can run "beyond the clearance level of the user". These user processes run "no-higher-than-necessary". In LDV, write-downs are confined by domains, not by trusting all code that runs under certain userids. Not only data, but also access to and existence of metadata is controlled by MAC, DAC, and domains. Because there is MAC, DAC, and domain enforcement by the TCB, it is our firm belief that less assurance effort is required than that required with a daemon organization.³ Furthermore, with that enforcement, audit is easy.

2.5. Comparison: Programming Languages / Databases / OS

Let's compare the style of DBMSs with several other things in the computing environment. The comparison is intended to show a similarity among reliability issues in programming languages, applications, and OSs.

Programming languages have generally been typed. Databases have been untyped. The typelessness of DBMS has been a boon (no change to DBMS needed when data characteristics have a small change) and bane (no enforcement of the data integrity provided by languages such as ALGOL, Pascal, and Ada). On-Behalf-Of is more like untyped data (such as in B) or weakly typed data (such as in C) while In-The-Name-Of is more like typed data (such

sion is beyond the scope of this document.

² Note that, in this case, the processing beyond the clearance level of the user exists, but is concealed.

³ To meet the letter-of-the-criteria, while ignoring its spirit, LDV could replace such user-processes with daemon processes. However, it would require more code, more assurance, be

as in Pascal and Ada).

However, more realistic items for comparison are multi-user applications, such as Database Management Systems (DBMS), Transaction Processing Executives (TPX), and Time-Sharing-Systems (TSS). Note that, in general, DBMS and TPX have been organized as On-Behalf-Of, while TSS have been organized as In-The-Name-Of.

The On-Behalf-Of organization has been both a boon (more efficient because lots of heavy-weight processes do not need to be spawned or switched) and bane (no-easy enforcement by environment and hardware, enforcement instead by explicit code) to DBMS and TPX.

3. Conclusion

We choose to run our database LDV In-The-Name-Of because it makes both authorization and audit easier. It is easier because we use good abstractions for our DAC, MAC, and MGR requirements and appropriate enforcement mechanisms for each.

We believe that there are four steps of abstraction in computer hardware and software from easy-instructions through stacks & searching, subprograms, and processes. We understand that most computer hardware and systems have a hard time handling the second abstraction step, let alone the fourth. We also understand the performance suffers greatly at each step of the abstraction if the underlying abstraction is poorly implemented. Nevertheless, we plan to use the fourth and organize or design using In-the-Name-Of to implement LDV. After all, hardware, OS, and programming languages abstractions should be designed to make it easier to write quality, correct, robust code; not to make it harder to do it right.

harder to understand, and have less performance.

MAC, DAC and the Need-to-Know

Gary W. Smith

George Mason University
School of Information Technology and Engineering
Fairfax, VA 22030

The Assertion: The mandatory need-to-know mechanisms (i.e., categories) and discretionary mechanisms (which only implement need-to-know) are not sufficient to meet all user requirements for restricting access to data. Specifically, users require need-to-know restrictions granted (i.e., controlled) by the organization for which mandatory categories cannot efficiently be implemented.

The Question: Can current (and planned) database systems provide mechanisms to implement these restrictions, or are fundamentally new mechanisms needed?

Discussion.

The US government's "need-to-know" policy is well established [DoD85, DoD87]. It says that even though a user may be trusted (e.g. have the right clearances) to access classified information, the user must also have a legitimate need (i.e., need-to-know), based upon operational requirements, to access the information. Enforcing need-to-know is another area in which operating system principles must be extended to meet the granularity of database security requirements.

The Orange Book [DoD85] provides a precise description of need-to-know through the definition of mandatory and discretionary controls, as follows.

In general, no person may have access to classified information unless: (a) that person has been determined to be trustworthy, i.e., granted a personnel security clearance --MANDATORY, and (b) access is necessary for the performance of official duties, i.e., the person is determined to have a need-to-know--DISCRETIONARY. [DoD85]

(Note that the above definition seems to equate need-to-know with discretionary and that mandatory seems to have to no connection with need-to-know.)

The Orange Book also provides the basic guidance for implementing these controls in the form of mandatory access control (MAC) and discretionary access control (DAC). *A Guide to Understanding Discretionary Access Control in Trusted Systems* [DoD87], which we call the DAC Guide, significantly amplifies the Orange Book guidance.

The Discretionary Security Control Objective in the Orange Book requires that security policies "include a consistent set of rules for controlling and limiting access based on identified individuals who have been determined to have a need-to-know for the information." Notice that the policy objective does not say who or what determines the individual's need-to-know. The Orange Book goes on to define a DAC security policy as:

The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g. self/group/public controls, access lists) shall allow users to specify and control sharing of those objects by named individuals or defined groups or both. [DoD85]

The key word in this definition is *user*. Although the Discretionary Security Control Objective does not indicate *who* is to determine need-to-know, the definition of DAC clearly states the *user* grants DAC to other users or groups. In addition, the DAC Guide goes on to add that DAC includes the following capability: "a user or process given discretionary access to information is capable of passing that information along to another subject." Thus, the current interpretation for DAC implicitly includes two concepts: the user (as opposed to the organization) grants access to the data; the user can potentially extend the access authorization to another subject. This is clearly why many researchers and practitioners consider DAC to be fundamentally flawed.

(It is interesting to note that this interpretation can be traced back to a 1976 document which says, "Discretionary security policy allows an individual to extend to another individual access to a document based upon his own discretion, constrained by non-discretionary [i.e. mandatory] security policy: that is, discretionary security policy allows an individual to extend access to a document to anyone that is allowed by non-discretionary [mandatory] security to view the document." The source--Volume IV of the Bell-LaPadula Model [BelLaP76].)

On the other hand, MAC is based upon labels (for users and data objects) that include both hierarchical classification levels and non-hierarchical categories. In fact, the non-hierarchical categories implement a formal need-to-know policy by granting an individual a category as part of his/her security clearance. Notice that MAC categories are granted to users by the organization. Further, *only* the organization can grant access based upon MAC categories and the user cannot extend access authorizations to other users.

Thus, MAC categories and DAC both implement need-to-know requirements. The first difference is that the user grants DAC (and can extend it to other users) but the organization formally grants access to MAC categories (and the user cannot extend them to other users). The second difference is that MAC is implemented using labels and DAC is implemented using a non-label based mechanism.

These two approaches may be sufficient and appropriate to control need-to-know access to operating system objects, but they are not sufficient when dealing with database objects. A need-to-know policy for database security is required that lies between MAC categories and DAC. Take the following need-to-know requirement from the financial accounting domain: account executives are authorized to access only their own accounts, i.e.,

the accounts to which they are assigned. Further, the organization considers all account data to be owned by the organization, not by a particular user, in this case the account executive. In this example, access must be controlled based upon the value of data (the account number or perhaps client) which is essentially a tuple level access control based upon value. MAC categories cannot easily implement this requirement (at least in any workable way in an organization with dozens of account executives). Yet it is an example of a real world database access control requirement. Specifically, a large organization will have more than 156 account executives, and 156 is the maximum number of MAC categories required by the Orange Book. Even if "the system" can accommodate the required number of categories, it is unclear that this approach is manageable.

Another example involves management's access to personnel records: managers are authorized to review the personnel records *only* of the employees that are directly in their chain-of-command. This seems like a simple requirement, but access is now based upon one's role which includes information not stored in the database. It is fairly easy to grant access based upon a comparison of the Department attribute of an employee with the Department attribute of the manager. But what about the next higher manager in the chain-of-command? A database will normally not store the data values necessary to make this decision. In particular, the access decision is based upon the concept of chain-of-command, but chain-of-command is not explicitly represented in the database. Large organizations have hundreds of managers (down to the branch or team supervisor). It should be obvious that maintaining labels (i.e., using MAC categories) to enforce this requirement would quickly become unmanageable.

Conclusions

To satisfy the type of need-to-know requirement illustrated by the above examples, the organization (through the system security officer and/or database security officer) needs to grant authorization to database objects---for lack of existing terminology we'll call it an *organizational DAC* (ODAC)---that implements the need-to-know policy in a way that MAC cannot. ODAC would have the desirable DAC-like property of implementing a need-to-know requirement without a formal addition to a user's clearance but would not have the undesirable properties of being user-granted and having access rights which can be propagated to other users or groups. ODAC would also have the desirable MAC-like properties of being organizationally-granted (vs. user-granted) without the restrictions inherent in formal categories.

The question remains: can existing mechanisms provide these (i.e. ODAC) capabilities? Or must database systems acquire fundamentally new mechanisms?

References

[BellLaP76] Bell, D. E. and LaPadula, L. J., *Secure Computer System: Unified Exposition and MULTICS Interpretation*, MITRE Technical Report, March, 1976.

[DoD85] DoD 5200.28.STD, *DoD Trusted Computer System Evaluation Criteria (TCSEC)*, December, 1985.

[DoD87] NCSC-TG-003, *A guide to Understanding Discretionary Access Controls in Trusted Systems*, September, 1987.

SECURITY CHECKING WITH PROLOG EXTENSIONS

M.B.Thuraisingham
The MITRE Corporation, Bedford MA, 01730

Abstract

We concentrate on two prolog extensions; fuzzy prolog and object prolog, for security checking in database systems. Fuzzy prolog will handle uncertain information and also handle inferences when users possess real-world knowledge. Object prolog handles inferences in object-oriented database systems. We also discuss the query modification technique implemented by inference engines based on fuzzy prolog and object prolog.

Introduction

Information storage and manipulation have become crucial to the day-to-day activities of many enterprises. However, protecting the data contained in databases is a difficult task especially because increasingly imaginative ways are being used to compromise database systems. Although attempts to incorporate security features into database management systems in general have been successful, these are not sufficient to satisfy the stringent requirements of military applications. To address this problem, the concept of multilevel security has been recommended and consequently database management systems are now being designed with this feature (see for example STAC89). Despite the many advances that have been made, additional problems have since surfaced. An example of these is the "inference problem" where users pose sets of queries and infer unauthorized information from the legitimate responses that they obtain (see for example THUR87, MORG87). Because of the insidious nature of these threats to database security ingenious solutions are needed to overcome them.

One of the approaches that has been taken to handle the inference problem is to augment a relational database management system with a logic-based inference engine and a knowledge base. During the query operation, the inference engine should examine the query, the security constraints which assign security levels to the data and the responses that have been previously released and determine whether the response released to the current query will result in security violation. If so the query is either modified or the request is not permitted.

A problem with first-order logic based theorem provers is that it is not straightforward to reason with probabilities and to detect partial inferences. In database applications it is possible for users to infer partial information from their knowledge of the real world. Based on this notion, fuzzy relational systems and theorem provers using fuzzy logic have been developed [MART87]. Such a theorem prover could be used to detect security violation via partial inferences (the idea of using fuzzy logic was also suggested by Dr. Dorothy Denning at the 1st RADC Database Security Invitational Workshop held in Menlo Park, CA, May 1988).

Secure relational database systems are not the only systems that are receiving attention. Recently some attempts have also been made to design secure object-oriented database systems (see for example THUR89a, THUR89b). This is because, object-oriented systems are being increasingly used for many applications such as military, CAD/CAM, and process control. For many of these applications it is important that they be secure. As in the case of relational systems, the object-oriented systems will also have to be protected against security violations via inference. Theorem provers based on object prolog, prolog extended to include object-oriented concepts, are being developed [ZANI84]. Such a theorem prover could be used to detect security violations via inference in object-oriented database systems.

This paper concentrates on the use of two prolog extensions; fuzzy prolog and object-prolog, for secure query processing. The organization of this paper is as follows. In section 2 we will briefly describe a previous approach to query modification implemented by first order logic based inference engines. In section 3 we will describe how fuzzy prolog could be used to detect security violations. Some background information on fuzzy relations and fuzzy prolog will also be provided. In section 4 we will discuss security checking in object-oriented systems augmented with inference engines. The paper is concluded in section 5. A more detailed discussion on logic programming applications in secure query processing is given in [THUR89c].

2. Query Modification in Augmented Relational Database Management Systems

Query modification technique has been used in the past to handle discretionary security and views [STON74]. This technique has been extended to include mandatory security (see for example THUR89d, KEEP89). In our design of the query processor, this technique will be used by the inference engine to modify the query depending on the security constraints, the previous responses released and real world information. When the modified query is posed, the response generated will not violate security. The security policy for query processing extends the simple property in [BELL75] to include inference. This policy is formalized below:

1. Given a security level L , $E(L)$ is the environment associated with L . That is, $E(L)$ will consist of all responses that have been released at security level L over a certain time period and the real world information at security level L .

2. Let a user U at security level L pose a query. Then the response R to the query will be released to this user if the following condition is satisfied:

For all security levels L^* where L^* dominates L ,

If $(E(L^*) \cup R) \implies X$ (for any X) Then L^* dominates Level(X).

Where $A \implies B$ means A implies B and level(X) is the security level of X .

The design of the query processor which implements the above policy is shown in Figure 1. Here a relational database is augmented with an inference engine. The inference engine has access to the knowledge base which includes security constraints, previously released responses and real world information. Conceptually one can think of the database to be part of the knowledge base. However, for efficiency reasons we separate the two. Logic is used to represent the information in the knowledge base. The user's query is posed in logic. The inference engine modifies the query. The modified query is translated into relational languages such as SQL or relational algebra. The relational query is evaluated against the relational database.

We will illustrate the query modification technique with examples. Consider a database which consists of a relation STUDENT with attributes S# (the key), Name, GPA and Dept. Let the knowledge base consist of the following rules:

1. Level(Y , Secret) \leftarrow STUDENT(X, Y, Z, D) and $Z > 3.5$
2. Level(Y , Top-Secret) \leftarrow STUDENT(X, Y, Z, D) and $D = \text{Physics}$
3. Level((Y, Z) , Secret) \leftarrow STUDENT(X, Y, Z, D)
4. Level(Y , Secret) \leftarrow STUDENT(X, Y, Z, D) and Release(Z , Unclassified)
5. Level(Z , Secret) \leftarrow STUDENT(X, Y, Z, D) and Release(Y , Unclassified)
6. Level(X , Unclassified) \leftarrow NOT(Level(X , Secret) or Level(X , Top-secret))

The first rule is a content-based constraint which classifies a name whose GPA is more than 3.5 at the secret level. Similarly the second rule is also a content-based constraint which classifies a name who is in the Physics department at the top-secret level. The third rule is a context-based constraint which classifies names and GPA values taken together at the secret level ([LUNT89] states that with "proper database design" it will not be necessary to handle context-based constraints during query processing. We will show in the next section that even with "proper database design" there are problems if users have access to real-world information - In any case, there is no standard rule which prohibits the use of context-based constraints as we have described above). The fourth and fifth rules are additional restrictions that are enforced as a result of the context-based constraint specified in rule 3. The sixth rule ensures that the default classification level of a data item is unclassified.

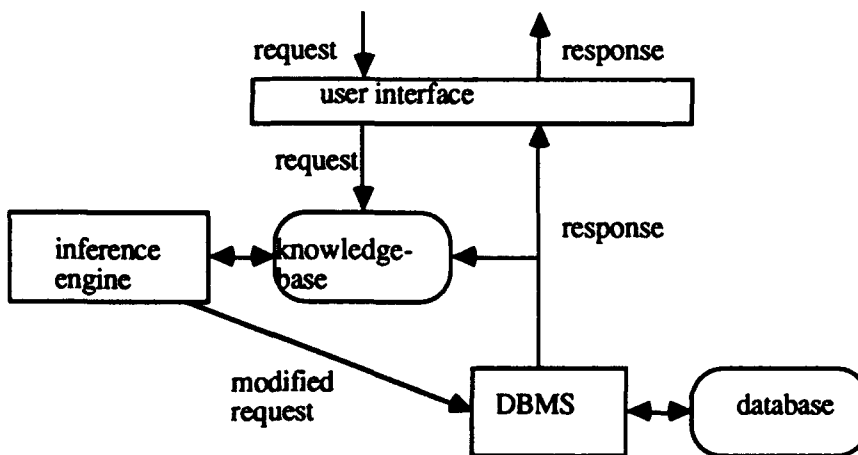


Figure 1 Query Processor

Suppose an unclassified user requests the names in EMP. This query is represented as follows:
 STUDENT(X, Y, Z, D) and Level(Y , Unclassified)

The proof procedure of the inference engine can be implemented using either a forward chaining or backward chaining mechanism. In our design the inference engine uses a backward chaining mechanism for query modification. That is, it will start with the query and perform appropriate substitutions for the various predicates occurring in the query. The following steps will be included in the query modification process:

Step 1: STUDENT(X,Y,Z,D) and NOT(Level(Y,Secret) or Level(Y,Top-secret))
 Step 2: STUDENT(X,Y,Z,D) and NOT(((Z > 3.5) or Release(Z,Unclassified)) or (D = Physics))
 Step 3: STUDENT(X,Y,Z,D) and NOT((Z > 3.5) or Release(Z,Unclassified)) and NOT(D = Physics)
 Step 4: STUDENT(X,Y,Z,D) and NOT(Z > 3.5) and NOT(Release(Z,Unclassified)) and NOT(D = Physics)
 Step 5: STUDENT(X,Y,Z,D) and Z <= 3.5 and D <> Physics
 (Note that since Release(Z,Unclassified) is not in the knowledge base, its negation is assumed - this is the Closed World Assumption).
 The modified query is STUDENT(X,Y,Z,D) and Z <= 3.5 and D <> Physics

3. Security Checking in Fuzzy Systems

3.1 Background on Fuzzy Systems

In [BALD84], concepts in relational databases have been extended to fuzzy systems. Consequently, a fuzzy relation $R(A_1, A_2, \dots, A_n)$ where A_i are attributes is defined to be a mapping $CHI_R : D_1 \times D_2 \times \dots \times D_n \rightarrow U, U = [0,1]$ where D_i is the domain of possible values of A_i . Figure 2 (following [BALD84]) illustrates a fuzzy database. This database consists of four fuzzy relations **LIKES**, **HIGHPOSITION**, **EMPLOYEE** and **WEALTHY**. For example, the fact that Jon likes Jill has a CHI value of 0.7. Therefore the CHI value of Jon does not like Jill is 0.3. If a tuple is not in a relation, then the CHI value for that tuple is 0.0. For example, the CHI value of Jon likes Jack is 0.0. In the relation **EMPLOYEE** all of the CHI values associated with the tuples are 1.0. Therefore **EMPLOYEE** can be regarded as an ordinary relation.

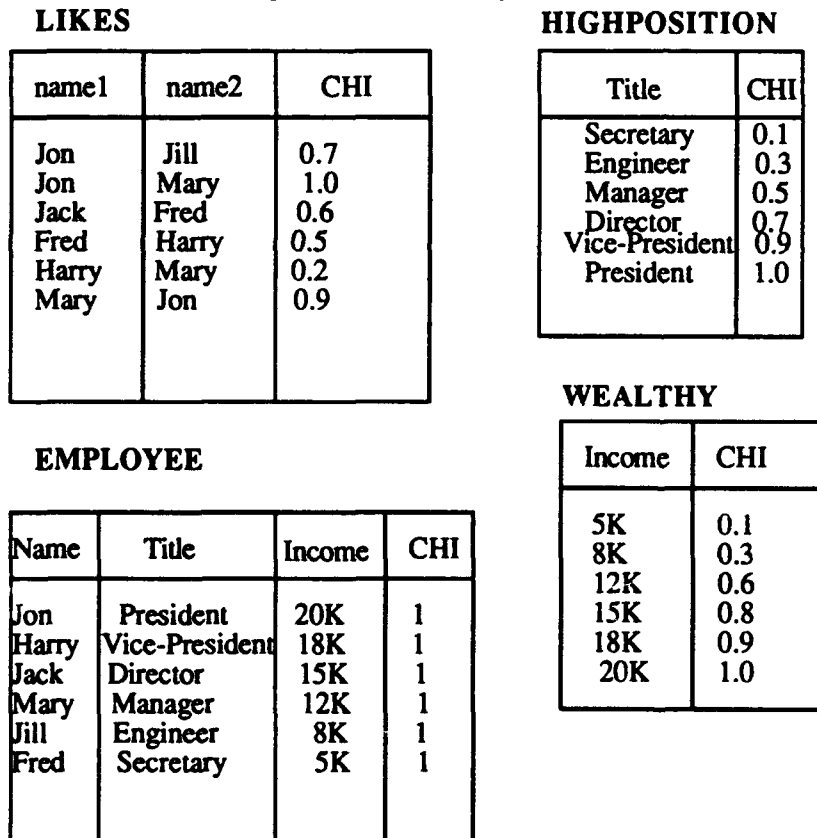


FIGURE 2 - Fuzzy Relations

The relational algebra operators **SELECT**, **PROJECT**, **UNION**, **INTERSECTION**, **DIVIDE**, **PRODUCT** and **JOIN** can be extended for fuzzy relations. Below we will only describe the **SELECT**, **PROJECT** and **JOIN** operators. In the case of the **SELECT** operation, the tuple selected from the base relation is assigned the CHI value that the tuple had in the base relation. For the **PROJECT** operation, for each tuple t in the resulting relation, the CHI values in the base relation of the tuples which have t as their sub-tuple are examined. The maximum value of these CHI values is the CHI value assigned to t . For the **JOIN** operation, the two fuzzy relations say R_1 and R_2 are joined on the specified common attributes. For each tuple t in the result, the CHI values of the tuples r_1 in R_1 and r_2 in R_2 used to produce t are examined. The minimum of the two values is the CHI value assigned to t . The discussion for the other operators is given in [BALD84].

Figure 3 describes the database of figure 2 as a fuzzy logic program. For each assertion there is a CHI value associated with it. One can also deduce new fuzzy information from existing fuzzy information. Consider the following rule:

SENIOR_EMPLOYEE(X) <-- EMPLOYEE(X,Y, -) and HIGHPOSITION(Y)

This rule states that for a person to be a senior employee, he must have a high position. The CHI value for "an employee P is a senior employee" is computed from the CHI values for "P is in EMPLOYEE" and "P has a high position". Furthermore, the computation also depends on the assignment of CHI values for the AND operator. We assume the following assignment for the AND, OR and NOT operators.

If C <-- A and B, then $CHI(C) = \text{minimum}(CHI(A), CHI(B))$

If C <-- A or B, then $CHI(C) = \text{maximum}(CHI(A), CHI(B))$

If C <-- NOT A, then $CHI(C) = 1 - CHI(A)$.

With the above definitions, it can be shown that the CHI value for "Harry is a senior employee" is 0.9.

In the above discussion, we have assumed that the rule "X is a senior employee if X has a high position" has a CHI value of 1.0. Note that as in the case of assertions, the rules also could be assigned CHI values. That is, the fact that X is a senior employee if X has a position could be assigned a CHI value between 0 and 1. Then the computation of the CHI value for "X has a high position" differs from what we have given above. To simplify the discussion, we assume that the CHI values assigned to all rules are 1.0.

LIKES(Jon, Jill) <--	(0.7)
LIKES(Jon, Mary) <--	(1.0)

HIGHPOSITION(Secretary) <--	(0.1)
HIGHPOSITION(Engineer) <--	(0.3)

EMPLOYEE(Jon, President, 20K) <--	(1.0)
STUDENT(Harry, Vice-President, 18K) <--	(1.0)

WEALTHY(5K) <--	(0.1)
WEALTHY(8K) <--	(0.3)

WEALTHY(20K) <--	(1.0)

Figure 3 - A Fuzzy Prolog Database

3.2 Handling Security Constraints

We will describe how fuzzy logic programming may be used to process queries when security constraints are present. We will discuss two examples; one which handles a content-based constraint and the other which deals with a context-based constraint.

Example 1: Content Constraint

Consider the database described in Figure 2. Let the content constraint be as follows:

Anyone in EMPLOYEE who likes a person with a high position is secret.

First it must be decided as to whether a person has to like someone with a high position with a CHI value of 1 in order to be classified as secret. Let us assume that the CHI value associated with the constraint should be greater than 0.0. That is if a person likes someone with a high position with a CHI value greater than 0.0 then he is classified at the secret level. Note that 0.0 is not a fixed value. This value could be anywhere between 0.0 and 1.0. The security constraint is specified as follows:

Anyone in EMPLOYEE who likes a person with a high position with a CHI value of greater than 0.0 is secret .

This constraint is expressed by the following rule:

$\text{Level}(X, \text{Secret}) \leftarrow \text{CHI}(\text{REL}(X)) > 0.0$

where $\text{REL}(X)$ is true if X is in EMPLOYEE and X likes someone clever. $\text{REL}(X)$ is defined by the following rule:

$\text{REL}(X) \leftarrow \text{EMPLOYEE}(X, -, -)$ and $\text{LIKES}(X, Y)$ and $\text{EMPLOYEE}(Y, Z, -)$ and $\text{HIGHPOSITION}(Z)$.

Note that the security constraint itself could be assigned a CHI value. To simplify the discussion we assume that the CHI value of all security constraints and integrity constraints are 1.0.

Let an unclassified user pose a query to retrieve all names in EMPLOYEE . There are two ways to process the query. In the first method, the query is decomposed into two sub-queries $Q1$ and $Q2$ where $Q1$ requests to retrieve all names in EMPLOYEE and $Q2$ requests to retrieve all names in EMPLOYEE which are secret. The response obtained from the query $Q2$ is subtracted from the response obtained from the query $Q1$. In the second method the query is modified to a query $Q3$ which requests to retrieve those names in EMPLOYEE which are unclassified. We will illustrate how the queries $Q1$, $Q2$ and $Q3$ are processed.

The query $Q1$ is expressed as

$\leftarrow \text{EMPLOYEE}(X, -, -)$

This query is resolved with the clauses in the knowledge base and the result generated will be as follows:

Jon 1.0
Harry 1.0
Jack 1.0
Mary 1.0
Jill 1.0
Fred 1.0

Note that this query could have been directly evaluated against the fuzzy relational database of figure 2. In this case the query will be represented as follows:

$\text{PROJECT}_{\text{name}} \text{EMPLOYEE}(\text{name}, \tau, -)$.

The query $Q2$ is expressed as follows:

$\leftarrow \text{EMPLOYEE}(X, -, -)$ and $\text{Level}(X, \text{Secret})$.

This query is resolved with the clauses in the knowledge base. The next step in the derivation is the following:

$\leftarrow \text{EMPLOYEE}(X, -, -)$ and $\text{CHI}(\text{REL}(X)) > 0.0$.

In the next step, $\text{REL}(X)$ will be substituted by

$\text{EMPLOYEE}(X, -, -)$ and $\text{LIKES}(X, Y)$ and $\text{EMPLOYEE}(Y, Z, -)$ and $\text{HIGHPOSITION}(Z)$.

At this point the query can be evaluated either against the fuzzy relational database or by continuing with the resolution process. If the query is to be evaluated against the relational database, then $\text{REL}(X)$ will be expressed by the following expression.

$\text{PROJECT}_X \text{EMPLOYEE}(X, -, -)$ and $\text{REL1}(X)$

where $\text{REL1}(X)$ is the following expression:

$\text{PROJECT}_X (\text{LIKES}(X, Y) \text{ JOIN}_Y (\text{REL2}(Y)))$

where $\text{REL2}(X)$ is the following expression:

$\text{PROJECT}_X (\text{REL3}(X, Y) \text{ JOIN}_Y \text{HIGHPOSITION}(Y))$

where $\text{REL3}(X, Y)$ is the following expression:

$\text{PROJECT}(X, Y) \text{EMPLOYEE}(X, Y, -)$.

The response for query $Q2$ is as follows:

$\text{REL}(\text{Jon}) \leftarrow (0.5)$

$\text{REL}(\text{Fred}) \leftarrow (0.5)$

$\text{REL}(\text{Harry}) \leftarrow (0.2)$

$\text{REL}(\text{Mary}) \leftarrow (0.9)$

$\text{REL}(\text{Jack}) \leftarrow (0.1)$

Therefore the response to the original query is: Jill.

The query $Q3$ will be expressed as follows:

$\leftarrow \text{EMPLOYEE}(X, -, -)$ and $\text{Level}(X, \text{Unclassified})$.

The knowledge base should have a rule which classifies any data which is not assigned the secret (or top-secret) level at the unclassified level. From this rule and the other rules in the knowledge base the response to the query will be "Jill".

Example 2: Context Constraint

In our treatment of context-based constraint, such as names and salary values taken together is classified at the secret level, we enforced some additional restrictions. For example, after the names are released at the unclassified level, the security level of the corresponding salary values were upgraded to the secret level.

Similarly after the salary values are released at the unclassified level, the security level of the corresponding names were upgraded to the secret level. In [LUNT89] an alternative approach to treat context constraints is proposed as follows: the names and salary values are classified at the unclassified level. But the association between the names and salary values are classified at the secret level. With this approach, the names as well as salary values can be released to an unclassified user. Since the user will not know which salary values belong to which names, he cannot infer secret information. However, if the user has some additional information such as "Mary earns more than Harry", then from the salary values released, the unclassified user may be able to infer some secret information. In the following discussion we will show how fuzzy reasoning may be used to detect such inferences.

Consider the database shown in Figure 4. Here the relations NAME and SALARY are not fuzzy relations. That is, the CHI value for each tuple in these relations is 1.0. The relations HASHIGHSAL and ISHIGHSAL are both fuzzy relations. These relations represent the additional knowledge that an unclassified user may have which will eventually result in this user inferring secret information. The relation HASHIGHSAL specifies the CHI value corresponding to a name having a high salary. The relation ISHIGHSAL specifies the CHI value which corresponds to a salary value being a high salary. Let us assume that if an unclassified user can infer, with a CHI value of greater than 0.6, that a name has a particular salary value, then he has obtained some unauthorized information. The knowledge base will have the following additional rules.

```

NAMEANDSAL(N,S) <-- NAME(N) and SALARY(S) and ((CHI(HASHIGHSAL(N) and
ISHIGHISAL(S)) > 0.6) or (CHI(NOT HASHIGHSAL(N) and
NOT ISHIGHISAL(S)) > 0.6))
LEVEL((N,S),Secret) <--NAMEANDSAL(N,S)
LEVEL(N,Secret) <-- NAMEANDSAL(N,S) and RELEASE(S,Unclassified)
LEVEL(S,Secret) <-- NAMEANDSAL(N,S) and RELEASE(N, Unclassified)

```

NAME		SALARY		HASHIGHSAL		ISHIGHSAL	
Name	CHI	Salary	CHI	Name	CHI	Salary	CHI
Jon	1	20K	1	Jon	0.8	20K	0.2
Mary	1	40K	1	Mary	0.4	40K	0.4
		60K	1			60K	0.8

FIGURE 4 - Fuzzy Relations for Context Constraint Example

Suppose an unclassified user poses the following queries:
 Retrieve the names in the relation NAME. Then both names in the relation NAME will be returned.
 Next, the unclassified user requests to retrieve the salary values in SALARY relation. There are only three values in the SALARY relation. They are 60K, 40K, and 20K. For each name, salary pair, the inference engine will compute the CHI value. The result of this computation will be as follows:

```

Jon 60K .8
Jon 40K .4
Jon 20K .2
Mary 60K .4
Mary 40K .6
Mary 20K .6

```

If the salary 60K is released, then the unclassified user can infer that Jon's salary is 60K with a CHI value of 0.8. Since the name Jon has been released to the unclassified user, the value 60K will be secret. That is the CHI value of the pair (Jon, 60K) belonging to the relation NAMEANDSAL is 0.8. Therefore, the salary 60K cannot be released. The other two salary values will remain unclassified. Therefore, the values 40K and 20K will be released to the unclassified user as response to the second query. This user can infer that Jon earns 40K with a CHI value of 0.4, Mary earns 40K with a CHI value of 0.6, Jon earns 20K with a CHI value of 0.2 and Mary earns 20K with a CHI value of 0.6. However, according to the security constraints, the user has not acquired any secret information.

4. Security Checking in Augmented Object-Oriented Systems

4.1 Concepts in Object-prolog

We assume that the reader is familiar with concepts in object-oriented systems. Description of an object-oriented data model can be obtained in [BANE87]. In this paper we will describe an extension to an

object-oriented database system which is based on object-prolog proposed in [ZANI84]. First we will describe object-oriented concepts and show how they may be represented in object-prolog. Then we will describe the security issues.

Let **EMPLOYEE** be a class with subclasses **SENIOR_EMPLOYEE** and **JUNIOR_EMPLOYEE**. The instance variable of **EMPLOYEE** is **Name**. The method of **EMPLOYEE** is **Status**. **Status** has no input parameters. It has one output parameter. The code for **Status** is shown below:

```
Status(X)
Begin
X := full-time
end;
```

The subclasses of **EMPLOYEE** will inherit the instance variables and methods. However, we assume that the subclass **SENIOR_EMPLOYEE** has its own **Status** method and is defined as follows:

```
Status(X)
Begin
X := part-time
end;
```

It is also assumed that the instances of **EMPLOYEE** are Fred and Jill; the instances of **SENIOR_EMPLOYEE** are Jon and Harry; the instances of **JUNIOR_EMPLOYEE** are Jack and Mary.

The database described above can be represented in object-prolog as follows:

- 1.EMPLOYEE(Fred).
- 2.EMPLOYEE(Jill).
- 3.SENIOR_EMPLOYEE(Jon).
- 4.SENIOR_EMPLOYEE(Harry).
- 5.JUNIOR_EMPLOYEE(Jack).
- 6.JUNIOR_EMPLOYEE(Mary).
- 7.SENIOR_EMPLOYEE(Name) ISA EMPLOYEE(Name)
- 8.JUNIOR_EMPLOYEE(Name) ISA EMPLOYEE(Name)
- 9.EMPLOYEE(Name) with [Status(full-time)].
- 10.SENIOR_EMPLOYEE(Name) with [Status(part-time)].

The clauses 1 to 6 are assertions which specify the instances of the classes. The clauses 7 and 8 specify the subclass definitions. The clauses 9 and 10 specify the methods.

We will describe how queries may be processed. Suppose a user wants to find out the status of an employee. The query is a message expressed as follows:

```
EMPLOYEE( ) : Status(X)?
```

The answer to the query is

```
X = full-time.
```

For the query **SENIOR_EMPLOYEE() : Status(Y)**, the answer is **Y = part-time**.

For the query **JUNIOR_EMPLOYEE() : Status(Z)**, the answer is **Z = full-time**.

One could include more information into the knowledge base by adding more rules. For example, the following rule defines the student-status of an employee.

```
employee_status(Anemp, T) :- Anemp, Anemp : Status(T).
```

That is, the **employee_status** of an employee instance is **T** if the instance is in the database and the **Status** of that instance is **T**. The answer to the query:

```
employee_status(EMPLOYEE(Fred), T)?
```

 is full-time while the queries

```
employee_status(EMPLOYEE(cat), T)?
```

```
and employee_status(EMPLOYEE(Jon), T)?
```

 both fail. However, the query

```
employee_status(SENIOR_EMPLOYEE(Jon), T)?
```

 is evaluated and the response is 'part-time'.

In the previous example, the query to find the **employee_status** of a senior employee is evaluated only if **SENIOR_EMPLOYEE(Name)** is substituted for **Anemp**. One way to ensure that the query is evaluated even if **EMPLOYEE(Jon)** is substituted for **Anemp** is to define sub-objects explicitly in the rule. This is shown below.

```
employee_status(Anemp, T) :- X sub Anemp, X, X:Status(T).
```

If the query

```
employee_status(EMPLOYEE(Jon), T)?
```

 is posed, the answer is **T = part-time**. However, the query

```
employee_status(EMPLOYEE(Cat), T)?
```

 still fails.

4.2 Handling Security Constraints

In this subsection we will describe how security constraints may be handled in object-prolog.

Example 1: Introduce an additional method **TITLE** to the class **EMPLOYEE**. **TITLE** returns the title of an employee instance. The code for **TITLE** is given below.

TITLE(N, T)

Begin

If N = Mary, T = Manager

If N = Fred, T = Secretary

If N = Jill, T = Engineer

If N = Harry, T = Vice-President

If N = Jack, T = Director

If N = Jon, T = President

end;

In the object-prolog representation, TITLE can be represented as a method or as a predicate. We will assume the latter representation. Therefore the following assertions are added to the knowledge base.

TITLE(JUNIOR_EMPLOYEE(Mary), Manager).

TITLE(JUNIOR_EMPLOYEE(Jack), Director).

TITLE(EMPLOYEE(Jill), Engineer).

TITLE(EMPLOYEE(Fred), Secretary).

TITLE(SENIOR_EMPLOYEE(Harry), Vice-President).

TITLE(SENIOR_EMPLOYEE(Jon), President).

Let the constraint enforced be the following: **An employee whose title is higher than Vice-President is secret.** This constraint is expressed in object-prolog as follows:

Level(Anemp, Secret) :- X sub Anemp, X, TITLE(Anemp, T), T > Vice-President

(Note: (i) The knowledge base has the inequality

Secretary < Engineer < Manager < Director < Vice-President < President

(ii) by having the sub clause sub, the constraint is applicable to any instance of EMPLOYEE, SENIOR_EMPLOYEE or JUNIOR_EMPLOYEE). We also assume that there are additional rules which ensure that if a piece of data is not secret then it is unclassified.)

Suppose an unclassified user poses a query to retrieve all senior employees. This query will be expressed as follows:

(SENIOR_EMPLOYEE(X), Level(SENIOR_EMPLOYEE(X), Unclassified))?

Since $\text{Level}(X, \text{Unclassified})$ is equivalent to $\text{NOT}(\text{Level}(X, \text{Secret}))$, the query is modified by the inference engine to the following:

(SENIOR_EMPLOYEE(X), TITLE(SENIOR_EMPLOYEE(X), T), T <= Vice-President)?

The answer to this query is 'Harry'.

Example 2: Let the constraint enforced be the following:
The Status of Jon is secret.

Since Jon is a senior employee, and the status of all senior employees is 'part-time'. If the clauses 3 and 10 (given in the previous subsection) are known to a user, this user can infer secret information. One approach is to ensure that both clauses cannot be classified at the unclassified level. In the absence of any tool to ensure the consistency of the knowledge base, some additional constraints are needed if both the clauses 3 and 10 are classified at the unclassified level. These additional constraints are the following:

Level(SENIOR_EMPLOYEE(Jon), Secret) :-

Release((SENIOR_EMPLOYEE(): Status(X)), Unclassified)

Level((SENIOR_EMPLOYEE(): Status(X)), Secret) :-

Release(SENIOR_EMPLOYEE(Jon), Unclassified)

That is, once the fact that Jon is a senior employee is released to an unclassified user, the status of all senior employees become secret. Similarly, once the status of all senior employees is released to an unclassified user, the fact that Jon is a senior employee becomes secret. Therefore, if an unclassified user requests for the status of senior employees after he has obtained the fact that Jon is a senior employee, the method associated with the superclass employee will be executed. The answer to the query is 'full-time' although the correct answer should be 'part-time'. If a secret user requests for the status of senior employees after getting the fact that Jon is a senior employee, the method executed is the one associated with senior employees. The answer returned is 'part-time'.

5. Conclusion

We have focussed on two prolog extensions; fuzzy prolog and object prolog, for security checking in database systems. Fuzzy prolog handles uncertain information and inferences when users possess real-world knowledge. Object prolog handles inferences in object-oriented database systems. We also discussed the query modification technique implemented by inference engines based on fuzzy prolog and object prolog.

References

- [BALD84] Baldwin J, and Zhou S., "A Fuzzy Relational Inference Language", *Fuzzy Sets and Systems*, Vol. 14, #2, November 1984, pp. 155-174.
- [BANE87] Banerjee J., et al., "Data Model Issues for Object-Oriented Applications", *ACM Transactions on Office Information Systems*, Vol. 5, #1, April 1987, pp. 3-26.
- [BELL75] Bell D., and LaPadula L., "Secure Computer Systems: Unified Exposition and Multics Interpretation", Technical Report MTIS AD-A023588, The MITRE Corporation, July 1975.
- [KEEF89] Keefe T., Thuraisingham M.B., and Tsai W.T., "Secure Query Processing Strategies", *IEEE Computer*, Vol. 22, #3, 1989, pp. 63-70.
- [LUNT89] Lunt T., "Inference and Aggregation, Facts and Fallacies", *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1989.
- [MART87] Martin T., et al., "The Implementation of Fprolog - A Fuzzy Prolog Interpreter", *Fuzzy Sets and Systems*, Vol. 23, 1987, pp. 119-129.
- [MORG87] Morgenstern M., "Security and Inference in Multilevel database and Knowledge Base Systems", *Proceedings of the ACM SIGMOD Conference*, San Francisco, CA, May 1987.
- [STAC89] Stachour P., and Thuraisingham M.B., "Design of LDV - a Multilevel Secure Relational Database Management System", Accepted for publication in *IEEE Transactions on Knowledge and Data Engineering*, 1989.
- [STON74] Stonebraker M., and Wong E., "Access Control in Relational Database Management Systems by Query Modification", *Proceedings ACM National Conference*, New York, NY, 1974.
- [THUR87] Thuraisingham M.B., "Security Checking in Relational Database Management Systems Augmented with Inference Engines," *Computers and Security*, Vol. 6, # 6, December 1987.
- [THUR89a] Thuraisingham M.B., "Security in Object-Oriented Database Systems", Accepted for publication in the *Journal of Object-Oriented Programming* 1989.
- [THUR89b] Thuraisingham M.B., "Mandatory Security in Object-Oriented Database Systems", *Proceedings of the (ACM) Object-Oriented Programming: Systems, Languages and Applications (OOPSLA) Conference*, New Orleans, LA, October 1989.
- [THUR89c] Thuraisingham M.B., "Secure Query Processing in Intelligent Database Management Systems", *Proceedings of the 5th Computer Security Applications Conference*, Tucson, Arizona, December 1989.
- [THUR89d] Thuraisingham M.B., "Towards the Design of a Secure Data/Knowledge Base Management System", Accepted for publication in *Data and Knowledge Engineering Journal*, 1989.
- [ZANI84] Zaniolo C., "Object-Oriented Programming in Prolog", *IEEE Logic Programming Symposium*, 1984.

Acknowledgement

The work reported in this paper was supported by the Department of Navy, SPAWAR.

Homework Problem #2

Title: MLS Database Design

Author: Gary W. Smith

General: The homework problem involves designing a multilevel secure database to support Smith Information Systems, Inc. (SI²).

Design Goals for the Homework Problem:

- Represent (close to) real world complexity both at the schema level and in terms of data content.
- Populate a database which is a representative sample of the type of departments and employees of an organization; but the size of the database is still trivially small (i.e. not sufficient for performance measurement).
- Illustrate the common (most, but not all) secrecy requirements of MLS applications.
- Provide a (mostly) realistic interpretation of the real world (i.e. only a few contrived requirements that stretch reality).
- Require both mandatory access control (MAC) and discretionary access control (DAC) with extensive use of MAC categories to implement need-to-know requirements.
- Illustrate integrity requirements for update and creation of data.

Security Requirements Included:

- Four Hierarchical-levels (HL).
- 9 Categories (CAT) for MAC need-to-know requirements.
- Data Elements with the following classification levels:
 - single HL with no CAT.
 - single HL with single CAT
 - multiple HL with no CAT
 - multiple HL with a single CAT
 - multiple HL with multiple CAT
- Classification is based upon one or more of:
 - association between attribute name and value (attribute-value pair)
 - association between two attribute-value pairs
 - association between an attribute-value pair and other (seeming) external data/information
 - attribute name by itself
 - all instances of data element are uniformly classified
 - instances are classified based a range of values

- specific instances are classified based upon the content of the data element
- specific instances are classified based upon some external criteria
- Cover story required for selected instances of a data element (used for disinformation)
- Unclassified and classified data elements used to represent the same entity (unclassified data element used for accounting purposes in unclassified system)

Rules:

1. The analysis that led to the set of data elements (Attachment B) accurately reflects what the users want to see. The data (as provided in Attachment E) reflects how the user is willing to see the data (i.e., codes as OK for some data elements but not for others).
2. If you need to create new data elements, want clarification, or have other questions about the requirements and content of the homework problem, contact the (surrogate) database security officer (DBSO), (Gary Smith GWSMITH@PENTAGON-OPTI.ARMY.MIL or GSMITH@GMUVAX2.GMU.EDU or (703) 764-6296 - Email is preferred).
3. Assume that a particular users' (or class of users) view of the database is determined by what he or she is authorized to see based upon their security clearance.
4. A planned cover story is used for selected sensitive projects. Users with the proper clearance should be shown *Project-Subject*. Users without the proper clearance should be shown the cover story project subject below:

<u>Project-Name</u>	<u>Project-Subject</u>	<u>Cover Story Project Subject</u>
SOLVER	CALS Proposal	CALS Proposal
ALPHA	HQ Relocation	HQ Relocation
ISTAR	Neural Net Security Kernel	Document Tracking System
EPCOT	Autonomous Land Spy Vehicle	Personnel System Redesign
BARBER	Pentagon Redesign	Inventory Control System
GEMINI	Presidential Inference Controller	Contracts Management System

5. If MAC requirements cannot be implemented with MAC mechanisms, use DAC mechanisms if available and describe how the DAC mechanisms can meet the intent of the MAC requirements.
6. Several DAC policies are specified. These policies must be implemented such that they are controlled by the organization (specifically the DBSO and Assistant DBSO), i.e., they must have the MAC-like qualities of being organizationally-granted (as opposed to user-granted) and are not able to be passed along by the users (as in DAC). MAC mechanisms may be used to enforce these requirements.
7. The intent is to show what capabilities can be provided by the DBMS and its facilities. If there is a need to go outside the DBMS and its facilities (i.e. to write code) to meet a re-

quirement, then just state that is what you would do--there is no need to write external code--we know that this can be done.

Tasks:

1. Design a multilevel secure database to support the security requirements of Smith Information Systems, Inc. as documented in attachments A through D.
2. Show the logical schema and physical schema/storage of the data resulting from the design.
3. Populate a database with the data for employees shown in Attachment E.
4. Generate the database queries in Attachment F.
5. Extra Credit: Include additional security requirements which are documented in Attachment G.

Attachments:

- A. Organization Description
- B. Database Requirements
- C. General Security Policy
- D. Security Requirements by Data Element
- E. Database Content
- F. Database Queries
- G. Extra Credit Requirements

Attachment A

Organization Description

General. The Smith Information Systems, Inc. (SI² - pronounced SI squared) is a high technology company which caters to the most sophisticated needs of the government and the defense industry. As such, the company conducts research and implements systems which are highly sensitive in nature. Senior management believes that protecting the organization's data as one of the most important aspects of company operations.

Organization. The company's organization is fairly typical. It has a President as chief executive officer; two Vice-Presidents which oversee the operations of the organization--one for headquarters staff and one for projects; headquarters staff departments (Personnel and Administration, Financial Management, Procurement, and Security); mission departments (Engineering, Integration, and Operations); personnel are assigned to staff or mission departments; and a matrix approach is used to allocate personnel to projects.

Department Manpower Summary. The following is a summary of the manpower for each department (not attached to a project) and project.

Executive Office

- President
- Executive Secretary
- Vice-President for Headquarters Staff
- Executive Secretary
- Vice-President for Project Management
- Executive Secretary
- Admin Assistant

Personnel and Administration

- Department Manager
- Secretary
- Personnel Analyst
- Management Analyst
- Personnel Clerk
- Admin Clerk

Financial Management

Department Manager
Secretary
Financial Analyst
Financial Clerk

Security Office

Department Manager
Secretary
Database Security Officer
Assistant DB Security Officer
Security Clerk

Procurement Department

Department Manager
Secretary
Procurement Analyst
Procurement Clerk

Engineering Department

Department Manager
Secretary
Tech Staff

Integration Department

Department Manager
Secretary
Tech Staff

Operations Department

Department Manager
Secretary
Tech Staff

Project 1 (CALs Proposal)

Project Manager
Secretary
Procurement Analyst
Financial Analyst
Personnel Analyst
Admin Assistant
Tech Staff-Eng
Tech Staff-Ops
Tech Staff-Int

Project 2 (HQ relocation)

Project Manager
Secretary
Admin Assistant
Financial Analyst
Management Analyst

Mission Projects (All other projects)

Project Manager
Secretary
Admin Assistant
Tech Staff-Eng
Tech Staff-Ops
Tech Staff-Int

Attachment B

Database Description

The database must contain data about two basic entities: employees and projects. The following attributes are required to be in the database. The names adequately describe the data element. Details of the exact meaning and content of each data element are contained in Attachment D.

Authorized to Update

Employee Entity

Employee-Num	Personnel Analyst (Myer) for all
Employee-Name	employees
Home-Address	Personnel Clerk (Chandler) for
Job-Title	employees assigned to HQ Depts
Department	Personnel Clerk (Bukowski) for
Assigned-Projects (1 or more)	employees assigned to projects
	-- and line depts (Eng, Ops, Int)
Salary	Executive Secretary
Profit-Share	Executive Secretary
Investigative-Status	Security Office
Credit Rating	Security Office
Skills (1 or more)	Security Office
Security-Clearance	Security Office

Project Entity

Project-ID	The Secretary or Administrative
Project-Name	Assistant working for the project
Project-Subject	to be updated
Project-Client	

Authorized to Create

Employee Entity	Personnel Analyst (Myer) and Personnel Clerk (Chandler)
Project Entity	DBSO and Assistant DSBO

Attachment C

General Security Policy

Senior management considers information (data) to be a valuable and important corporate resource. Management is therefore extremely concerned about security of data contained in its automated information systems. Management recognizes that both secrecy (protection from unauthorized disclosure) and integrity (ensuring the correctness and protecting from unauthorized modification of data) are part of the computer security needed. From a secrecy standpoint, the primary concern is protecting classified data through mandatory access controls, especially need-to-know requirements. But management is equally concerned with overclassification. It has proved extremely expensive to obtain clearances for the system high environment previously used. The organization has made a commitment to an MLS environment. Job responsibilities have been defined to allow to assignment of the minimum security clearances to as many employees as possible.

Security Policy Statements (PS):

PS1: All data brought under control of the DBMS are considered a corporate resource and will managed and controlled by the database security officer.

PS2: Access to corporate data must be made only through the DBMS. (I.e., no application programs can directly access corporate data.)

PS3: Only users which are explicitly authorized (as evidenced by their *Security-Clearance*, the proper hierarchical and need-to-know categories if applicable) will be able to access data (mandatory access control). (See rule 5 for substitution of DAC mechanisms.)

PS4: Overclassification of data is to be avoided.

PS5: Authorization to update data and create data entities will be strictly controlled within mandatory access controls (PS3) as indicated in Attachment B.

PS6: Unless otherwise specified in a secrecy constraint (see Attachment D) the system need not hide the existence of classified data in the database.

PS7: The classification of each secrecy constraint and integrity constraint is Unclassified unless otherwise stated in another secrecy constraint.

PS8: The fact that there is a cover story for classified projects is classified TS.

PS9: Users attempting queries which are partially unauthorized (e.g. ask for unauthorized data element to be displayed) will be given as much of the response that is possible within MAC and DAC requirements.

PS10: Users attempting queries for which the total response is unauthorized and involve classifications with MAC categories will be answered with the message "System error--see your DBSO."

PS11: Users attempting queries for which the total response is unauthorized and do not involve MAC categories will be answered with the message "Unauthorized Query"

Classification Description:

Hierarchical Levels. There is an ordered set of four secrecy levels ranging in level of sensitivity (from low to high) as follows:

- **Unclassified (U)** - No protection is necessary.
- **Company Private (C)** - Disclosure would result in minimal damage to the organization.
- **Sensitive (S)** - Disclosure would result in significant damage to the organization.
- **Truly Sensitive (TS)** - Disclosure would result in grave damage to the organization.

(Note: the letter abbreviations (i.e., U, C, S, TS) will be used through this document.)

Categories. There are 9 categories which are used to implement need-to-know criteria for mandatory access control as follows:

- **Category A:** Credit Rating Information.
- **Category J:** Indicates Type-1 data class.
- **Category K:** A sub-category of Category J. (Note: Category K and Category L are disjoint sub-categories of Category J.) **
- **Category L:** A sub-category of Category J. (Note: Category K and Category L are disjoint sub-categories of Category J.) **
- **Category Q:** Indicates Type-2 data class.
- **Category W:** Project ISTAR.
- **Category X:** Project EPCOT.
- **Category Y:** Project BARBER.
- **Category Z:** Project GEMINI.
-

** The "disjoint subcategory" requirement is enforced through the assignment of categories to projects (already done) and is, therefore, not a database design problem.

(Note: the syntax used for representing classifications and clearances in this specification is TS-XYZ. The levels (C, S, TS) and categories must be used, but the delimiter (in this case the "dash" can be system dependent.)

Attachment D

Security Requirements (by Data Element)

The security requirements are stated in terms of *secrecy constraints* (SC) and *integrity constraints* (IC) applicable to each data element. Also included is a brief description of the data element.

Employee-Num

Description: A unique identifier given to each employee upon assignment to the organization.

SC#1. *Employee-Num* classification is U for all values in all contexts.

IC#1. *Employee-Num* is a unique numeric code generated randomly by the system.

Employee-Name

Description: The name of the employee which may not be unique. (For convenience, the last name is used instead of last name, first name and middle initial)

SC#2. *Employee-Name* classification is U for all values in all contexts.

IC#2. *Employee-Name* is a string of characters.

Home-Address

Description: The employee's complete home address. (For convenience, only the street address is used.)

SC#3. *Home-Address* classification is U for all values in all contexts.

SC#4. The association of the *Home-Address* attribute-value pair with the attribute-values pairs *Employee-Name* or *Employee-Num* is protected by the following DAC requirements: only members of the Executive department and Personnel Analysts and all other managers (department or project) are authorized access for all managers down to front line supervisors; only personnel clerks and the secretary of the element to which the employee is assigned are authorized access for all other employees.

IC#3. *Home-Address* is an alpha-numeric string of characters.

Job-Title

Description: The employee's job title.

SC#5. The association of the *Job-Title* attribute-value pair with the *Employee-Name* or *Employee-Num* attribute-value pairs is classified C.

IC#4. *Job-Title* must be one of the following strings of characters: Pres, VP-HQ, VP-PM, Exec Secy, Dept Mgr, Proj Mgr, Tech Staff, DBSO, Ast DBSO, Sec Clk,

Adm Ast, Per Anal, Pers Clk, Fin Anal, Fin Clk, Proc Anal, Proc Clk, Mgt Anal,
Adm Clk, Secy.

Salary

Description: Provides the annual salary (in thousands of dollars) of the employee-- does not include bonuses or profit sharing.

SC#6. The association of the *Salary* attribute-value pair with *Employee-Name* or *Employee-Num* ranges in classification from U to S based upon the following criteria: *Salary* < \$25K is U; *Salary* > = \$25 and < \$75K is C; and *Salary* > = \$75K is S.

IC#5. *Salary* is numeric and non-zero.

Profit-Share

Description: Identifies the percentage of salary given to the employee for bonuses and/or share of profit.

SC#7. The association of the *Profit-Share* attribute-value pair with the *Employee-Name* or *Employee-Num* attribute-value pairs is classified S with the following DAC requirements: only the President, Vice Presidents and their Executive Secretaries are authorized to access.

IC#6. *Profit-Share* is numeric.

Investigative-Status

Description: Provides a code which represents the current status of ongoing investigations of the employee.

SC#8. The association of the *Investigative-Status* attribute-value pair with the *Employee-Name* or *Employee-Num* attribute-value pairs is classified C with following DAC requirement: only the President, Vice Presidents and members of the Security Department are authorized access.

IC#7. Two digit alpha-numeric field.

Credit Rating

Description: Indicates the credit rating of the employee.

SC#9. The Attribute name "*Credit Rating*" is classified S-A. (To hide the existence of the fact that the organization keeps this type of data on employees.)

SC#10. The association of the *Credit Rating* attribute-value pair with the *Employee-Name* or *Employee-Num* attribute value pairs is classified TS-A.

SC#11. SC#9 and SC#10 are classified TS-A.

IC#8. One digit numeric code from 1 to 10.

Department

Description: Indicates the department to which the employee is assigned.

SC#12. The association of *Department* attribute name with a value is classified S.

IC#9. An alpha-numeric string from the following list: Exec, Pers, Proc, Fin, Sec, Eng, Int, Ops.

Skill

Description: Indicates the specific skills (one or more) in which the employee is qualified.

SC#13. The association of any *Skill* attribute-value pair with the SI² organization is classified S.

SC#14. The association of the *Skill* attribute-value pair with the *Employee-Name* or *Employee-Num* attribute-value pairs is classified TS.

SC#15. The association of the *Skill* attribute-value pair with a *Project-Name* attribute-value pair is classified at which ever is highest--TS or the classification of the *Project-Name*. (E.g. *Skill* + a S-WZ project is TS-WZ.)

IC#10. An alpha string from one of the following: 3A (laser technician), 5B (neural net design), D7 (robotics engineering), 8F (operations engineering), C9 (integration engineering).

Assigned-Projects

Description: Indicates the *Project-ID* of all the projects (one or more) to which the employee is attached.

SC#16. The *Assigned-Project* attribute-value pair is Unclassified.

SC#17. The association of *Assigned-Project* with *Project-Name* is classified at the same level as the *Project-Name* attribute-value pair.

IC#11. Must be numeric from 1 to 6.

Security-Clearance

Description: Represents the highest hierarchical level to which the employee is trusted and the categories to which the employee has been granted a need-to-know.

SC#18. The *Security-Clearance* attribute-value pair is classified at the level of the hierarchical level of the value. (E.g., a value of TS-ZTQ is classified TS when combined with the attribute name.)

IC#12. One of hierarchical levels U, C, S or TS followed by the delimiter "-" followed by zero or more of the following categories: A, J, K, L, Q, W, X, Y, Z. (No embedded spaces or other punctuation included in the stored value.)

Project-ID

Description: A unique, unclassified code to represent projects

SC#19. The following mapping between is classified S.

<u>Project-ID</u>	<u>Project-Name</u>	<u>Project-ID</u>	<u>Project-Name</u>
3	ISTAR	4	EPCOT
5	BARBER	6	GEMINI

IC#13. Must be numeric from 1 to 6.

Project-Name

Description: The unique code name for a project.

SC#20. The *Project-Name* attribute-value is classified in the range from U to S on project-by-project basis as follows: SOLVER is C; ALPHA is U; ISTAR is S; EPCOT is S; BARBER is S; and GEMINI is S.

IC#14. The only authorized values are one of SOLVER, ALPHA, ISTAR, EPCOT, BARBER, GEMINI.

Project-Subject

Description: The name of the project which indicates the actual subject of the effort.

SC#21: The association of the *Project-Subject* attribute-value pair with the *Project-Name* attribute-value pair is classified as follows: SOLVER is C; ALPHA is U; ISTAR is S-LW; EPCOT is TS-JX; BARBER is TS-KY; and GEMINI is TS-KQZ.

SC#22: The association of the *Project-Subject* attribute-value pair with the cover story project subject is classified at the corresponding level of SC#21.

IC#15: The only authorized values are based upon the *Project-Name* as follows: CALS Proposal - SOLVER; HQ Relocation - ALPHA; Neural Net Security Kernel - ISTAR; Autonomous Land Spy Vehicle - EPCOT; Pentagon Redesign - BARBER; Presidential Inference Controller - GEMINI

SC#23: IC#15 is classified TS/JKLQWXYZ.

Project-Client

Description: The name of the organization which is the client for the project.

SC#24: The association of the *Project-Client* attribute-value pair with the *Project-Name* is classified at the classification level of the *Project-Name*.

SC#25: The association of the *Project-Client* attribute-value pair with this organization is classified S.

IC#16: The only authorized values are one of A, B, C, D.

Attachment F

Database Queries

Recurring Reports

Report RR1

Description: Listing of home addresses

Data Elements Displayed: Employee-Num, Employee-Name, Home-Address

Sorted By: Employee-Name

User Requesting Query: Myer (Per Anal)

Report RR2

Description: Listing of job titles

Data Elements Displayed: Employee-Name, Job-Title

Sorted By: Employee-Name

User Requesting Query: Chandler (Per Clk)

Report RR3

Description: Listing of job titles and departments

Data Elements Displayed: Employee-Name, Job-Title, Department

Sorted By: Employee-Name within Department

User Requesting Query: Bukowski (Per Clk)

Report RR4

Description: Listing of all employee salaries

Data Elements Displayed: Employee-Name, Salary

Sorted By: Employee-Name

User Requesting Query: Bukowski (Per Clk)

Report RR5

Description: Listing of all employee salaries and profit shares

Data Elements Displayed: Employee-Name, Salary, Profit-Share

Sorted By: Employee-Name

User Requesting Query: Bukowski (Per Clk)

Report RR6

Description: Listing of employees with Investigative-Status not equal to "1A"

Data Elements Displayed: Employee-Name, Investigative-Status

Sorted By: Employee-Name within Investigative-Status code

Requested By: Bukowski (Per Clk)

Report RR7

Description: Listing of employees with Credit-Rating other than "1"

Data Elements Displayed: Employee-Name, Credit-Rating

Sorted By: Employee-Name

User Requesting Query: Thomas (Sec Clk)

Report RR8

Description: Listing of all employees with specific skills

Data Elements Displayed: Employee-Name, Skills

Sorted By: Employee-Name

User Requesting Query: Kelley (Per Dept Mgr)

Report RR9

Description: Listing of employees assigned to projects

Data Elements Displayed: Employee-Name, Assigned-Project

Sorted By: Employee-Name

User Requesting Query: Smith, E. (Adm Ast, Exec Dept)

Report RR10

Description: Listing of all employees' security clearances

Data Elements Displayed: Employee-Name, Security-Clearance

Sorted By: Employee-Name

User Requesting Query: Thomas (Sec Clk)

Report RR11

Description: Listing of projects

Data Elements Displayed: Project-ID, Project-Name

Sorted By: Project-ID

User Requesting Query: Thomas (Sec Clk)

Report RR12

Description: Listing of all project subjects

Data Elements Displayed: Project-ID, Project-Name, Project-Subject

Sorted By: Project-ID

User Requesting Query: Falbo (Sec Secy)

Report RR13

Description: Listing of all project subjects

Data Elements Displayed: Project-ID, Project-Name, Project-Subject

Sorted By: Project-ID

User Requesting Query: Kelley (Per Dept Mgr)

Report RR14

Description: Listing of project clients

Data Elements Displayed: Project-Name, Project-Client

Sorted By: Project-Name

User Requesting Query: Brimer (DBSO)

Adhoc Queries**Report AQ1**

Description: Listing of home addresses for all employees

Data Elements Displayed: Employee-Name, Home-Address

Sorted by: Employee-Name

User Requesting Query: Chandler (Per Clk)

Report AQ2

Description: Listing of home address for all employees

Data Elements Displayed: Employee-Name, Home-Address

Sorted by: Employee-Name

User Requesting Query: Mahoney (Proc Dept Secy)

Report AQ3

Description: Listing of home address for Procurement Department employees

Data Elements Displayed: Employee-Name, Home-Address

Sorted By: Employee-Name

User Requesting Query: Mahoney (Proc Dept Secy)

Report AQ4

Description: Listing of home address for Financial Department employees

Data Elements Displayed: Employee-Name, Home-Address

Sorted By: Employee-Name

User Requesting Query: Wolcott (Fin Clk)

Report AQ5

Description: Listing of employees with salary above 50K

Data Elements Displayed: Employee-Name, Salary

Sorted By: Employee-Name

User Requesting Query: Rodriquez (Fin Anal)

Report AQ6

Description: Listing of employees with salary above 50L

Data Elements Displayed: Employee-Name, Salary, Profit-Share

Sorted By: Employee-Name

User Requesting Query: Rodriquez (Fin Anal)

Report AQ7

Description: Listing of investigative status other than "1A"

Data Elements Displayed: Employee-Name, Investigative-Status

Sorted By: Employee-Name

User Requesting Query: Bender (Exec Secy)

Report AQ8

Description: Listing of employee data for the Engineering Department

Data Elements Displayed: Employee-Name, Job-Title, Credit-Rating

Sorted By: Employee-Name

User Requesting Query: Kelley (Per Dept Mgr)

Report AQ9

Description: Listing of employee data for Operations Department

Data Elements Displayed: Employee-Name, Job-Title

Sorted By: Employee-Name

User Requesting Query: Wolcott (Fin Clk)

Report AQ10

Description: Listing of employee data

Data Elements Displayed: Employee-Name, Job-Title, Department

Sorted By: Employee-Name

User Requesting Query: Wolcott (Fin Clk)

Report AQ11

Description: How many employees have Skill "C9" ?

Data Elements Displayed: number

Sorted By: n/a

User Requesting Query: Thomas (Sec Clk)

Report AQ12

Description: Listing of Skills for employees assigned to projects

Data Elements Displayed: Employee-Name, Skill, Project-Name

Sorted By: Employee-Name within Project-Name

User Requesting Query: Brimer (DBSO)

Report AQ13

Description: Listing of skills for employees assigned to Project-Name equal ISTAR

Data Elements Displayed: Employee-Name, Skill

Sorted By: Employee-Name

User Requesting Query: Brimer (DBSO)

Report AQ14

Description: Listing of skills for employees assigned to Project-ID equal 3

Data Elements Displayed: Employee-Name, Skill

Sorted By: Employee-Name

User Requesting Query: Brimer (DBSO)

Report AQ15

Description: Listing of employees assigned to Project-Name equal GEMINI

Data Elements Displayed: Employee-Name, Project-Name

Sorted By: Employee-Name

User Requesting Query: Brimer (DBSO)

Report AQ16

Description: Listing of employees assigned to Project-ID equal 6

Data Elements Displayed: Employee-Name

Sorted By: Employee-Name

User Requesting Query: Brimer (DBSO)

Report AQ17

Description: Listing of skills of employees assigned to Project-Name = SOLVER

Data Elements Displayed: Employee-Name, Skill

Sorted By: Employee-Name

User Requesting Query: Brimer (DBSO)

Report AQ18

Description: Listing of employee data

Data Elements Displayed: Employee-Name, Department, Security-Clearance

Sorted By: Employee-Name

User Requesting Query: Myer (Per Anal)

Report AQ19

Description: List of project clients

Data Elements Displayed: Project-Client

Sorted By: Project-Client

User Requesting Query: Brimer (DBSO)

Report AQ20

Description: Listing of project subjects

Data Elements Displayed: Project-Name, Project-Subject

Sorted By: Project-Name

User Requesting Query: Wood (Proj Mgr, Proj 5)

Attachment G

Extra Credit

Implement the following security requirements:

E1: Managers can only see the *Salary* of their immediate subordinates. (Note that the company is too large to use MAC categories to meet this requirement.)

E2: Managers can only see the salary of ALL their subordinates (i.e., VP can see *Salary* of all employees subordinate to the managers the VP directly supervises. (Note: same constraint as E1-cannot use MAC categories to meet this requirement.)

E3: Only the Executive Secretary for the President (Opel) or Executive Secretary of the VP-HQ (Flippin) can update *Salary* and *Profit Share* for employees of all HQ Staff Elements.

E4: Only the Executive Secretary for the President (Opel) or Executive Secretary of the VP-PM (Bender) can update *Salary* and *Profit Share* for employees of all Projects and Line Departments (Engineering, Integration and Operations).

E5: The existence of the project GEMINI is extremely sensitive--so sensitive that any association between the text string "GEMINI" and this organization is classified TS.

E6: Map all of the data objects used in your design/implementation into the Trusted Database Interpretation (TDI) concepts of stored, encapsulated, and named objects.

E7: What is the minimum security clearance needed by the database designer?

Report on the Homework Problem

Gary W. Smith

George Mason University
School of Information Technology and Engineering
Fairfax, VA 22030

One of the most beneficial parts of the First RADC Database Security Workshop proved to be Rae Burn's Homework problem [Burn89]. It dramatically illustrated to the workshop participants some of the challenges in designing a multilevel database. Many of these challenges related to the application-dependent nature of designing databases as opposed to the abstract nature of operating system requirements. The primary motivation behind developing the Homework Problem for the 2nd Workshop was that papers and presentations on database security always use examples which are trivial. Many times it is difficult to really understand how a particular technique or approach works and more importantly, the implications of more complex requirements.

The homework problem involves designing a multilevel secure database. The design goals were:

- Represent (close to) real world complexity both at the schema level and in terms of data content.
- Populate a database which is a representative sample of the type of departments and employees of an organization; but the size of the database is still trivially small (i.e. not sufficient for performance measurement).
- Illustrate the common (most, but not all) secrecy requirements of MLS applications.
- Provide a (mostly) realistic (but unclassified) interpretation of the real world (i.e. only a few contrived requirements stretch reality).
- Require both mandatory access control (MAC) and discretionary access control (DAC) with extensive use of MAC categories to implement need-to-know requirements.
- Illustrate integrity requirements for update and creation of data.

General security policies were also stated that had a significant bearing on the design. Examples include:

PS1: All data brought under control of the DBMS are considered a corporate resource and will be managed and controlled by the database security officer.

PS4: Overclassification of data is to be avoided.

PS5: Authorization to update data and create data entities will be strictly controlled within mandatory access controls (PS3) as indicated in Attachment B.

PS6: Unless otherwise specified in a secrecy constraint (see Attachment D), the system need not hide the existence of classified data in the database.

PS7: The classification of each secrecy constraint and integrity constraint is Unclassified unless otherwise stated in another secrecy constraint.

Security Requirements Included:

- Four Hierarchical-levels: U, C, S, and TS.
- 9 Categories for MAC need-to-know requirements.
- Data Elements with the following classification levels:
 - single hierarchical level with no categories.
 - single hierarchical level with single category.
 - multiple hierarchical levels with no categories.
 - multiple hierarchical levels with a single category.
 - multiple hierarchical levels with multiple categories.
- Classification is based upon one or more of:
 - association between attribute name and value (attribute-value pair)
 - association between two attribute-value pairs
 - association between an attribute-value pair and other (seeming) external data/information
 - attribute name by itself
 - all instances of data element are uniformly classified
 - instances are classified based a range of values
 - specific instances are classified based upon the content of the data element
 - specific instances are classified based upon some external criteria
- Cover story required for selected instances of a data element (used for disinformation)
- Unclassified and classified data elements used to represent the same entity (unclassified data element used for accounting purposes in unclassified system)

Tasks:

1. Design a multilevel secure database to support the security requirements of Smith Information Systems.
2. Show the logical schema and physical schema/storage of the data resulting from the design.
3. Populate a database with the data for employees.
4. Generate a set of test database queries.

Homework Results.

The schema for the homework problem, as shown in Figure 1, is used to illustrate the basic types of secrecy constraints and how the various DBMSs approached the solution. The syntax and meaning of the graphical technique used to display the schema is provided below.

The basic constructs of the model are shown in Figure 2. There are three types of objects: abstract objects (circles) represent objects of interest in the application; attributes (rectangles) represent characteristics of an abstract object; and identificates (egg-shaped) represent a special type of attribute--one that serves to identify the abstract object. An identificate is either a key (i.e., uniquely identifies the abstract object) or a *near-key* (i.e., a non-unique attribute that can be used to identify the abstract object most of the time). An association, which is the relationship between two abstract objects or between an abstract object and one of its attributes/identificates, is shown as a line connecting two objects. Bold lines and text are used to indicate classified objects and associations. The classification level(s) are shown next to the object.

There are two major entities of interest in the application domain: Employee and Project. Throughout this paper, attributes will be referenced by combining the abstract object name (e.g., Employee) with the attribute name (e.g. Name) when necessary to provide an unambiguous name (e.g., Employee.Name). Relations will use the following syntax: relation-name (attribute-1, attribute-2, ... , attribute-n) followed by a classification level for the relation (e.g., U, C, S, or TS).

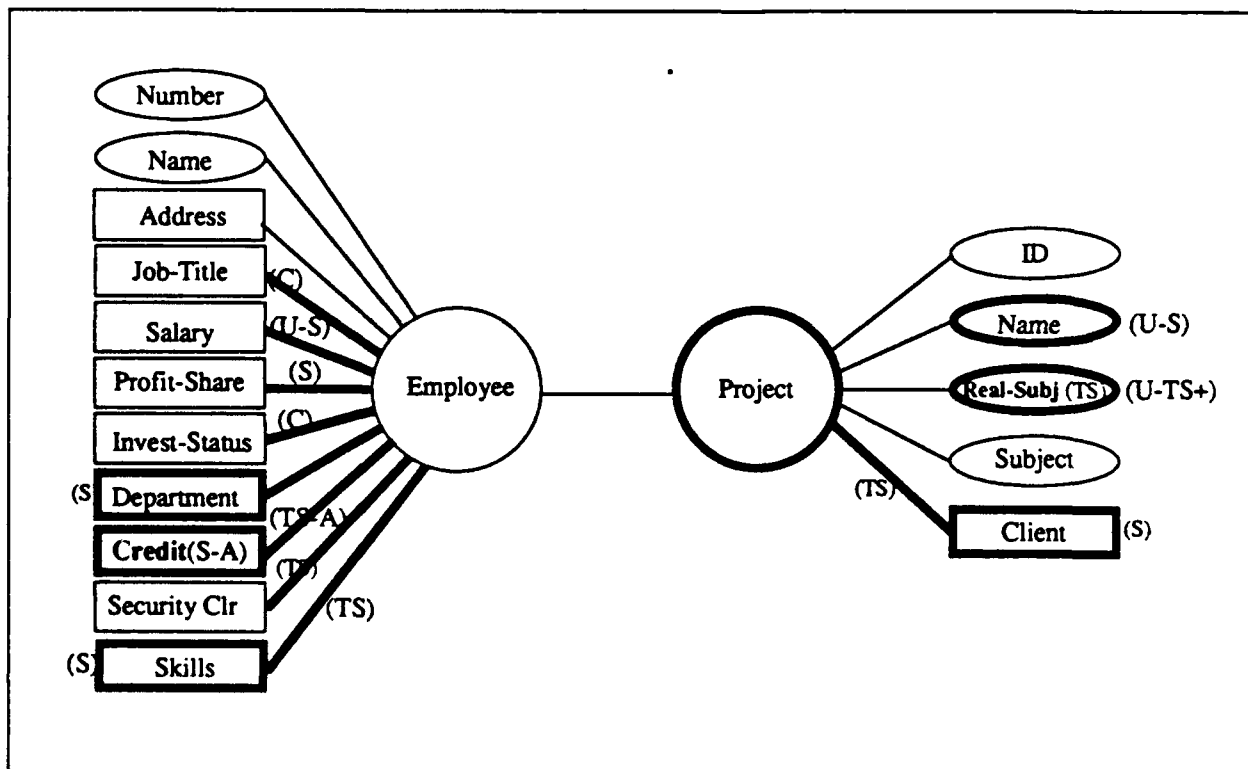


Figure 1: Conceptual Data Model

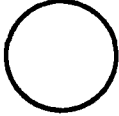
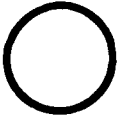






<u>Type</u>	<u>Unclassified</u>	<u>Classified</u>
Abstract Object		 (S)
Identificate		 (U-P)
Attribute		 (U,S)
Association		 (S)
External Identifier	Project	Project (C)

Figure 2: Secrecy Constructs

Database Management Systems. The approaches used by four DBMSs will be discussed in this report.

Sybase Secure SQL Server--a product available from Sybase and jointly developed with TRW. Sybase has two secure products underway: one is aimed at satisfying a B1 level of trust and one is aimed at satisfying a B2 level of trust. The Sybase standard DBMS (non-secure) has been on the market for several years. One secure version (aimed at B1) runs on top of the Ultrix operating system. It uses MAC labels on tuples and provides DAC on relations. The B2 secure version runs on bare hardware. All remaining references to Sybase in this paper refer to the B1 version which was used to implement the homework problem and presented at the workshop.

A1 Secure DBMS (ASD)--a prototype trusted DBMS built by TRW. ASD is assumed to run on top of a trusted operating system (currently ASOS-the Army Secure Operating System). ASD uses tuple-level MAC and relation-level DAC.

LOCK Data Views (LDV)--a prototype trusted DBMS designed by Honeywell. A special prototype of the Honeywell design of a trusted DBMS was used to demonstrate an implementation of the homework problem. LDV is designed to run on Honeywell's LOCK operating system.

Oracle--a prototype trusted DBMS designed by Oracle Corp. Oracle uses the TCB subset approach similar to SRI's SeaView design specifications and will run on top of a variety of trusted operating systems up to A1. It offers MAC labeling down to the element level and DAC on views and relations.

TRW (using Sybase), Oracle, and LDV implemented the homework problem, including loading the database and executing the queries. ASD did a paper design without any implementation.

Possible modifications to Tandem Computers' operating system to implement the homework problem were also presented at the workshop. Since the Tandem solution is dependent on the Tandem-unique architecture and mechanisms (which space does not allow to be presented here), it will not be discussed in this report.

General Approach. The general approach taken by all DBMSs, to one degree or another, was to decompose a relation by security level into multiple relations--either vertically or horizontally or both. For example, the following simple employee relation

Employee (Employee-Number, Employee-Name, Job-Title, Salary)

where number is the key, name is always U, job-title is always C, and salary is either U, C or S, would be decomposed vertically into three relations:

Employee-Name (Employee-Number, Employee-Name) U

Employee-Job-Title (Employee-Number, Job-Title) C

Employee-Salary (Employee-Number, Salary) U, C, S

Employee-Salary is still multilevel, but would be decomposed horizontally by security level into three relations, each containing instances of Employee-Salary at the same level:

Employee-Salary-U (Contains all U instances of Employee-Salary)

Employee-Salary-C (Contains all C instances of Employee-Salary)

Employee-Salary-S (Contains all S instances of Employee-Salary)

Figure 3 graphically shows this decomposition concept. For Sybase and ASD, the vertical decomposition is a part of the logical database design seen by the user; however, no horizontal decomposition is necessary since row-level labeling is the basis for the DBMS to enforce MAC. The SeaView and Oracle approach requires all data objects to be of a single security level so that the underlying operating system can enforce MAC. However, the logical database design seen by the user is multilevel; the decomposition, vertically and horizontally, is accomplished by the DBMS and is transparent to the user. DAC authorizations on the multilevel view of the database are automatically inherited by the decomposed relations. In LDV, the amount of vertical decomposition seen by the user is a function of database design; multilevel relations or single-level relations or both are available to the database designer to model the application domain.

Specific Approaches. The following sections describe design solutions for requirements (policies and constraints) of the homework problem.

Access to DBMS-Managed Data. PS2: Access to corporate data must be made only through the DBMS. (I.e., no application programs can directly access corporate data.) This policy is essential in order to ensure that the DBMS has complete control of the database. All the DBMSs rely on the underlying operating system to enforce this separation. Sybase relies on Ultrix. ASD uses a combination of storing all data at the system-high security level and integrity constraints to enforce this requirement. LDV uses the typing and domain mechanisms of the LOCK operating system. Oracle uses whatever mechanisms are avail-

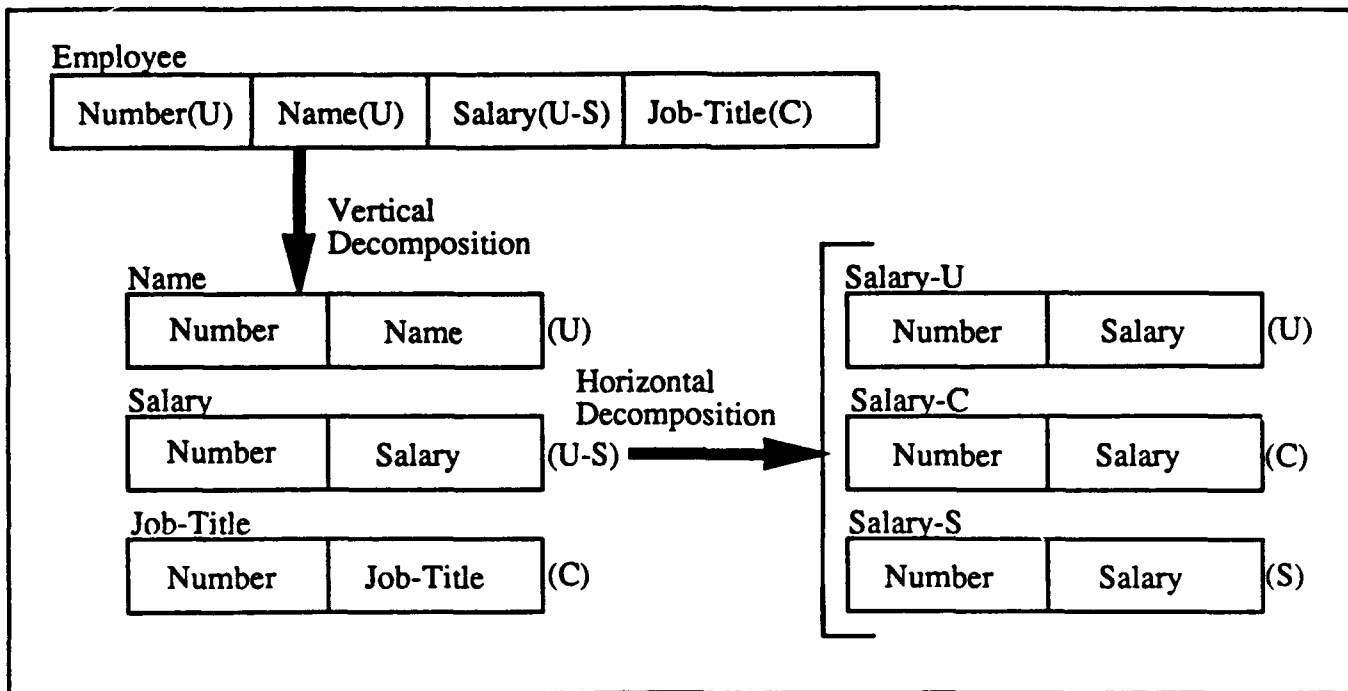


Figure 3. Decomposition by Security Level

able in the underlying operating system. In the case of the SeaView project it would be the ring domain architecture of the GEMSOS operating system:

Uniformly Classified Attribute. Secrecy Constraint (SC)#12: The association of *Department* attribute name with a value is classified S. The *Department* attribute of the *Employee* entity is used to illustrate the case in which all instances of an attribute are uniformly classified--in this example all instances of attribute *Department* are classified S.

The approach used by ASD and Sybase for this requirement was to vertically decompose the *Department* attribute into a separate relation, rather than putting it into one of the *Employee* relations:

Department (Employee-Number, Employee-Department) S

Oracle included *Department* in a multilevel relation, but would automatically decompose into the above relation. LDV included the *Department* attribute in a base relation, *Employee-Base* (Employee-Number, Employee-Name, Department, Salary, Security Clearance). One capability that LDV has is to specify a default, or minimum, security level for an attribute in the data definition language. In this case, LDV can enforce the fact that all instances of *Department* are classified S by establishing the maximum and default security level as S. The same is true for SeaView. Sybase provides a minimum security level.

Classified Attribute--Conditional on Content. SC#20: The *Project-Name* attribute-value is classified in the range from U to S on a project-by-project basis as follows: SOLVER is C; ALPHA is U; ISTAR is S; EPCOT is S; BARBER is S; and GEMINI is S. Where uniform classification of an attribute requires vertical decomposition, conditional classification of an

attribute requires both vertical and horizontal decomposition (transparent to the user) for Oracle. (A full SeaView implementation would use classification constraints to assign the correct security level.) Since ASD and Sybase use tuple-level labeling, they can distinguish between security levels within a relation and need not horizontally decompose. The security level for each tuple is based upon the user's logon level at the time the tuple is entered; thus the security depends upon the user knowing enough to be logged in at the appropriate security level.

Uniformly Classified Association. SC#5: The association of the *Job-Title* attribute-value pair with the *Employee-Name* or *Employee-Num* attribute-value pairs is classified C. The approach as proposed by [Lunt89] could be used by ASD, Oracle, and Sybase to meet this requirement to add an additional data element, e.g., *Employee-Code*, and use three relations:

Employee-Name (Employee-Number, Employee-Name) U

Job-Title (Employee-Code, Job-Title) U

Employee-Job-Title (Employee-Number, Employee-Code) C

The only way to associate *Employee-Number* with *Job-Title* is to go through the *Employee-Job-Title* relation, which is classified C.

The LDV approach has just one relation, *Employee-Title* (*Employee-Number*, *Job-Title*) which is classified U. With this approach, a user can protect the association by stating a context constraint that the DBMS uses to enforce the fact that the association of the two data elements is classified. This approach keeps history files of previous queries to detect when returning data will allow a user to make a sensitive association.

Conditionally Classified Association. SC#6: The association of the *Salary* attribute-value pair with *Employee-Name* or *Employee-Number* ranges in classification from U to S based upon the following criteria: *Salary* < \$25K is U; *Salary* > = \$25 and < \$75K is C; and *Salary* > = \$75K is S. The conditions for classification are based upon a range of values. Taking the same approach at the logical level as taken with uniform classification, a classified association that is conditional appears the same as a uniformly classified association--three relations are required:

Employee (Employee-Number, Employee-Name) U

Employee-Salary (Employee-Number, Employee-Code) U, C, S

Salary (Employee-Code, Salary) U

However, both Sybase and ASD implemented this constraint with a single relation containing *Employee-Number* and *Salary* in which the classification of each row is based on the value of *Salary*. This approach protects the association, but does not allow *Salary*, by itself, to be retrieved at the U level--which was the intended requirement stated by the homework problem.

Oracle took a different approach since it is able to restrict access based on views which are protected by DAC. The *Employee* relation has *Employee-Number* and *Salary* (in addition to other attributes). An additional relation, *Salary-Authorization*, is created which contains a key attribute, called *Label* (representing a security level), and the attribute

Maxsal (representing the maximum salary for that security level). In this case Salary-Authorization would contain these three tuples:

<u>LABEL</u>	<u>MAXSAL</u>
Unclass	25000
Confidential	75000
Secret	NULL (means there is no known limit)

Several views are then used to restrict access as described below: Salaries alone can be accessed through the salary view:

```
SELECT Salary from Employee
```

Employee numbers alone can be accessed through the emp-num view:

```
SELECT employee-number from Employee
```

Employee numbers together with salaries must be accessed through the emp-sal view:

```
SELECT Employee-Number, Salary from Employee
WHERE Salary < = (SELECT nvl(Maxsal, Employee-Salary)
                  from Salary-Authorization
                  WHERE label = :user-label)
```

The nvl function, nvl(x,y) returns x if x is not null and y if x is null. The current logon security level of the user's process is :user_label.

All users would be granted privileges to select from the three views, but not from the base tables. The salary and emp-num views would return the same values for any user, but the emp-sal view would return only those emp-sal pairs that the user is allowed to see.

Classified Attribute and Association. SC#13: The association of any *Skill* attribute-value pair with the organization is classified S. SC#14: The association of the *Skill* attribute-value pair with the *Employee-Name* or *Employee-Number* attribute-value pairs is classified TS. These constraints are logically the same as a simple classified association. The fact that the base security level for *Skill* is S rather than U does not change the problem or the form of the solution--it only changes the security level of the Skills relation. The relations are:

Employee (Employee-Number, Employee-Name) U

Employee-Skills (Employee-Number, Skill-Number) TS

Skills (Skill-Number, Skill) S

Classified Name. SC#9: the Attribute name "*Credit Rating*" is classified S-A. (This constraint is intended to hide the fact that the organization keeps this type of data on employees.) This constraint is an example of classified "meta-data." All systems allow meta-data to be classified. In fact, meta-data is treated like any other data and is stored in relations. In the case of Sybase and SeaView, all of the meta-data about a relation (relation name and column names) are classified at the same level. Thus, the Credit-Rating attribute must be vertically decomposed into a separate relation classified at S-A.

Classified Association between two attributes. SC#15: The association of the *Skill* attribute-value pair with a *Project-Name* attribute-value pair is classified at which ever is highest--TS or the classification of the *Project-Name*. A database design could be generated to implement this constraint as follows:

Employee (Employee-Number, Employee-Name, Project-ID) U

Project (Project-ID, Project-Name) U, C, S

Skills (Skill-Number, Skill) S

Employee-Skills (Employee-Number, Skill-Number) TS

A constraint between attributes of two different entities (or columns of two different relations) is potentially insecure if there is another relationship between the two entities. In particular, a database design for the homework problem, as stated, is not inherently insecure. The association between Employee-Skills and Project-Name is TS even though the association of an Employee and Project-Name is U-S based upon the classification of Project-Name. The design is secure because the association between Employee and Employee-Skill is TS. If on the other hand, the Employee/Employee-Skill association was S, the design would be inherently insecure--TS data (the association between Employee.Skill and Project.Name) could be easily derived by combining the unclassified association between Employee and Project-Name (which is U-S) with the S association between Employee and Employee-Skill.

Additional Access Restrictions. Several additional constraints on access were stated that went beyond normal MAC requirements. These constraints were the biggest challenges. They are difficult to implement in a straightforward manner.

Read Access to an attribute. SC#8: the association of the *Investigative-Status* attribute-value pair with the *Employee-Name* or *Employee-Number* attribute-value pairs is classified C with following access requirement: only the President, Vice Presidents and members of the Security Department are authorized access.

For the systems that only provide DAC on relations, this constraint can be implemented by further decomposing the *Investigative-Status* into a separate relation. Before this decomposition all associations which were classified C could be in the same relation:

Employee (Employee-Number, Employee-Name) U

Employee-Number-Code (Employee-Number, Employee-Code) C

Employee-Job-Title-Invest (Employee-Code, Job-Title, Invest-Status) U

To satisfy this constraint, the Employee-Job-Title-Invest relation would be decomposed into the following:

Job-Title (Employee-Code, Job-Title) U

Employee-Invest-Status (Employee-Code, Invest-Status) U

For systems which also provide DAC on views, the logical decomposition is not required. Different views can be established for different users--the attribute is not available in the views for those who are not allowed access, but is for the others. Oracle is also im-

plementing a "role" mechanism, whereby access to views can be based upon a user's assigned role.

Read Access to an Attribute--Conditional on External Criteria. SC#4: the association of the *Home-Address* attribute-value pair with the attribute-values pairs *Employee-Name* or *Employee-Num* is protected by the following access requirements: only members of the Executive department and Personnel Analysts and all other managers (department or project) are authorized access for all managers down to front line supervisors; only personnel clerks and the secretary of the element to which the employee is assigned are authorized access for all other employees.

This constraint is even more difficult when DAC is applied at the table level. In essence all relations that contain employee data, for ASD there are six relations, must be duplicated--one set contains data on managers, the other set contains data on all other employees. With this solution, queries about employees will require joins across two sets of relations.

Sybase did not implement this constraint but proposed three possible solutions: use a complex stored procedure; add an extra field containing the type of employee, and a stored procedure could use the field to enforce access; and assign mandatory categories. Using procedures adds the need to consider the relative security of the procedures. In Sybase, procedures are not named objects and thus access to procedures is not enforced as part of the security policy.

The Oracle approach would use views to enforce the constraint and not require logical decomposition. In Oracle, views are named objects and thus access to views is enforced as part of the security policy.

Update Access to an Attribute--Conditional. Two extra credit constraints were stated relating to update: E3-Only the Executive Secretary for the President (Opel) or Executive Secretary of the VP-HQ (Flippin) can update *Salary* and *Profit Share* for employees of all HQ Staff Elements; E4-Only the Executive Secretary for the President (Opel) or Executive Secretary of the VP-PM (Bender) can update *Salary* and *Profit Share* for employees of all Projects and Line Departments (Engineering, Integration and Operations).

Once again, if DAC is applied to relations, then multiple sets of tables containing employee data must be generated--one set for HQ Staff Elements and one for projects and line departments. The proliferation of sets of employee relations, especially if there is a different criteria for read versus update access, is a serious problem.

If DAC on views is used to implement this constraint, then updating through views must be allowed--a capability that is not allowed in most current relational DBMSs (but is theoretically possible in some cases).

Miscellaneous Requirements.

Cover Story. A cover story was required for the subject of a project. The approach taken to model the cover story in the database schema (Figure 1) was to provide two attributes: *Real-Subject*, with each instance classified TS and one or more categories; and *Subject*, which was an unclassified cover story. The attribute name, *Real-Subj*, was classified TS to

meet the requirement that the fact that there was a cover story for a project's subject was TS.

Alternately, each DBMS can use its polyinstantiation capability to provide a U tuple (the cover story) and a TS + tuple (the real subject) for each project.

Hiding the Existence of Classified Data. PS6: unless otherwise specified in a secrecy constraint the system need not hide the existence of classified data in the database. All four systems automatically polyinstantiate when instances of data are attempted to be inserted at a different security level than already exists in the database.

Behavior Issues. (PS9-11) How the system would respond to unauthorized behavior received minimal discussion. This is an important subject because it is critical to understand the policies that are implicitly implemented in a DBMS as well as those that are explicitly implemented.

Conclusions

The first conclusion by those participating in the Homework problem was that it was challenging to implement and it was a meaningful effort. The results clearly indicate that there are significant differences between the DBMSs reviewed.

The DBMSs differ in how much decomposition the user sees. This has a direct impact on the complexity forced upon the user. LDV and Oracle provide logical multilevel relations, whereas ASD and Sybase require vertical decomposition by security level. At a lower level, transparent to the user, LDV and Oracle must further decompose horizontally, whereas ASD and Sybase need not further decompose (but this is transparent to the user). There was some question about the performance impact of all this decomposition. The database community advocates this kind of "normalization" as part of effective database design. It is too early to tell what, if any, performance degradation will be experienced and how much can be attributed solely to decomposition for security reasons.

The previous conclusion related to design considerations to implement MAC requirements. Restrictions that are value-dependent, which were stated as DAC requirements, had a greater impact on those systems which only provide DAC on relations. Relations must be further decomposed based upon the access criteria for read and update--if they are different. The resulting fragmentation of relations is unpleasant at best. Providing DAC on views which restrict access avoids the additional decomposition. This solution appears to be acceptable for read permissions, but all of the ramifications of updating through views are not apparent at this time.

All the systems automatically polyinstantiate data either for intentional cover stories or for inadvertent/malicious attempts to update data that already exists in the database at a higher security level.

One issue the Homework Problem attempted (unsuccessfully) to introduce related to the behavior of the DBMS when faced with apparent or real unauthorized behavior by the user. This is an important issue, because as these DBMS are developed, the policies on how the DBMS should respond to unauthorized behavior is imbedded in the implementation. Yet how the DBMS responds is a policy issue, and the response to the user for the same action may need to be different, depending upon which data is involved in the un-

authorized action. The relative flexibility that a DBMS provides the security officer (and users) to implement effective security policies for an application domain is an issue that requires additional attention in the research community.

References

[Burn89] Burns, R., The Homework Problem, In *Research Directions on Database Security*, Lunt, T. F. ed., Springer-Verlag, 1989.

[Lunt89] Lunt, T. F., Aggregation and Inference: Facts and Fallacies, *Proceedings of the 1989 IEEE Symposium on Security and Privacy, May, 1989, pp. 102-109.*

Database Design with Row Level MAC and Table Level DAC

Thomas H. Hinke
TRW Defense Systems Group
One Space Park
Redondo Beach, Calif. 90278

1 Abstract

This paper describes an exercise in which the mandatory and discretionary security policy of TRW's A1 Secure DBMS prototype (ASD) was used to address the Homework Problem #2 [Smith89] presented at the Second RADC Database Security Workshop.

2 Introduction

TRW's ASD is a prototype trusted database management system that TRW is building to satisfy the highest level of security. In the absence of a finalized Trusted Database Interpretation (TDI), TRW has structured the system to satisfy the requirements of DoD 5200.28-STD (the Orange book)[DoD85] and TRW's interpretation of what can be expected in an approved TDI. Currently ASD represents a research prototype and not a product.

ASD is a relational DBMS that supports the SQL language. It enforces both mandatory access control (MAC) and discretionary access control (DAC). The granularity of control for MAC is the row - meaning that each row in a relation or table can have its own distinct security level label. The granularity of control for DAC is the whole table, meaning that each table can have its own distinct DAC access requirements.

The data dictionary is stored within ASD tables and comes under the ASD security policy. The data dictionary can thus be stored with any level of protection from unclassified to system high, the highest level of classification provided by ASD.

3 Support for the Homework Problem

This section presents the database design for using ASD to support the security requirements of the homework problem.

Each table is shown as **table-name**(field-name1, field-name2, ... field-nameN).

The following headquarters department tables contain information from the following departments: Personnel and Administrations, Financial Management, Procurement, and Security. For each of the headquarters department tables the DAC update restrictions are that only the Personnel Analyst (Myers) and the Personnel Clerk (Chandler) can update these tables. The headquarters department tables include:

- **Name** (Employee-number, Employee-name) - The rows are unclassified. DAC read is public.
- **Management-Addresses** (Employee-number, Address) - The rows are unclassified. DAC read is limited to the Executive Department, Personnel Analyst (Myers) and all managers
- **Assigned-Project** (Employee-number, Project) - The classification of the rows are U, C or S depending upon the project. DAC read is public.
- **Job-Title** (Employee-number, Job-title) - The classification of the rows are C. DAC read is public.
- **Department** (Employee-number, Department) - The classification of the rows are S. DAC read is public.
- **Assigned-Projects** (Employee-number, Project) - The classification of the rows are U, C or S depending upon the project. DAC read is public.
- **Non-Management-Department X-Addresses** (Employee-number, Project) - The rows are unclassified. There exists one of these tables for each of the four headquarters departments. DAC read is Personnel Clerk (Chandler) for each table and the department secretary for the table that contains the non-management employees for the secretary's department.

The next group of tables support the projects and line departments. The DAC update restrictions applying to all of these tables is that only the Personnel Analyst (Myers) and the Personnel Clerk (Bukowski) can update the tables. The reason that these tables are distinct from the previous headquarters department tables is that the Personnel Clerk that has update access to these tables (i.e. Bukowski) is different from the Personnel Clerk that has update access to the headquarters department tables.

The project and line department tables include:

- **Name (Employee-name, Employee-number)** - The rows are unclassified. DAC read is public.
- **Management-Addresses (Employee-number, Address)** - The rows are unclassified. DAC read includes the Executive Department, Personnel Analyst and all managers.
- **Assigned-Projects (Employee-number, Project)** - The classification of a row is U, C or S depending upon the project. DAC read is public.
- **Job-Title (Employee-number, Job-title)** - Classification of the rows is C. DAC read is public.
- **Department (Employee-number, Department)** - The classification of the rows is S. DAC read is public.
- **Non-Management-Addresses (Employee-number, Project)** - The rows are unclassified. There exists one of these tables for each of the projects and line departments, requiring eight such tables. DAC read is Personnel Clerk (Bukowski) for each table and the department or project secretary for the table that contains the non-management employees for the secretary's department or project.

The following set of tables have distinct DAC read and update requirements for each table.

- **Salary (Employee-number, Salary)** - The classification of a row is U, C or S based on amount of salary. DAC update is Executive Secretary. DAC read is public.
- **Profit-Share (Employee-number, % Salary)** - The classification of the rows is S. DAC update is Executive Secretary. DAC read is President, Vice-president and their Executive Secretaries.
- **Investigative-Status (Employee-number, Investigative-status)** - The classification of the rows is C. DAC update is Security Officer. DAC read is President, Vice-president and members of the Security Department.
- **Credit (Employee-number, Credit-rating)** - Classification of the rows is TS-Category A. DAC update is Security Office. DAC read is public. Data dictionary entries for this table are classified at S-Category A.
- **Skills (Employee-number, Skill)** - The classification of the rows is TS. DAC update is Security Office. DAC read is public.
- **Clearance (Employee-number, Security-clearance)** - The classification of the rows is U, C, S or TS depending upon the level of the clearance. DAC update is Security Office. DAC Read is public.

The following tables contain information about projects. Each table is to be updated by the secretary of the respective project. With table level DAC, these would have to be handled with one set of tables for each project and only a single row in all but the Project-Subject table. An alternative is to designate a single update authority to perform all of the DAC update access, and have only one set of tables supporting all projects. The table set that would exist for each project is as follows:

- **Project-Subject (Project-Id, Subject)** - For each project, there exists two rows in the table. One row, represents an unclassified description of the project in which the Project-identifier is a unclassified and the Subject is the unclassified cover story. The second row associated with each project contains the Project-Identifier and the actual Project-subject which ranges from unclassified to TS-Compartmented. ¹ DAC update is the secretary of the project. DAC read is public.
- **Project-Name (Project-Id, Project-name)** - This table contains rows that are classified S, since the association of Project-ID with Project-name is classified S. DAC update is the secretary of the project. DAC read is public.
- **Project-Client (Project-Id, Project-client)** - Since the association of Project-Client with the organization is classified S, all of the rows are classified S. This overrides the requirement that the project-client association is to be classified at the same level as the project name since S dominates the classification of all project names. DAC update is the secretary of the project. DAC read is public.

4 Conclusions

As exhibited by the database design presented in this paper, a secure DBMS that supports table level DAC and row level DAC can be used to support a database such as the homework problem that has fairly stringent DAC requirements. The only situation in which row level DAC would have been a useful feature is with respect to the project information, since each project secretary was to have update privilege for their respective projects. However, for this database design, this was handled quite adequately with single row tables. We believe that the added overhead required to have a small number of single row tables is preferable to having DAC on each row of every table.

¹Note that there appears to be some conflict in the requirements of the problem. Security Policy Statement 8 indicates that the fact that there is a cover story for classified projects is TS. However, the fact that the cover story is unclassified and the classification of Project-Subject ranges from U to TS-Compartmented means that for a project with a subject that is classified less than TS, the authorized user can infer that cover stories exist, and this inference can be made at less than the TS level.

References

- [Smith89] Smith, Gary W., Homework Problem #2 - Title: MLS Database Design, Second RADC Workshop on Database Security, Bethlehem, New Hampshire, May 1989.
- [DoD85] Department of Defense Trusted Computer System Evaluation Criteria. U.S. Department of Defense. DOD 5200.28-STD. December 1985.

MLS Implementation Using the Sybase Secure SQL Server

**Melody F. O'Brien
Edward D. Sturms**

**TRW Federal Systems Group
2750 Prosperity Avenue
Fairfax, VA 22031**

1. Introduction

TRW and Sybase have been involved in the research and development of a commercial, multilevel secure relational DBMS since 1985. This research has produced the Sybase Secure SQL Server, a product aimed at satisfying the class B2 level of trust found in both the DoD Trusted Computer Systems Evaluation Criteria and its expected companion document for databases, the Trusted Database Interpretations.

This paper presents TRW's solution to a real-world problem involving multi-level security. This problem was written by Gary W. Smith and presented as a homework problem entitled "MLS Design". TRW took the initiative to implement the MLS Design requirements in an existing multi-level secure relational database management system, the Sybase Secure SQL Server. The Sybase product provides all of the security and integrity features needed to solve this problem.

The requirements stated in the MLS Design problem contained a variety of integrity and security constraints. The system requirements include assigning security labels to single as well as pairs of attributes. To satisfy this requirement using a relational DBMS with row-level security presents a challenge to the database designer. In addition, security requirements based on a range of values in a field, and discretionary access based on the value of a field, are also included in the problem. Integrity constraints on lists of values and classifications of requests associated with string values, such as the company name, are introduced. As a result of the mixture of requirements listed in the assignment, TRW chose to approach the problem by first satisfying the requirements for security of the data, then implementing the integrity constraints. This ordering is also used throughout this paper.

The objective in solving this problem was to allow as many users as possible to have access to as much data as possible without violating security constraints. Integrity constraints apply to the entry and update of data, rather than its access. In some cases, our implementation uses upward classification of the data for single attributes to prevent disclosure of sensitive data.

Alternate methods which prevent this are discussed in the paper. However, these alternatives require strict enforcement of user access methods and to some extent restrict system flexibility. For these reasons, the classifications of some attributes have been increased to satisfy the overall requirements.

2. Overview of Required Secure SQL Server Components

The Sybase Secure SQL Server provides many features which support both database security and integrity. Users perform queries and updates to the database through any user terminal connected to a host accessing the server via the network. There are two dedicated trusted interface terminals for each server: the System Security Officer Trusted Interface (SSOTI), and the User Trusted Interface (USERTI). The System Security Officer (SSO) controls all security-related operations in the Secure SQL Server via the SSOTI, such as creating user accounts and granting maximum login levels. Only users with SSO privileges may use this terminal. The USERTI is available to all users and provides such options as changing passwords and granting and revoking discretionary access control to databases and tables. Users with System Administrator (SA) privileges, as granted by the SSO, have additional options available to them via the USERTI, including managing system resources. The separation of roles for security and administrative tasks prevents any one user from having complete control over the system.

Many of the Secure SQL Server features are utilized in the implementation of the MLS Design problem. As previously mentioned, the Sybase Secure SQL Server provides mandatory and discretionary access controls within its environment. Thus, the system managers of the various parts of the system may enforce controls on the data that users are able to access or update. Integrity controls in the system reduce the possibility of incorrect data being entered into the system. The Sybase Secure SQL Server ensures both the physical and logical integrity of the data.

2.1. Mandatory Access Control

The Sybase Secure SQL Server provides mandatory access control on a row-level basis. Each record within the system has a security label associated with it, and all mandatory access controls are strictly enforced based on record labels. This model covers all of the records in the database, including metadata in the data dictionary. When a user creates a table in the Secure SQL Server, the system stores the schema information about that table in records in system tables. These records are labeled with the security level of the user. The side effect of this model is that it creates a minimum access level for the table. As a result, users granted discretionary access to a table must be logged in at or above the level at which that table was created in order to know that the table and its attributes exist. If a user does not have discretionary access to a table, then all requests referencing the table result in a 'table not found' message. All tables, including

system tables, are protected from general user access by mandatory and discretionary controls.

User accounts are created and granted a maximum security login level via the SSOTI. This maximum login level for the user is made up of two parts: a hierarchical level from 1 to 16, and a combination of up to 64 non-hierarchical compartments. "System high" is considered login level 16, with compartments 1 through 64. Again, when a login account is created, information is stored as a record in a system table. These user records are labeled system high and contain information about the current state of the account such as username, password, maximum login security level, and roles (i.e., SA or SSO).

The homework problem requires 4 hierarchical levels and 9 compartments, all of which have been assigned equivalent hierarchical and compartmental values in the Sybase Secure SQL Server as listed below. Throughout this paper, the hierarchic level will be referred to by its alphabetic code and compartments by their numeric code:

Hierarchical Levels

Homework Problem	Code	Sybase Secure SQL Server
Unclassified	U	1
Company Private	C	2
Sensitive	S	3
Truly Sensitive	TS	4

Compartments

Homework Problem	Sybase Secure SQL Server
A	1
J	2
K	3
L	4
Q	5
W	6
X	7
Y	8
Z	9

Each record of data within the Sybase Secure SQL Server is assigned a security label corresponding to the login level of the user who created or most recently updated the record. Thus, if a piece of data is to remain classified TS with compartment 1, then users must make sure that they modify that data while logged in at (TS-1). Otherwise, the record will be polyinstantiated, producing two

similar records with different labels.

Users logged in at or above the security level of a table, and having been granted discretionary select access for a table, will be able to select all records in that table labeled at or below their current hierarchical login level and with any subset of compartments that they chose upon logging in. For example, a user logged in at TS-1,2 will be able to see all U, C, and TS data with compartments 1 only, 2 only, both 1 and 2, or no compartments.

2.2 Discretionary Access Control

Discretionary access is granted to users in the Secure SQL Server via the USERTI. Users owning objects, such as databases or tables, may grant other users, or groups of users, access to those objects via this interface. Discretionary access for databases consists of whether or not the user may use the database. Discretionary access for tables consist of any combination of select, insert, update, and/or delete permissions on tables for each individual user. Once a user has access to a table(s), he/she will also be able to execute any of the views or stored procedures that reference the table(s).

2.3 Groups

Related users may be assigned to groups. This feature has many practical uses, such as assigning all members of a department to a single group. Then, if the department is to be granted discretionary access to a table, such as select access to table1, the table owner can quickly give the department access to the table by granting the group select access. All users of a database are members of that database's "public" group.

2.4 Rules

Rules are not part of the Sybase Secure SQL Server Trusted Computing Base (TCB) but can be used as additional integrity controls on the data. Rules specify what data users are or are not allowed to enter in a particular column of a table. They should be created at system low (level U with no compartments) to guarantee access for all fields, although this is not required. Once created, a rule can be "bound" to a field at the creation level of the table containing the field. Because of the way in which rules are created, the same rule may be bound to many different fields. Rules are used to perform integrity checks for values entered into a field and are checked each time an attempt is made to insert or update information in the field bound to the rule.

2.5 Views

Views are not part of the TCB but can be used to aid user access to data, and provide an alternate way of looking at the data in one or more tables. This definition may include reducing the data that the user of the view may see as well as joining tables so that the

user can obtain a wider range of data than is available from just one table. However, since the view is not part of the TCB, it should not be used to restrict access to data. If a view implements a join of two or more tables, each resulting row will have a "high water mark" of the highest hierarchical level and the union of all of the compartments of all of the joined rows. If the user's login level does not dominate the resulting high water mark, then the record will not be displayed to the user. Obviously, this similar phenomenon occurs when a user performs a join of two or more tables without using a view.

Access to a view is not restricted; instead, access to the base tables are controlled. Users with access to all of the objects referenced in a view can successfully use the view. The reasoning behind this feature is that if a user has access to all of the objects mentioned in a view, then they already have the ability to reproduce the results of the view.

2.6 Stored Procedures

Stored procedures are collections of SQL statements and optional control-of-flow statements stored in the database under a user-defined name. The first time a stored procedure is executed, it is compiled and its execution plan is stored, so that subsequent execution will be faster. This allows frequently needed activities to be performed almost instantaneously by the server. Stored procedures were used in our implementation to develop the recurring reports and ad-hoc queries. As in the case for views, there is no discretionary access control mechanism applied to stored procedures. The stored procedure will execute as long as the user has discretionary and mandatory access to the objects referenced to the stored procedure.

3. Data Structures

There are two main entities required by the homework problem: the employee entity and the project entity. In order to satisfy the requirements stated in the homework problem, each entity was broken down to the attribute level.

3.1 Employee Entity

According to the homework problem, the employee entity consists of the employee number, name, home address, job title, salary, profit share, investigative status, credit rating, department, skills, and assigned projects. The field contents are as follows:

- o emp_no - a unique identifier given to each employee upon assignment to the organization
- o emp_name - the name of the employee
- o home_addr - the street address of the employee
- o job_title - the employee's job title
- o salary - the annual salary of the employee in thousands of dollars

- o profit_share - the percentage of salary given to the employee for bonuses and/or share of profit
- o invest_stat - the code representing the current status of ongoing investigations of the employee
- o credit_rating - the credit rating of the employee
- o dept - the department to which the employee is assigned
- o skill - a specific skill in which the employee is qualified
- o project_no a project to which the employee has been assigned

There may be more than one skill or project_no field per emp_no.

The homework problem also requested a field called security_clearance, which contained the maximum hierarchic level and all of the compartments to which that user was to have access. However, the Sybase Secure SQL Server maintains this information automatically, and once login accounts are created for users in the system, their maximum login level is also stored in a system table. Users are allowed to log in at or below their maximum clearance level.

The homework problem required some of the attribute-value pairs to be classified at a different security level than other attribute-value pairs. In the Sybase system, the smallest entity that can be classified is a record. To meet the requirements of attribute level classifications, the employee entity must be divided into several different sub-entities, with the primary key being the employee number, so that the sub-entities may be classified at different levels. By joining tables together, the correct joined classification is automatically generated by the system.

For example, the attributes emp_name and dept are classified at (U), the attribute-value for the employee name by itself is classified (U), and the department attribute-value is classified (S). If this information was stored in one record, the record would necessarily be labeled (S). This would over-classify the employee name attribute-value. In order to allow users to access the employee name data at login level (U), this data must be separated from the department data which is classified (S). Therefore, the employee name must be placed in a table classified (U), while the department data is placed into a separate table and labeled (S). Since the existence of the department attribute is classified (U), the department table was created at (U) so that its attribute name would be available to users logged in at (U).

One unique index was placed on the employee table to prevent multiple emp_no values in the unique emp_no field. Unique indexes in the Secure SQL Server require the sec_label field to be included in the definition of the unique index.

3.2 Project Entity

The project entity is comprised of a project id, project name, project subject, and project client. The description of the

contents of each field are as follows:

- o **proj_id** - the unique unclassified code used to represent each project
- o **proj_name** - the unique code name for a project
- o **proj_subject** - the name of the project which indicates the actual subject of the effort
- o **proj_client** - the name of the organization which is the client of the project.

There may be more than one project performing work for the same client.

Two unique indexes were required to assure the uniqueness of the project id and the project name data. One unique index was placed on **project_id** and one on **project_name**.

The drawings below indicate the relationship between the tables. The bolded field names are fields which occur in more than one table and may be used to perform joins with other tables. The **project_no** field in the **assigned_projects** table can be joined with the **proj_id** field of any of the project tables in order to access information regarding projects to which an employee is assigned.

Employee Entity Table Structure

(U) Employee

sec_label	emp_no	emp_name
U	168	Sprito

(U) Address

sec_label	emp_no	home_addr
U	168	14th St

(U) Job_Title

sec_label	emp_no	job_title
C	168	Tech Staff

(U) Salary

sec_label	emp_no	salary
S	168	80

(U) Profit_Share

sec_label	emp_no	profit
S	168	30

(U) Invest_Status

sec_label	emp_no	invest_stat
C	168	1C

(S-1) Credit_Rating

sec_label	emp_no	credit
TS-1	168	1

(U) Skill

sec_label	emp_no	skill
TS	168	5B

(U) Department

sec_label	emp_no	dept
U	168	Erg
U	354	Exec

(U) Assigned_Projects

sec_label	emp_no	project_no
U	168	3
U	168	1

Project Entity Table Structure

(U) Project_ID

sec_label	proj_id
U	1
U	3

(U) Project_Name

sec_label	proj_id	proj_name
U	1	SOLVER
S	3	ISTAR

(U) Project_Client

sec_label	proj_id	proj_client
U	1	A
U	3	B

(U) Project_Subject

sec_label	proj_id	proj_subject
U	1	CALS Proposal
U	3	Doc. Tracking System
S-4, 6	3	Neural Net Sec. Kernel

4. Sybase Secure SQL Server Implementation

4.1 Mandatory Access Controls

The following charts indicate the mandatory access, creation, and data levels for each table. Data levels for ranges of data must be maintained by the user, unless external code is used. The chart below indicates the mandatory access level/creation level and data entry level. The mandatory access level indicates the minimum login level required to gain access to a table and its attributes. The mandatory access level is the same as the level at which the table was created.

EMPLOYEE ENTITY

Table Name	Mandatory Access/ Creation Level	Data Level
employee	U	U
address	U	U
job_title	U	C
salary	U	U if salary < 25 C if 25 <= salary < 75 S if salary >= 75
profit_share	U	S
invest_status	U	C
credit_rating	S-1	TS-1
department	U	S
skill	U	TS
assigned_projects	U	U

PROJECT ENTITY

Table Name	Mandatory Access/ Creation Level	Data Level
project_id	U	U
project_name	U	SOLVER (C) ALPHA (U) ISTAR (S) EPCOT (S) BARBER (S) GEMINI (S)
project_subject	U	CALS Proposal (C) HQ Relocation (U) Neural Net Security Kernel (S-4,6) Autonomous Land Spy Vehicle (TS-3,7) Pentagon Redesign (TS-3,8) Presidential Inference Controller (TS-3,5,9)

4.2 Discretionary Access Controls

Discretionary access controls were required by the homework problem to maintain a "need to know" policy for employee data. As a result, certain users and groups were allowed certain types of discretionary access to specific tables. The groups execsec, execmgr and secdept were formed to respectively group the executive secretaries, executive managers, and security department. The following table lists a representative sample of the discretionary access control requirements implemented on the Secure SQL Server. Although 'delete' is also a discretionary access that can be granted, it was not mentioned in the homework problem so it is not

depicted in the table below.

EMPLOYEE ENTITY

Table Name	Select	Insert	Update
employee	public	myer	myer
address	public	myer	myer
job_title	public	myer	myer
department	public	myer	myer
assigned_projects	public	myer	myer
salary	execsec	myer	execsec
profit_share	execmgr		
invest_status	execsec	myer	execsec
credit_rating	execmgr		
skill	secdept		
	public	myer	secdept
	public	myer	secdept

PROJECT ENTITY

Table Name	Select	Insert	Update
project_id	public		
project_name	public		
project_subject	public		
project_client	public		

4.3 Integrity Constraints

Many integrity constraints were imposed by the requirements listed in the homework problem. However, none were beyond the scope of the Secure SQL Server's capabilities. To force a field to only contain a value from a distinct subset of possible values, a rule was created and then bound to the column which was supposed to follow the rule. The following commands are examples of the types of rules used to enforce integrity.

Since the employee number field is a positive number, a rule was created to force emp_no to be positive:

```
create rule posnum as @emp_no > 0
```

Then, the rule was bound to the employee table's emp_no field:

```
bind rule posnum to employee.emp_no
```

Another requirement by the homework problem constraining the values that could be contained in the dept table was satisfied by the rule:

```
create rule deptname as @dept in ("EXEC", "PERS", "PROC", "FIN", "SEC", "ENG", "INT", "OPS")
```

bind rule deptname to department.dept

Although only two rules are mentioned in this paper, a variety of different rules were incorporated into the implementation.

5. Design Variations

All but a few of the security and integrity constraints were able to be implemented in our system. In any relational system, there are several alternatives to any implementation. The following variations were used in the Sybase implementation.

5.1 Security Constraints

The homework problem requested that the classification of each user's maximum login level be classified at its hierarchic level, excluding the compartments. The Sybase Secure SQL Server handles all classifications internally. Each user's login account information is labeled system high. Thus, the desired functionality was not implemented. Other portions of the assignment requested that the classification of a record's security level be different than its security level. As mentioned above, the classification of the security level of any record is the classification of the record.

The requirements state that discretionary access control be placed on the address table based on an employee's title and/or department. For example, only managers, personnel clerks, and department secretaries should be granted discretionary access to the address table, since they are the only ones allowed to see information of this kind. Beyond simple access restriction to the table, the homework problem requested that certain users be given access to only portions of the data in the table. There are a variety of approaches to this problem:

- (1) Create a stored procedure based on the current database structures which determines whether or not the user is allowed access to the information. However, this stored procedure would be fairly complex and the users would be forced to use stored procedures when accessing certain attribute-value pairs.
- (2) Add an extra field to the table containing the address information for each employee, indicating the type of person within the company that is allowed to access that person's address (e.g., a value of '1' imply that this person is a member of the "management group" which is allowed to see any other manager's address). A stored procedure could then (a) check the value of the employee's code field and compare it to the code field of the requestor, and (b) verify that the requestor is the secretary of the department to which they belong.

- (3) Assign the address information additional compartments, depending on the characteristics of the employee, to which only acceptable users will be granted access. This also involves granting additional compartments to each user with access.

Any one of these approaches can be used to satisfy the intent of the homework problem.

Another security constraint imposed on the system throughout the homework problem is data aggregation. Two different attributes may result in a higher classification than either of the two individual attribute-values require by themselves. This introduces a popular problem among secure relational databases in which a piece of data selected by itself is not really sensitive, but when selected with another piece of individually non-sensitive data, produces a sensitive result. As an example from the homework problem, consider the instance in which the employee name by itself is classified (U), and the employee's job title is classified (U), but in association with each other they are classified (C). Using two tables with the primary key emp_no to satisfy this requirement, one of the following scenarios results:

- (1) Both attributes are singly classified at (U), causing their association to be classified at (U);
- (2) The employee name is classified (U), and the job title data is classified (C), causing the join to be classified (C) but resulting in the loss of job title data to users logged in at security level (U).

Thus, further controls must be imposed upon the system to satisfy all of the requirements. The complete answer to the problem presents a third solution, which requires a third intermediate table and an extra field:

- (3) A third intermediate table can be created, containing employee numbers and an associated employee code. The job title table then consists of the employee code and the code's job title. Users wishing to associate the employee name with the job title will be forced to go through the intermediate employee code table, which is classified at (C). This method provides complete resolution of the security constraints placed on these two fields.

A pictorial mapping of the third option listed above is depicted below. The field 'high_water_mark' represents the resulting system classification of the joined row:

Option (3)

(U) Employee

sec_label	emp_no	emp_name
U	354	Smith, G

(U) Employee_Code

sec_label	emp_no	emp_code
U	354	1984

(U) Job_Title

sec_label	emp_no	job_title
C	1984	Pres

Resulting Join:

high_water_mark	emp_no	emp_name	job_title
-----	-----	-----	-----
C	354	smithg	pres

The homework problem requires that data security levels be maintained based on the value of an attribute. As depicted in section IV, the salary data security level is dependent on the value of the field. Currently, the users will have to verify that they are logged in at the correct level in order to update the data. Additional code can be implemented to check the user's login level against the value of the data in order to guarantee that inserts and updates take place at the correct login level.

The security constraints listed in the homework problem indicate a need to classify the classification of the security level of an attribute (such as credit_rating) at a different level than that at which the attribute is classified. Our system classifies the classification of an attribute at the same level as its actual security level, which is the level of the table containing the attribute.

A "cover story" implementation is requested by the homework problem, which displays a cover story project subject to users without the proper clearance. Users with the proper clearance will have access to both the "real story" and the "cover story". The requirements state that the association of the proj_subject attribute-value with the proj_name attribute-value pair is classified at varying levels, depending on the pair. This task was accomplished by placing the project name data in one table with its respective security classifications, and then placing the project subject data into another table with its respective classifications. The high water mark of the association also happens to be the security level of the project subject. However, the system requirements call for implementing a cover story for the above-mentioned association if a user is not logged in at the minimum required level to see the information. This was implemented by classifying the cover story for each project at system low, and classifying the actual proj_subject at its required

level. Thus, users selecting the proj_subject at an incorrect level will obtain the cover story. Users logged in at the correct level will see both the real proj_subject information as well as the cover story proj_subject.

The homework problem requires some integrity constraints to have specific classifications. Integrity constraints are normally created at system low and bound to the necessary table at the creation level of the table. However, if necessary, a rule can be created at the creation level of a table in order to have it classified at the same level as a table's attribute. Thus, integrity constraints on an attribute may be classified at or below the level of an attribute (which is determined by its containing table) but not above the level of the attribute. As an example, the integrity constraint on the association of the proj_subject and proj_name data is classified at a higher level (TS-2,3,4,5,6,7,8,9) than any of the data contained in the two fields (see section 4.).

5.2 Integrity Constraints

The MLS Design project required that the system generate unique random numeric codes for employee number values. The "uniqueness" of the value was implemented by imposing a unique index on the employee table. The "randomness" of the value can be done by external software, but was not implemented.

The homework problem required that integrity checks be performed on fields allowing combinations of values and presented the data as a list of values separated by commas in one field. In order to perform integrity checks on data that may contain one or more values, the data separated by commas (as in the skill table, "5E,D7,8F") was split into a separate table, with multiple entries allowed per primary key. As an example, a rule was created for the skill field in the skill table to restrict the valid values for skill codes as follows:

```
create rule skills as @skill in ("3A", "5B", "D7", "8F", "C9")
bind rule skills to skill.skill
```

As a result, each entered skill value can be checked against the rule bound to the skill table. This occurs for both the skill and assigned projects employee data. The first drawing below indicates the resulting implementation.

sec_label	emp_no	skill
TS	753	C9
TS	753	8F
TS	753	5B

vs

sec_label	emp_no	skill
TS	753	C9, 8F, 5B

The integrity constraint on the proj_subject field is that it is based on a mapping from the proj_name to the proj_subject field.

This can be accomplished by verifying that only authorized personnel can update that data. The table owner of this table can prevent all but authorized personnel from being granted access to this table, thus restricting the improper update of the data in the table. In this manner, each skill entered into its own field can be checked against the valid values for that field.

6. Conclusions

This paper presented TRW's solution to the MLS Design homework problem written by Gary Smith. The Sybase Secure SQL Server provides the means to successfully implement the security and integrity requirements as stated in the MLS Design homework problem. This paper provides examples of variations from the requirements in order to demonstrate the flexibility of our system to adapt to difficult, real-world system constraints.

Various components of the Sybase Secure SQL Server were discussed, including the methods by which mandatory and discretionary access controls are implemented in the system. The concepts of groups, rules, views, and stored procedures were also discussed.

SCTC Technical Note
Secure Database Homework Problem #2
2nd RADC Secure Database Conference

LOCK Data Views

Prepared by
Honeywell Inc.
Secure Computing Technology Center
St. Anthony, Minnesota, 55418

May 15, 1989

Prepared For:
Rome Air Development Center
Workshop on Database Security
Franconia, New Hampshire
May 16..18, 1989

PREFACE

This report entitled "Homework #2" describes a sample database, as it might be designed, if defined for Lock Data Views (LDV). It gives the logical and physical schema and reasoning behind the LDV style. It shows sample queries from an LDV prototype against a sample database. This sample database and associated policies and requirements were provided by Gary W. Smith, for use at the 2nd RADC RADC Workshop on Database Security.

The work was supported by Honeywell from IR&D funds, and is not a part of the Secure Distributed Data Views (SDDV) Contract (contract no: F30602-86-C-0003), nor of any other Federal Government Contract. It is related to work done on the SDDV and LOCK contracts by the Secure Computing Technology Center (SCTC), Honeywell Inc.

This report describes activity performed by the LDV team on the Homework Problem during the period 1 February 1989 through 15 May 1989. The SCTC personnel were Paul Stachour and Dan Thomsen

CHAPTER 1

Introduction

This report describes the design of a sample database for a Multilevel Secure relational Database Management System (MLS/RDBMS), LOCK Data Views (LDV), being designed to run on the Honeywell LOGical Coprocessor Kernel (LOCK) Trusted Computing Base (TCB) [HONE87]. This chapter presents an overview of the LDV design of the Gary W. Smith Homework Problem, and the LDV approach towards solution.

1.1. Problem Statement

Within the Department of Defense (DoD), the number of computerized databases containing classified or otherwise sensitive data is increasing rapidly. Access to these databases must be restricted and controlled to limit the unauthorized disclosure, or malicious modification, of data contained in them. Present DBMSs do not provide adequate mechanisms to support such control. Penetration studies have clearly shown that the mechanisms provided even by "security enhanced" database systems can be bypassed, often due to fundamental flaws in the systems which host the DBMS. This has led to a reliance on a number of techniques for isolating sensitive database information. These include physical protection, "system high" operation, and use of manual techniques for data sharing. These actions are very costly and detrimental to operational utility and flexibility.

1.2. Why a Sample

It is easy to write a paper design to solve almost any problem. The "proof" of the design is whether the design is capable of solving real problems, or whether it is merely a paper tiger. In this report, we present the LDV design for a database that is held to be representative of the kinds of databases that exist in the secure database user community. LDV should be measured on how well it can do realistic security requirements and associated database schema, such as this homework problem. In the final analysis, LDV can only be measured by how a prototype or real implementation meets the needs of both database programmers and end users.

1.3. Report Outline

This report is organized as follows. Chapter 2 presents Task 1: Database Design. Chapter 3 presents Task 2: Database Schema. Chapter 4 presents Task 3: Database Population. Chapter 5 presents "Task 4: Database Queries. Chapter 6 summarizes the LDV work and this example.

CHAPTER 2

Task 1: Database Design

Designing a database depends not only on the user, but on the underlying system capabilities. In this database design, we use the following two items from LOCK.

```
/*
 * The "Strings" that are the classification levels are NOT defined
 * by the DBMS, but are reflected upwards to it by the underlying TCB,
 * which will translate them to internal level-ids where appropriate.
 * This includes determination of the dominates relation.
 */

/*
 * The "Names" appearing as authorization identifiers are those of
 * "Groups" to which individuals may belong. Assignment of individuals
 * to a group is a LOCK TCB function. Authorization of groups to access
 * relations is a LDV DB function. A user belongs to one default
 * group. There is a LOCK command (change_group) to allow a user
 * to change his group (if authorized), and thus his "role" as seen
 * by the LDV database sub-system.
 */
```

The approach taken by the LDV team is to define the base relations of the database with an understanding of the security requirements, and to provide a view on top of those base relations that reflects the visibility desired by the external customer. Accordingly, we define the view (as seen by the external user), the relations (as seen by the DBA), the security constraints (as seen by the DBSSO), and finally the security constraints on the security constraints (as seen by the DBSSO). As LDV was not designed to handle integrity constraints, [1] they are not defined in this document except as they fall out "naturally" from the LDV style. [2]

Wherever tables belonging to the LDV database system are shown, they are identified by the initial string "DB\$", such as DB\$RELATIONS.

2.1. View Definitions

```

/*
*****
* Views Definitions
* May be done by DBA (Database Administrator)
*/

create view EMPLOYEE (
  (
    Employee-Num
    Employee-Name
    Home-Address
    Job-Title
    Department
    Assigned-Projects
    Salary
    Profit-Share
    Investigative-Status
    Skills
    Security-Clearance
  )
as select
  EMPLOYEE-BASE.Employee-Num
  Employee-Name
  EMPLOYEE-ADDRESS.Home-Address
  EMPLOYEE-TITLE.Job-Title
  EMPLOYEE-BASE.Department
  Salary
  EMPLOYEE-PROFIT.Profit-Share
  EMPLOYEE-INVESTIGATIVE.Investigative-Status
  Security-Clearance
from
  EMPLOYEE-BASE,
  EMPLOYEE-ADDRESS,
  EMPLOYEE-TITLE,
  EMPLOYEE-PROFIT,
  EMPLOYEE-INVESTIGATIVE,
  EMPLOYEE-CREDIT
where
  EMPLOYEE-BASE.Employee-Num = EMPLOYEE-ADDRESS.Employee-Num and
  EMPLOYEE-BASE.Employee-Num = EMPLOYEE-TITLE.Employee-Num and
  EMPLOYEE-BASE.Employee-Num = EMPLOYEE-ADDRESS.Employee-Num and
  EMPLOYEE-BASE.Employee-Num = EMPLOYEE-PROFIT.Employee-Num and
  EMPLOYEE-BASE.Employee-Num = EMPLOYEE-INVESTIGATIVE.Employee-Num and
  EMPLOYEE-BASE.Employee-Num = EMPLOYEE-CREDIT.Employee-Num

```

2.2. Record Definitions


```

/*
*****
* Record Definitions
* May be done by DBA (Database Administrator)
*/

```

```

/* Employee Record */
create table EMPLOYEE-BASE
    maximum level TS
    default level U
(
    Employee-Num          decimal(3)
                        maximum level U
                        default level U
                        unique

    Employee-Name        character(12)
                        default level U
    /* DAC restriction: Home-Address */
    /* DAC restriction: Job-Title */
    Department           character(3)
                        default level S
    /* Multi-Value Column: Assigned-Projects */
    Salary               decimal(4)
                        default level U
    /* DAC restriction: Profit-Share */
    /* MAC restriction: Investigative-Status */
    /* MAC restriction:      /* Multi-Value Column: Skills */
    Security-Clearance   character(10)
)
    primary key Employee-Num

```

```

/* Project Record */
create table PROJECT
    maximum level TS
    default level U
(
    Project-Id           decimal(2)
                        default level U

    Project-Name        character(10)
                        default level U

    Project-Subject     character(30)
                        default level U
                        maximum level TS-KQZ

    Project-Client      character(3)
                        default level S
                        maximum level S
)
    primary key Project-Id

```

```
/* Multi-Attribute Project within Employee */  
create table EMPLOYEE-PROJECT
```

```
        maximum level TS  
        default level U  
  
(  
  Employee-Num          decimal(3)  
                        default level U  
  Assigned-Project      decimal(2)  
                        maximum level U  
                        minimum level U  
)  
  primary key Employee-Num
```

```
/* Multi-Attribute Skill within Employee */  
create table EMPLOYEE-SKILL
```

```
        maximum level TS  
        default level U  
  
(  
  Employee-Num          decimal(3)  
                        default level U  
  Skill-Code            character(2)  
                        default level S  
)  
  primary key Employee-Num
```

```
/* DAC restriction upon Home-Address within Employee */  
create table EMPLOYEE-ADDRESS
```

```
        maximum level U  
        default level U  
  
(  
  Employee-Num          decimal(3)  
                        default level U  
                        unique  
  Home-Address          character(12)  
                        default level U  
)  
  primary key Employee-Num
```

```
/* MAC restriction upon Job-Title within Employee */
create table EMPLOYEE-TITLE
```

```
    maximum level C
    default level U
(
  Employee-Num          decimal(3)
                        default level U
                        unique
  Job-Title             character(12)
)
primary key Employee-Num
```

```
/* DAC restriction upon Profit-Share within Employee */
create table EMPLOYEE-PROFIT
```

```
    maximum level U
    default level U
(
  Employee-Num          decimal(3)
                        default level U
                        unique
  Profit-Share         decimal(3)
                        default level U
)
primary key Employee-Num
```

```
/* MAC restriction upon Investigate-Status within Employee */
create table EMPLOYEE-INVESTIGATIVE
```

```
    maximum level C
    default level U
(
  Employee-Num          decimal(3)
                        default level U
                        unique
  Investigative-Status character(2)
                        default level C
)
primary key Employee-Num
```

```

/* MAC restriction upon Investigate-Status within Employee */
create table EMPLOYEE-CREDIT
        maximum level S-A
        default level U
(
    Employee-Num          decimal(3)
                        default level U
                        unique
    Credit-Rating         decimal(1)
                        default level S-A
)
    primary key Employee-Num

```

2.3. Security and Integrity Assertions

```

/*
*****
* DAC Security Assertions
* Must be done by DBSSO (Database System Security Officer)
*/

```

2.3.1. Row Insertions: Record Creation

```

grant insert
    on EMPLOYEE
    to PersonnelAnalyst

```

```

grant insert
    on EMPLOYEE
    to PersonnelClerk

```

```

grant insert
    on PROJECT
    to DBSO

```

```

grant insert
    on PROJECT
    to AssistantDBSO

```

2.3.2. Update Integrity

2.3.2.1. Project Update

```
grant update
  on PROJECT
  to Secretary
where
  select Project-Num from Project
  =
  select Project-Num from EMPLOYEE-BASE
  where Project-Num = [user project-num]
```

2.3.2.2. Employee Selection

```
grant select ( Employee-Num, Home-Address )
  on EMPLOYEE-ADDRESS
  to ExecutiveDepartment

grant select ( Employee-Num, Home-Address )
  on EMPLOYEE-ADDRESS
  to PersonnelAnalyst

grant select ( Employee-Num, Home-Address )
  on EMPLOYEE-ADDRESS
  to Manager

grant select ( Employee-Num, Home-Address )
  on EMPLOYEE-ADDRESS
  to Secretary
```

2.3.2.3. Employee Update

```
grant update ( Investigative-Status, Credit-Rating, Security-Clearance)
  on EMPLOYEE
  to SecurityOffice

grant update
  on EMPLOYEE-SKILL
  to SecurityOffice

grant update ( Salary, Profit-Share )
  on EMPLOYEE
  to ExecutiveSecretary
```

```
grant update ( Employee-Num,
               Employee-Name,
               Home-Address,
               Job-Title,
               Assigned-Projects )
  on EMPLOYEE
  to PersonnelAnalyst
```

```
grant update ( Employee-Num,
               Employee-Name,
               Home-Address,
               Job-Title,
               Assigned-Projects )
  on EMPLOYEE
  to PersonnelClerk
  where ...
```

2.3.3. Security Assertions

```
/*
*****
* MAC Security Assertions
* Must be done by DBSSO (Database System Security Officer)
*/
```

DAC Security Assertions not provided in this prototype.[3]

2.3.3.1. SC#1

This constraint is met by the fact that the field Employee-Num in the record EMPLOYEE-BASE has an attribute that specifies that the maximum-level is U and the default-level is U. Since it is used as a join-key, thus any operation upon it through the EMPLOYEE table must also have the (U,U) attributes.

[1] Our research shows that security and integrity constraints provide pulls in opposite directions. Enforcing one often means giving up on the other. An example is an airplane flight with a SECRET cargo. Allowing a U user to add cargo will overload the plane; denying the U user will provide strong-suspicion of SECRET cargo.

[2] Some of the integrity constraints are much better given as type-constraints or mapping rules in programming languages than as values along with object within a database. One way in which this can be done is defined in [AdaSQL].

2.3.3.2. SC#2

This constraint is met by the fact that the field Home-Address in the record EMPLOYEE_ADDRESS has an attribute that specifies that the maximum-level is U and the default-level is U.

2.3.3.3. SC#3

This constraint is met by the fact that the field Home-Address in the record EMPLOYEE_ADDRESS has an attribute that specifies that the maximum-level is U and the default-level is U.

2.3.3.4. SC#4

LDV does not handle context based on DAC policies. It is felt that this should be a combination of OS/TCB and DBMS enforcement.

2.3.3.5. SC#5

```
assert security SC5a to
  select Employee-Num,
         Job-Title
  from EMPLOYEE-TITLE
  level C
```

```
/* Note: subsumed in above due to database design */
assert security SC5b to
  select Employee-Name
         from EMPLOYEE-BASE
         Job-Title
         from EMPLOYEE-TITLE
  level C
```

[3]

There are many ways to do DAC. The general consensus (from our observations of existing systems) is that they are implemented as explicit code in the DBMS. We would prefer to do DAC via ACLS on the files, and have enforcement done by the underlying TCB. Whether this can be done under the letter-of-the-law according to the criteria is a question under discussion.

2.3.3.6. SC#6

```
assert security SC6a to
  select Salary,
         Employee-Num
  from EMPLOYEE-BASE
   where Salary >= 25
   and   Salary < 75
  level C
```

```
assert security SC6b to
  select Salary,
         Employee-Num
  from EMPLOYEE-BASE
   where Salary >= 75
  level S
```

Note that Employee-Name "falls out" as well, due to the way the DBA designed the database, and so need not be explicitly specified.

2.3.3.7. SC#7

```
assert security SC7 to
  select Employee-Num,
         Profit-Share
  from EMPLOYEE-PROFIT
  level S
```

2.3.3.8. SC#8

```
assert security SC8a to
  select Employee-Num,
         Investigate-Status
  from EMPLOYEE-INVESTIGATIVE
  level C
/* security SC8b, that of Employee-Name, is automatic by
   this design, since Employee-Num is needed to do the join.
*/
```

2.3.3.9. SC#9

```
assert security SC9a to
  select *
  from DB$RELATIONS
   where Table_Name = "EMPLOYEE-CREDIT"
  level S-A
```



```
assert security SC9b to
  select *
    from DB$ATTRIBUTES
     where Table_Name = "EMPLOYEE-CREDIT"
  level S-A
```

```
assert security SC9c to
  select *
    from DB$PRIMARY_KEYS
     where Table_Name = "EMPLOYEE-CREDIT"
  level S-A
```

2.3.3.10. SC#10

```
assert security SC10 to
  select Employee-Num,
         Credit-Rating
    from EMPLOYEE-CREDIT
  level TS-A
```

2.3.3.11. SC#11

```
assert security SC11 to
  select *
    from DB$CLASSIFICATION_CONSTRAINTS
     where Constraint_Name = "SC9a"
     or   Constraint_Name = "SC9b"
     or   Constraint_Name = "SC9c"
     or   Constraint_Name = "SC10"
     or   Constraint_Name = "SC11"
  level TS-A
```

2.3.3.12. SC#12

SC#12 is inherent from the database design, where the default level for all the values of the attribute department is set to "S".

2.3.3.13. SC#13

SC#13 is inherent from the database design, where the default level for all the values of the attribute Skill-Code is set to "S".

2.3.3.14. SC#14

```
assert security SC14 to
  select Employee-Num,
         Skill-Code
  from EMPLOYEE-SKILL
  level TS-A
```

2.3.3.15. SC#15

```
assert security SC15a to
  select Employee-Num,
         Skill-Code
  from EMPLOYEE-SKILL
  Employee-Num,
  Assigned Project
  from EMPLOYEE-PROJECT
  Project-Id,
  Project-Name
  from PROJECT
  level TS
```

```
assert security SC15b to
  select Employee-Num,
         Skill-Code
  from EMPLOYEE-SKILL
  Employee-Num,
  Assigned Project
  from EMPLOYEE-PROJECT
  Project-Id,
  Project-Name
  from PROJECT
  level level(PROJECT.Project-Name)
```

This is the kind of context constraint where LDV shines. Given the way LDV is built, the constraint will be enforced no matter what the time-difference between the various portions of the retrieval.

2.3.3.16. SC#16

SC#16 is inherent from the database design, where the default level for all the values of the attribute Assigned-Projects is set to "U".

2.3.3.17. SC#17

```
assert security SC17 to
  select Assigned-Project
         from EMPLOYEE-PROJECT
         Project-Name
         from PROJECT
         level level(PROJECT.Project-Name)
```

SC#17 falls out since by our schema design, the level of the Project-Name is stored in such a way that the desired level is automatically enforced.

2.3.3.18. SC#18

```
assert security SC18 to
  select Security-Clearance
         from EMPLOYEE-BASE
         level level(EMPLOYEE-BASE.Security-Clearance)
```

2.3.3.19. SC#19

```
assert security SC19 to
  select Project-Id, Project-Name
         from PROJECT
         where Project-Id >=3
         and Project-Id <= 6
         level S
```

2.3.3.20. SC#20

```
assert security SC20a to
  select Project-Name
         from PROJECT
         where Project-Name = "SOLVER"
         level C
```

```
assert security SC20b to
  select Project-Name
         from PROJECT
         where Project-Name = "ALPHA"
         level U
```

```
assert security SC20c to
  select Project-Name
    from PROJECT
    where Project-Name = "ISTAR"
    or Project-Name = "EPCOT"
    or Project-Name = "BARBER"
    or Project-Name = "GEMINI"
  level S
```

2.3.3.21. SC#21

SC#21 and SC#22 are a cover-story vs real-value setup.

```
assert security SC21a to
  select Project-Name,
    Project-Subject
    from PROJECT
    where Project-Name = "SOLVER"
  level C
```

```
assert security SC21b to
  select Project-Name,
    Project-Subject
    from PROJECT
    where Project-Name = "ALPHA"
  level U
```

```
assert security SC21c to
  select Project-Name,
    Project-Subject
    from PROJECT
    where Project-Name = "ISTAR"
  level S-LW
```

```
assert security SC21d to
  select Project-Name,
    Project-Subject
    from PROJECT
    where Project-Name = "EPCOT"
  level TS-JX
```

```
assert security SC21e to
  select Project-Name,
    Project-Subject
    from PROJECT
    where Project-Name = "BARBER"
  level TS-KY
```

```
assert security SC21f to
  select Project-Name,
         Project-Subject
  from PROJECT
  where Project-Name = "GEMINI"
  level TS-KQZ
```

In reality, we would set the data at their actual levels, and create an unclassified cover story, as we show in the prototype.

2.3.3.22. SC#22

This falls out naturally from the LDV design since the user will not know that there is a cover story until they are at the level where they can see the real story.

2.3.3.23. SC#23

```
assert security SC23 to
  select *
  from DB$INTEGRITY_CONSTRAINTS
  where Constraint_Name = "IC15"
  level TS-JKLQWXYZ
```

2.3.3.24. SC#24

```
assert security SC24 to
  select Project-Name,
         Project-Client
  from PROJECT
  level level(PROJECT.Project-Name)
```

This falls out naturally from the LDV design since the Project-Name is stored at the level of Project-Name, you will not be able to combine it with anything unless you are at that level.

2.3.3.25. SC#25

SC25 is done by classifying all values of the Project-Client attribute at "S" or higher. This is done with the "default level S" classification at the definition of the Project-Client attribute.

2.3.4. Integrity Assertions

The LDV SQL Language definition includes limited integrity enforcement. This is not in the prototype.

2.3.4.1. IC#1

Unique handled by database design. Randomly generated must be done by application.

2.3.4.2. IC#2

Allowed values handled by database design.

2.3.4.3. IC#3

Alphanumeric not enforced by LDV. This is a language typedef question not handled by SQL.

2.3.4.4. IC#4

Enumerations not enforced by LDV. This is a language typedef question not handled by SQL.

```
assert integrity IC4 IMMEDIATE to
EMPLOYEE-TITLE:
```

```
  not (
    (EMPLOYEE-TITLE.Job-Title = "Pres" ) and
    (EMPLOYEE-TITLE.Job-Title = "VP-HQ" ) and
    (EMPLOYEE-TITLE.Job-Title = "VP-PM" ) and
    (EMPLOYEE-TITLE.Job-Title = "ExecSecy" ) and
    (EMPLOYEE-TITLE.Job-Title = "DeptMgr" ) and
    (EMPLOYEE-TITLE.Job-Title = "TechStaff" ) and
    (EMPLOYEE-TITLE.Job-Title = "DBSO" ) and
    (EMPLOYEE-TITLE.Job-Title = "AstDBSO" ) and
    (EMPLOYEE-TITLE.Job-Title = "SecClk" ) and
    (EMPLOYEE-TITLE.Job-Title = "AdmAst" ) and
    (EMPLOYEE-TITLE.Job-Title = "PerAnal" ) and
    (EMPLOYEE-TITLE.Job-Title = "PersClk" ) and
    (EMPLOYEE-TITLE.Job-Title = "FinAnal" ) and
    (EMPLOYEE-TITLE.Job-Title = "FinClk" ) and
    (EMPLOYEE-TITLE.Job-Title = "ProcAnal" ) and
    (EMPLOYEE-TITLE.Job-Title = "ProcClk" ) and
    (EMPLOYEE-TITLE.Job-Title = "MgtAnal" ) and
    (EMPLOYEE-TITLE.Job-Title = "Adm Clk" ) and
    (EMPLOYEE-TITLE.Job-Title = "Secy" )
  )
on
  insert to EMPLOYEE-TITLE,
  update to EMPLOYEE-TITLE
violation action abort
```

2.3.4.5. IC#5

Numeric values handled by database design. Allowed numeric values not enforced by LDV. This is a language typedef question not handled by SQL.

```
assert integrity IC5 IMMEDIATE to
  EMPLOYEE-BASE
  not (
    (EMPLOYEE-BASE.Salary > 0 )
  )
on
  insert to EMPLOYEE-BASE,
  update to EMPLOYEE-BASE
violation action abort
```

2.3.4.6. IC#6

Numeric values handled by database design.

2.3.4.7. IC#7

Alphanumeric not enforced by LDV. This is a language typedef question not handled by SQL.

2.3.4.8. IC#8

Numeric values handled by database design. Allowed numeric values not enforced by LDV. Allowed numeric values enforced by database typedef.

2.3.4.9. IC#9

Enumerations not enforced by LDV. This is a language
typedef question not handled by SQL.

```

assert integrity IC9 IMMEDIATE to
  EMPLOYEE-BASE:
    not (
      (EMPLOYEE-BASE.Department = "Exec" ) and
      (EMPLOYEE-BASE.Department = "Pers" ) and
      (EMPLOYEE-BASE.Department = "Proc" ) and
      (EMPLOYEE-BASE.Department = "Fin" ) and
      (EMPLOYEE-BASE.Department = "Sec" ) and
      (EMPLOYEE-BASE.Department = "Eng" ) and
      (EMPLOYEE-BASE.Department = "Int" ) and
      (EMPLOYEE-BASE.Department = "Ops" )
    )
  on
    insert to EMPLOYEE-BASE,
    update to EMPLOYEE-BASE
  violation action abort

```

2.3.4.10. IC#10

Enumerations not enforced by LDV.
This is a language typedef question
not handled by SQL.

```

assert integrity IC10 IMMEDIATE to
  EMPLOYEE-SKILL:
    not (
      (EMPLOYEE-SKILL.Skill_Code = "3A" ) and
      (EMPLOYEE-SKILL.Skill_Code = "5B" ) and
      (EMPLOYEE-SKILL.Skill_Code = "D7" ) and
      (EMPLOYEE-SKILL.Skill_Code = "8F" ) and
      (EMPLOYEE-SKILL.Skill_Code = "C9" )
    )
  on
    insert to EMPLOYEE-SKILL,
    update to EMPLOYEE-SKILL
  violation action abort

```

2.3.4.11. IC#11

Numeric values handled by database design. Allowed numeric values not enforced by LDV. This is a language typedef question not handled by SQL.

```
assert integrity IC11 IMMEDIATE to
  EMPLOYEE-PROJECT
  not (
    (EMPLOYEE-PROJECT.Assigned-Project >= 1 ) and
    (EMPLOYEE-PROJECT.Assigned-Project <= 6 )
  )
on
  insert to EMPLOYEE-BASE,
  update to EMPLOYEE-BASE
violation action abort
```

2.3.4.12. IC#12

Enumerations not enforced by LDV. This is a language typedef question not handled by SQL.

Application (at appropriate security level) may validate by call to an appropriate OS subroutine, such as the LOCK create_object.

2.3.4.13. IC#13

Numeric values handled by database design. Allowed numeric values not enforced by LDV. This is a language typedef question not handled by SQL.

```
assert integrity IC13 IMMEDIATE to
  EMPLOYEE-PROJECT
  not (
    (EMPLOYEE-PROJECT.Assigned-Project >= 1 ) and
    (EMPLOYEE-PROJECT.Assigned-Project <= 6 )
  )
on
  insert to EMPLOYEE-BASE,
  update to EMPLOYEE-BASE
violation action abort
```

Note that this is identical to IC#11, and the one-write rule tells us that this should be a language typedef and not an database integrity constraint.

2.3.4.14. IC#14

Enumerations not enforced by LDV. This is a language typedef question not handled by SQL.

```
assert integrity IC14 IMMEDIATE to
PROJECT:
  not (
    (PROJECT.Project-Name = "SOLVER" )
and    (PROJECT.Project-Name = "ALPHA" ) and
    (PROJECT.Project-Name = "ISTAR" ) and
    (PROJECT.Project-Name = "EPCOT" ) and
    (PROJECT.Project-Name = "BARBER" ) and
    (PROJECT.Project-Name = "GEMINI" )
  )
on
  insert to PROJECT,          update to PROJECT
violation action abort
```

2.3.4.15. IC#15

Mappings not enforced by LDV. This is a language typedef question not handled by SQL.

This item was ambiguously worded in the homework. It has been done here in terms of both cover and real names. Since the constraint itself is classified, this is not a leak.

```
assert integrity IC15 IMMEDIATE to
PROJECT:
  not (
    (PROJECT.Project-Name = "SOLVER" and
    PROJECT.Project-Subject = "CALs Proposal" ) and
    (PROJECT.Project-Name = "ALPHA" and
    PROJECT.Project-Subject = "HQ Relocation" ) and
    (PROJECT.Project-Name = "ISTAR" and
    PROJECT.Project-Subject = "Document Tracking System" ) and
    (PROJECT.Project-Name = "EPCOT" and
    PROJECT.Project-Subject = "Personnel System Redesign" ) and
    (PROJECT.Project-Name = "BARBER" and
    PROJECT.Project-Subject = "Inventory Control System" ) and
    (PROJECT.Project-Name = "GEMINI" and
    PROJECT.Project-Subject = "Contracts Management System" ) and
    (PROJECT.Project-Name = "ISTAR" and
    PROJECT.Project-Subject = "Neural Net Security Kernel" ) and
    (PROJECT.Project-Name = "EPCOT" and
    PROJECT.Project-Subject = "Autonomous Land Spy Vehicle" ) and
    (PROJECT.Project-Name = "BARBER" and
    PROJECT.Project-Subject = "Pentagon Redesign" ) and
    (PROJECT.Project-Name = "GEMINI" and
    PROJECT.Project-Subject = "Presidential Inference Controller
  )
on
  insert to PROJECT,
  update to PROJECT
violation action abort
```

2.3.4.16. IC#16

Enumerations not enforced by LDV. This is a language
typedef question not handled by SQL.

```
assert integrity IC16 IMMEDIATE to
PROJECT:
  not (
    (PROJECT.Project-Client = "A" ) and
    (PROJECT.Project-Client = "B" ) and
    (PROJECT.Project-Client = "C" ) and
    (PROJECT.Project-Client = "D" ) and
  )
on
  insert to PROJECT,
  update to PROJECT
violation action abort
```

CHAPTER 3

Task 2: Database Schema

3.1. Schema for a MultiLevel Secure Database

LDV works by splitting the logical view of the employee-base relation [Figure 1] into several files [Figure 2]. Each time a user accesses one of the files while doing a query a tuple is added to the history relation. Each tuple contains the user's name, level, and the file he accessed. The history relation is something that was invented for the prototype. While it seems to be a viable solution, in an actual implementation we plan to explore other possibilities for storing the history information.

A brief explanation of the figures. Each box signifies a file. The file name appears above the box. The file is stored at the level indicated to the left of the box. The context constraint level for each file is expressed in round parenthesis after the file name. SID stands for a system generated ID. SIDS are used when the normal key can not be stored in the same file due to context constraints. The context constraints for the files in the relation are written separately below the boxes.

3.2. Schema Comments

In preparation of the data for project-subject and the associated cover-story, for the cover story it is unclear what level to make the real subject attributes. We put each project-subject in a file at the level it should be classified at, and the cover story at Unclassified. For the CALS proposal subject we also created a cover story at Unclassified even though none was given. The cover story is CALS - COVER STORY. In LDV it will also be possible for the user to specify that the security level of the tuple be shown, so that he can distinguish between the cover stories and the real stories. However, in the Prototype we do not have this feature. To simulate this feature we have added the level of the project-subject to the character string of the actual project-subject.

In LDV, we currently cannot provide a cover-story in the case where both the cover and the real item are at the same security level. We are not sure that such a feature would be needed or useful.

TS

Employee-base U

<u>Employee-Num</u> U	Employee-Name	Department	Salary	Security-Clearance
U	U	S	U	

TS

Project U

<u>Project-ID</u>	Project-Name	Project-Subject	Project-Client
U	U		S

TS

Employee-project U

<u>Employee-Num</u>	<u>Project-ID</u>
U	U

TS

Employee-skill U

<u>Employee-Num</u>	Skill-code
U	S

U

Employee-address U

<u>Employee-Num</u>	Home-address
U	U

C

Employee-title U

<u>Employee-Num</u>	Job title
U	

U

Employee-profit U

<u>Employee-Num</u>	Profit-share
U	U

C

Employee-investigate U

<u>Employee-Num</u>	Investigation-status
U	C

S-A

Employee-credit U

<u>Employee-Num</u>	Credit-rating
U	S-A

Physical Mapping of Employee Relation

Employee-base

Employee-Clearance-TS

Employee-SID	Employee-Clearance	Employee-Compartment
--------------	--------------------	----------------------

TS

Employee-Salary-High (S)

Employee-SID	Employee-Salary
--------------	-----------------

U

Employee-Clearance-S

Employee-SID	Employee-Clearance	Employee-Compartment
--------------	--------------------	----------------------

S

Employee-Salary-Mid (C)

Employee-SID	Employee-Salary
--------------	-----------------

U

Employee-Clearance-C

Employee-SID	Employee-Clearance	Employee-Compartment
--------------	--------------------	----------------------

C

Employee-Dept

Employee-SID	Employee-Dept
--------------	---------------

S

Employee-Salary-Low (U)

Employee-SID	Employee-Salary
--------------	-----------------

U

Employee-Clearance-U

Employee-SID	Employee-Clearance	Employee-Compartment
--------------	--------------------	----------------------

U

Employee-Num-Name

Employee-SID	Employee-Num	Employee-Name
--------------	--------------	---------------

U

Context Constraints

emp-num-name emp-salary-mid Confidential

emp-num-name emp-salary-mid Secret

Physical Mapping of Project Relation

Project

TS-JX

Project-SID	Project-Subject
-------------	-----------------

Project-Subject-TS-JX

TS-KQZ

Project-SID	Project-Subject
-------------	-----------------

Project-Subject-TS-KQZ

TS-KY

Project-SID	Project-Subject
-------------	-----------------

Project-Subject-TS-KY

S

Project-SID	Project-name
-------------	--------------

Project-name-S

C

Project-SID	Project-name
-------------	--------------

Project-name-C

S

Project-SID	Project-Client
-------------	----------------

Project-Client

U

Project-SID	Project-ID
-------------	------------

Project-ID

S

Project-SID	Project-Subject
-------------	-----------------

Project-Subject-S-LW

C

Project-SID	Project-Subject
-------------	-----------------

Project-Subject-C

U

Project-SID	Project-Subject
-------------	-----------------

Project-Subject-U

CHAPTER 4

Task 3: Database Population

4.1. Selection of Database System

Before one can populate a database, unless one is just using something such as flat-files and searching programs such as "awk", one needs to have a database system to use. For this work, we chose to build upon a database system called INQUIRE[HELD88], developed at the University of Minnesota for a class in designing databases. INQUIRE is written in common-lisp, we use a version under KCL on sun workstations. INQUIRE provides relational operators, an interface to file-storage, and the ability to store and process relations as lisp-lists.

4.2. Preparing the Database

After doing the logical and physical database design just presented, the relations, files, and corresponding linkages were hand-crafted into INQUIRE. If this had been a fully implemented database rather than a prototype, the database (LDV) would have done the decomposing from conceptual to internal (physical) schema.

4.3. Preparing the Inquiry System

A set of lisp-functions were then written to act on top of the INQUIRE basic retrieval and relational operators. These provided for storing and retrieving data into/from INQUIRE.

4.4. Populating the Database

The data provided by the homework problem were then hand-entered into INQUIRE using the facilities built. No problems were encountered during entry, but we did note that there were no entries corresponding to the "always unclassified" salary, i.e., that less-than 25K. We also noted that there were duplicates in the employee-number field. These were for employees in different compartments at the same level. We suspect this was to check for the conflict between uniqueness and level-separation, forcing polyinstantiation.[1]

[1] Since our prototype database was done with only 4 levels, we did not test this case, and made the numbers unique.

CHAPTER 5

Task 4: Database Queries

5.1. The LDV Prototype

The LDV prototype provides several commands unique to a secure database. There are commands to tell what compartment and what the level the user is currently running in. There is also a command that will erase the history file. The erase-history command is provided so that for testing purposes the database can be reinitialized. In LDV the erase-history command will be available only to the DBSSO, and might be selective in its operation as to what portions of history are erased.

The LDV prototype has just a few relational operators. INQUIRE is rich in operators, and in the future we hope to add more of them to the prototype. For the time being, the operations are: project, select, join, count, and sort.

Notice that for the cases where the attribute does not exist, or the attribute is above the users level, or attributes the user is not allowed to see because of context constraints, the same message is returned.

```
Attribute ATTRIBUTE-NAME does not exist
in relation RELATION-NAME.
```

We present here the queries, as we provided them to our prototype, and the responses that it returned.

5.2. Recurring-Reports

```
>(urms)
Welcome to the LDV Prototype.

LDV> level

Level is SECRET

LDV> erase-history

This is a command that only the System Security
Officer (SSO) will be able use.

LDV> ;RR1: S
(sortrel
(project
```

```

(njoin
  (project employee-base '(Employee-Num Employee-Name))
  employee-address
)
'(Employee-Num Employee-Name Home-Address)
)
'(Employee-Name)
)

```

EMPLOYEE-NUM EMPLOYEE-NAME HOME-ADDRESS

369	ANDERSON	13th St
135	BECKER	Joyce St
1	BENDER	Crush St
573	BOURKE	Maple St
476	BRIMER	Munson Rd
249	BRIMMER	Hull Rd
309	BRINDLE	Taney St
255	BUKOWSKI	East Ave
266	CARPEN	Maple St
356	CARR	Farms Ct
254	CHANDLER	Greeley Rd
982	CHASE	Taylor Rd
694	CLARK	20th St
794	CURRAN	Laurel Rd
985	DAILEY	Main St
479	DELANEY	Cherry St
189	DENNY	Kirby St
850	DOYLE	Eads St
432	EBERT	Main St
956	ECKHOF	Crosly Rd
754	EPPLEY	King St
975	FALBO	Birth St
168	FISHER	Green Rd
231	FLIPPIN	Hill St
930	GIBSON	Cloud Dr
456	HALL	Clifton Rd
324	HILL	Maple St
649	HUTCHINSON	Wilson St
793	JACKSON	Sadler Rd
375	JORDAN	Lear Rd
876	KELLEY	Manor Rd
123	KHOSLA	Oak St
91	KLAREN	Ligth Rd
650	KNOLL	Gregg Ct
693	KOLLER	Joyce Rd
431	LANGE	37th St
435	LEVY	Minton St
853	MACCAULEY	Elk Pt
487	MAHONEY	Sutter Rd
955	MARSHALL	Shore Rd

186	MCALLISTER	Briar Rd
632	MCINTOSH	6th Rd
765	MITCHELL	Gypsey St
521	MYER	Lynn Ct
308	MYERSON	Filmore Rd
368	NAUROZ	Sterling
409	OBERLY	Pierce St
256	OPEL	Lee Hwy
791	OTTENBERG	24th St
857	PERKINS	Yoakum St
980	POLANCO	Hunter Rd
965	POLANKA	Ripley Rd
753	PYLE	Prince St
43	REYNOLDS	Landess St
678	RODRIGUEZ	Ripley Rd
931	SHANNON	Essex Ct
870	SIEBEL	Rose St
156	SMALL	Terry Rd
206	SOUSA	Pekay St
167	SPIRITO	14th St
391	STROTHER	Dover St
675	Smith E.	Rolling St.
354	Smith G.	17th St
239	Smith R.	Jordan Rd
489	Smith S.	River Rd
145	THOMAS	Peake St
545	TUCKER	34th St
378	URSINI	Essec Ct
572	WALTERS	Eaton St
624	WILLIAMS	Burke Ct
286	WOLCOTT	9th Rd
795	WOOD	25th St
520	YANCEY	Motley St
398	ZUZACK	Arden Rd

LDV> ; The style of the prototype is such that the attributes of a relation
; must be explicitly listed. Otherwise the prototype attempts to
; let the user see all the attributes at or below his level. Often this
; violates other security constraints so not all the tuples are released.
; The end result is that not all the tuples are returned to the user unless
; the user specifies the attributes explicitly. Note both types of queries
; are secure, they just return different results.

```

;
; RR2: S
; This query shows what happens when the attributes are not specified.
(sortrel
  (project
    (njoin employee-base employee-title)
    '(Employee-Name Job-Title)
  )
  '(Employee-Name)
)

```

 EMPLOYEE-NAME JOB-TITLE

ANDERSON	Secy
BUKOWSKI	Per Clk
CHANDLER	Per Clk
CHASE	Proc Clk
DELANEY	Proc Anal
ECKHOF	Adm Ast
HALL	Fin Anal
HUTCHINSON	Adm Ast
KHOSLA	Adm Clk
LEVY	Secy
MACCAULEY	Mgt Anal
MAHONEY	Secy
MYER	Per Anal
NAUROZ	Secy
PERKINS	Secy
REYNOLDS	Per Anal
RODRIGUEZ	Fin Anal
SMALL	Proj Mgr
TUCKER	Fin Anal
WILLIAMS	Mgt Anal
WOLCOTT	Fin Clk
YANCEY	Dept Mgr
ZUZACK	Proc Anal

```
LDV> ;RR3: S
(sortrel
  (project
    (njoin
      (project employee-base '(Department Employee-Name Employee-Num))
      employee-title
    )
    '(Department Employee-Name Job-Title)
  )
  '(Department Employee-Name)
)
```

 DEPARTMENT EMPLOYEE-NAME JOB-TITLE

ENG	CURRAN	Proj Mgr
ENG	DOYLE	Tech Staff
ENG	FISHER	Tech Staff
ENG	GIBSON	Tech Staff
ENG	MCINTOSH	Secy
ENG	OTTENBERG	Dept Mgr
ENG	SHANNON	Tech Staff
ENG	SPIRITO	Tech Staff
ENG	URSINI	Tech Staff
ENG	WOOD	Proj Mgr

EXEC	BENDER	Exec Secy
EXEC	CARR	VP-PM
EXEC	FLIPPIN	Exec Secy
EXEC	OPEL	Exec Secy
EXEC	SIEBEL	VP-HQ
EXEC	Smith E.	Adm Ast
EXEC	Smith G.	Pres
FIN	HALL	Fin Anal
FIN	NAUROZ	Secy
FIN	POLANKA	Dept Mgr
FIN	RODRIGUEZ	Fin Anal
FIN	TUCKER	Fin Anal
FIN	WOLCOTT	Fin Clk
INT	BRIMMER	Secy
INT	BRINDLE	Tech Staff
INT	CARPEN	Proj Mgr
INT	EPPLEY	Tech Staff
INT	KNOLL	Tech Staff
INT	MYERSON	Tech Staff
INT	OBERLY	Tech Staff
INT	PYLE	Tech Staff
INT	SOUSA	Proj Mgr
INT	Smith S.	Dept Mgr
OPS	BOURKE	Tech Staff
OPS	CLARK	Tech Staff
OPS	DENNY	Tech Staff
OPS	JACKSON	Secy
OPS	JORDAN	Tech Staff
OPS	KOLLER	Tech Staff
OPS	MARSHALL	Dept Mgr
OPS	Smith R.	Proj Mgr
OPS	WALTERS	Tech Staff
PER	ANDERSON	Secy
PER	BECKER	Adm Ast
PER	BUKOWSKI	Per Clk
PER	CHANDLER	Per Clk
PER	DAILEY	Adm Ast
PER	EBERT	Secy
PER	ECKHOF	Adm Ast
PER	HUTCHINSON	Adm Ast
PER	KELLEY	Dept Mgr
PER	KHOSLA	Adm Clk
PER	KLAREN	Secy
PER	LANGE	Secy
PER	LEVY	Secy
PER	MACCAULEY	Mgt Anal
PER	MCALLISTER	Adm Ast
PER	MYER	Per Anal
PER	PERKINS	Secy
PER	POLANCO	Adm Ast
PER	REYNOLDS	Per Anal
PER	SMALL	Proj Mgr
PER	STROTHER	Secy

PER	WILLIAMS	Mgt Anal
PROC	CHASE	Proc Clk
PROC	DELANEY	Proc Anal
PROC	MAHONEY	Secy
PROC	YANCEY	Dept Mgr
PROC	ZUZACK	Proc Anal
SEC	BRIMER	DMSO
SEC	FALBO	Secy
SEC	HILL	Dept Mgr
SEC	MITCHELL	Ast DBSO
SEC	THOMAS	Sec Clk

```
LDV> ; RR4: S
; Bukowski (Per Clk) ; SECRET NULL
(sortrel
  (project employee-base '(Employee-Name Salary))
  '(Employee-Name)
)
```

EMPLOYEE-NAME SALARY

ANDERSON	35
BECKER	60
BENDER	45
BOURKE	80
BRIMER	100
BRIMMER	35
BRINDLE	80
BUKOWSKI	55
CARPEN	110
CARR	125
CHANDLER	55
CHASE	55
CLARK	80
CURRAN	110
DAILEY	60
DELANEY	80
DENNY	80
DOYLE	80
EBERT	35
ECKHOF	60
EPPLEY	80
FALBO	35
FISHER	80
FLIPPIN	45
GIBSON	80
HALL	80
HILL	90
HUTCHINSON	60
JACKSON	35
JORDAN	80

KELLEY	90
KHOSLA	55
KLAREN	35
KNOLL	80
KOLLER	80
LANGE	35
LEVY	35
MACCAULEY	80
MAHONEY	35
MARSHALL	90
MCALLISTER	60
MCINTOSH	35
MITCHELL	90
MYER	75
MYERSON	80
NAUROZ	35
OBERLY	80
OPEL	45
OTTENBERG	90
PERKINS	35
POLANCO	60
POLANKA	90
PYLE	80
REYNOLDS	75
RODRIGUEZ	80
SHANNON	80
SIEBEL	125
SMALL	110
SOUSA	110
SPIRITO	80
STROTHER	35
Smith E.	70
Smith G.	150
Smith R.	110
Smith S.	90
THOMAS	35
TUCKER	80
URSINI	80
WALTERS	80
WILLIAMS	80
WOLCOTT	55
WOOD	110
YANCEY	90
ZUZACK	80

```
LDV> ;RR5: S
(sortrel
  (project
    (njoin
      (project employee-base '(Salary Employee-Name Employee-Num))
      employee-profit
    )
  )
  '(Employee-Name Salary Profit-Share)
```

```

)
' (Employee-Name)
)

```

```

-----
EMPLOYEE-NAME SALARY PROFIT-SHARE
-----

```

ANDERSON	35	10
BECKER	60	20
BENDER	45	20
BOURKE	80	30
BRIMER	100	30
BRIMMER	35	10
BRINDLE	80	30
BUKOWSKI	55	5
CARPEN	110	50
CARR	125	125
CHANDLER	55	5
CHASE	55	5
CLARK	80	30
CURRAN	110	50
DAILEY	60	30
DELANEY	80	20
DENNY	80	30
DOYLE	80	30
EBERT	35	20
ECKHOF	60	30
EPPLEY	80	30
FALBO	35	15
FISHER	80	30
FLIPPIN	45	15
GIBSON	80	30
HALL	80	20
HILL	90	45
HUTCHINSON	60	30
JACKSON	35	10
JORDAN	80	30
KELLEY	90	35
KHOSLA	55	5
KLAREN	35	30
KNOLL	80	30
KOLLER	80	30
LANGE	35	20
LEVY	35	20
MACCAULEY	80	20
MAHONEY	35	10
MARSHALL	90	40
MCALLISTER	60	20
MCINTOSH	35	10
MITCHELL	90	25
MYER	75	20
MYERSON	80	30

NAUROZ	35	10
OBERLY	80	30
OPEL	45	25
OTTENBERG	90	40
PERKINS	35	20
POLANCO	60	30
POLANKA	90	35
PYLE	80	30
REYNOLDS	75	20
RODRIGUEZ	80	20
SHANNON	80	30
SIEBEL	125	95
SMALL	110	50
SOUSA	110	50
SPIRITO	80	30
STROTHER	35	30
Smith E.	70	20
Smith G.	150	150
Smith R.	110	50
Smith S.	90	40
THOMAS	35	10
TUCKER	80	20
URSINI	80	30
WALTERS	80	30
WILLIAMS	80	20
WOLCOTT	55	0
WOOD	110	50
YANCEY	90	35
ZUZACK	80	20

```
LDV> ;RR6: S
(sortrel
  (where
    (project
      (njoin
        (project employee-base '(Employee-Name Employee-Num))
        employee-investigate
      )
      '(IS Employee-Name)
    )
    '(not (equal IS '1A))
  )
  '(IS Employee-Name)
)
```

IS EMPLOYEE-NAME

1B ECKHOF
 1B FISHER
 1B MACCAULEY
 1B POLANKA

1C KOLLER
1C SPIRITO
1C STROTHER
1C Smith E.
1C YANCEY

LDV> exit
NIL

>(urmts)
Welcome to the LDV Prototype.

LDV> level

Level is TOP_SECRET

LDV> ;RR7: TS
(sortrel
 (where
 (project
 (njoin
 (project employee-base '(Employee-Name Employee-Num))
 employee-credit
)
 '(Employee-Name CR))
 '(not (equal CR 1))
)
 '(Employee-Name)
)

EMPLOYEE-NAME CR

CHANDLER	2
ECKHOF	2
FISHER	4
KOLLER	2
MARSHALL	3
MITCHELL	4
NAUROZ	2
POLANCO	2
REYNOLDS	2
SOUSA	2
Smith E.	3
Smith S.	2
WOLCOTT	3

LDV> ;RR8: TS
(sortrel
 (project
 (njoin
 (project employee-base '(Employee-Name Employee-Num))

```

        employee-skill
    )
    ' (Employee-Name Skill-Code)
)
' (Employee-Name)
)

```

EMPLOYEE-NAME SKILL-CODE

BOURKE	8F
BRINDLE	D7
BRINDLE	C9
BRINDLE	5B
CARPEN	C9
CARR	C9
CLARK	8F
CLARK	3A
CURRAN	C9
CURRAN	5B
DENNY	8F
DENNY	3A
DOYLE	8F
EPPLEY	C9
EPPLEY	3A
FISHER	C9
FISHER	8F
GIBSON	D7
GIBSON	C9
JORDAN	8F
KNOLL	C9
KOLLER	8F
KOLLER	5B
MARSHALL	8F
MYERSON	D7
MYERSON	C9
OBERLY	C9
OTTENBERG	C9
PYLE	C9
PYLE	8F
PYLE	5B
SHANNON	D7
SHANNON	8F
SIEBEL	8F
SOUSA	D7
SOUSA	C9
SPIRITO	8F
SPIRITO	5B
Smith G.	C9
Smith S.	C9
URSINI	C9
WALTERS	D7

AUDITING IN SECURE DATABASE MANAGEMENT SYSTEMS

Sushil Jajodia
Shashi K. Gadia
Gautam Bhargava
Edgar H. Sibley

ABSTRACT

Existing databases make a distinction between the current and the past data. A logical structure exists only upon the *current* data to allow its retrieval through a query language. Any old information is either deleted or stored in the form of a *log*. Such an "audit trail" has an *ad hoc* structure and generally cannot be queried. In this paper, we describe a *database activity model* that imposes a uniform logical structure upon the past, present, and the future data. Moreover, there is never any loss of essential information in this model, thus providing a convenient mechanism for not only recording but a complete reconstruction of every action taken in the database at the same time.

Sushil Jajodia and Edgar Sibley are with the Department of Information Systems and Systems Engineering, George Mason University, Fairfax, Virginia 22030-4444.
Shashi Gadia and Gautam Bhargava are with the Department of Computer Science, Iowa State University, Ames, Iowa 50011.

1. INTRODUCTION

It is well known that in situations where the data is sufficiently sensitive, an audit trail becomes a necessity. Audit trails are normally used to retain superceded data and thus they provide a way to assign responsibility and accountability for events that take place in the system. Unfortunately, existing database systems treat only the current data in a systematic manner. The old information spooled to the log has an *ad hoc* structure and cannot be queried. We need a schema that defines the structure of this data and a query language which will enable us to interrogate the log so that any particular purpose could be efficiently achieved [2].

In this paper, we describe a *database activity model* that imposes a uniform logical structure upon the past, present, and the future data. Moreover, there is never any loss of essential information in this model, thus providing a convenient mechanism for not only recording but a complete reconstruction of every action taken in the database at the same time.

The organization of this paper is as follows. We argue in Section 2 for the need for our database activity model, and the two sections that follow contain some notation and definitions. Section 5 contains the description of our model. We have chosen an informal approach to introduce our model here. Section 6 shows how we may restrict access to the database. Finally, Section 7 summarizes the paper and lists some challenges that face us.

2. AUDIT REQUIREMENTS

Reference [3] contains a detailed discussion of audit requirements in trusted systems. In the Trusted Computer System Evaluation Criteria [1], the accountability control objective is stated to be as follows:

“Systems that are used to process or handle classified or sensitive information must assure individual accountability whenever either a mandatory or discretionary security policy is invoked. Furthermore, to assure accountability the capability must exist for an authorized and competent agent to access and evaluate accountability information by a secure means, within a reasonable amount of time and without undue difficulty.”

Existing databases clearly fail to meet the above objective. The central point of this paper is to propose a new data model called the *database activity model* which not only meets this objective, but provides a convenient mechanism for recording and controlling accesses to the database at the same time.

2.1. NEED FOR TWO TIME DIMENSIONS

In an auditable system, there must be a mechanism for a complete reconstruction of every action taken. This requires at least two time dimensions: one time dimension to order every operation against an object and a second time dimension to timestamp values of objects with their periods of validity in the real world [4, 8] (see also [9, 10]). This second time dimension is different from the first time dimension since

an earlier value of an object may be corrected later in time.

As an example, suppose we have a relation called EMPLOYEE having two attributes NAME and SALARY. Suppose we inserted the tuple (Smith, 25K) on 11/85. On 2/86, we discovered an error in Smith's salary and made a retroactive change to 30K. Now, in an auditable system we have in mind, it is possible to overwrite errors, but records are kept of any errors that are corrected. Thus, we need to retain both facts in the database:

1. Smith's salary was known to be 25K from 11/85 to 2/86.
2. On 2/86, a change was made to the salary from 25K to 30K, and this change was retroactive.

If we simply change the salary from 25K to 30K, we lose the second fact that an error was discovered in Smith's salary on 2/86, a change was made, and this change was retroactive. On the other hand, if we just change the value of the first time parameter from 11/85 to 2/86, we lose the the first fact that the salary was known to be 25K from 11/85 to 2/86.

Therefore, we need two different time parameters to describe above situation. We call the first time dimension the *transaction time*, the time at which the information is stored in the database and the second time dimension the *valid time*, the time when the information in the database is valid.

3. THE INTERVAL NOTATION

Let S be a linearly ordered set. By the closed interval $[a,b]$, we mean the set $\{x \in S : a \leq x \leq b\}$. A half-open interval is of the form $[a,b)$ or $(a,b]$ and denotes the set $\{x \in S : a \leq x < b\}$ or the set $\{x \in S : a < x \leq b\}$, respectively. Finally, an open interval (a, b) is the set $\{x \in S : a < x < b\}$. In this paper, we take S to be either the set of natural numbers or the real numbers. In this case, we denote by $[a,\infty)$ the set $\{x \in S : x \geq a\}$.

4. THE RELATIONAL MODEL

We work in the context of the relational model of data. A *relation scheme* R is a collection of attributes. K is said to be a *key* of R if the functional dependency $K \rightarrow R$ holds and if K does not contain a proper subset K' such that $K' \rightarrow R$ holds. A *relational scheme* \mathbf{R} is a collection $\{R_1, \dots, R_n\}$ of relation schemes, and a database r over \mathbf{R} is a set of relations $\{r_1, \dots, r_n\}$ such that each r_i is a relation over R_i . We assume that transactions are used to create, modify, delete, and retrieve data from the database relations in r .

4.1. SECURITY CLASSIFICATION OF DATA ITEMS

In our model, a user accesses the databases by executing a *transaction*, and the security level of a data item in the database takes on the level of the transaction that modifies it or creates it. The mandatory security is enforced by allowing a transaction to complete if its security level dominates the security level of all data that it affects.

5. THE DATABASE ACTIVITY MODEL

Let R be a relational scheme. The *database activity scheme* over R is a triple $\langle D, \text{shadow}, Q\text{-Log} \rangle$ where

- For every relational scheme R in R , there is a two-dimensional temporal relation (as illustrated in Figure 1) in D . It encapsulates the complete history of each and every modification made to a value of an attribute in R .
- For every r in r , there is a relation $\text{shadow}(r)$ in Shadow . $\text{Shadow}(r)$ records the circumstances surrounding the updates made to r .
- $Q\text{-Log}$ is a single relation. It is essentially a log of all queries.

We describe each of these in detail next.

5.1. THE TWO-DIMENSIONAL TEMPORAL RELATIONS

We will describe a two-dimensional temporal relation [6, 7] that incorporates the concept of time at the logical level of design and use it to represent the two time dimensions. It uses time intervals in the range $[0, \infty)$, called the *valid time*, to timestamp a value with its period of validity in the real world, giving rise to a historical relation. Since our knowledge of history changes with time requiring updates to the historical database, another time dimension in the interval $[0, \text{now}]$, called the *transaction time*, is used to capture history of operations on objects, where the symbol *now* is used to denote the changing value of the current time. In this way, our changing knowledge of the real world is modeled through two dimensional timestamps, and $[0, \text{now}] \times [0, \infty)$ serves as the universe of time. The resulting relation is called a *two-dimensional temporal relation*. An example of a two-dimensional temporal relation is shown in Figure 1.

NAME	SALARY	DEPT
$[8, \text{now}] \times [11, \infty)$ John	$[8, 53) \times [11, \infty)$ 15K $[53, \text{now}] \times [11, 49]$ 15K $[53, \text{now}] \times [50, \infty)$ 20K	$[8, 40) \times [11, \infty)$ Toys $[40, \text{now}] \times [11, 44]$ Toys $[40, \text{now}] \times [45, \infty)$ Shoes
$[48, \text{now}] \times [48, \infty)$ Doug	$[48, \text{now}] \times [48, \infty)$ 20K	$[48, \text{now}] \times [48, \infty)$ Auto

Figure 1. A two-dimensional temporal relation

Let us consider the values taken by the attribute SALARY for the employee John in Figure 1. The semantics of the values is that “there was a transaction posted at time 8 declaring John’s salary to be 15K from time 11 onwards, but another transaction that was posted at time 53 updated John’s salary to be 15K from time 11 to 49 and 20K from time 50 onwards.”

For our example in section 2.1, the two-dimensional temporal relation is given in Figure 2.

NAME	SALARY
$[11/85,now] \times [11/85,\infty)$ Smith	$[11/85,2/86) \times [11/85,\infty)$ 25K $[2/86,now] \times [11/85,\infty)$ 30K

Figure 2. The two-dimensional temporal relation for the example in Section 2.1

It is easy to verify that the temporal relation given in Figure 2 cleanly and accurately models the situation described in Section 2.1.

Thus, we see that the reexamination of historical data is possible in a two-dimensional temporal relation, and this allows corrections to be made as necessary (but still retains the fact and data associated with the original “estimate” or “observation” for future query and examination). Moreover, the database may carry predictions of future operations or values; e.g., the data may contain the required list of actions for the following weeks, such as times and places of any meetings, personnel actions to be made, etc.

5.2. SHADOW AND QUERY RELATIONS

Our model maintains for each two-dimensional temporal relation, a relation called a *shadow* relation that records information about all updates to the relation and a relation called a *query-log* relation which is essentially a log of all queries concerning the relation.

More formally, let R be a relation scheme in \mathbf{R} , and let K be the key for R . Then the attributes of *shadow*(R) are the key K , the transaction time (TT), the transaction security level (TSL), the authorizer of the transaction (AUTHORIZER), the user of the transaction (USER), and the reason for executing the transaction (REASON). See Figure 3.

The *Query-Log* relation is the relation containing a log of all the queries. It has the following attributes: the query (QUERY), the query time (TT), the query security level (TSL), and the maker of the query (USER). See Figure 4.

One interesting property is that once we supplement the two-dimensional temporal relations by the shadow and query-log relations, the transaction log can be restored from these three relations. There is never any loss of essential information, and therefore, we have the complete audit trail concept. We illustrate this next by way of an example.

Suppose our database consists of a single relation scheme EMP with attributes NAME, SALARY, and DEPT such that the attribute NAME is the key. Suppose for each update, we wish to keep track of the following audit information: transaction time (TT), transaction security level (TSL), who authorized the update (AUTHORIZER), user making the update (USER), and reason for the update (REASON). For each query, we store the query (Q-id), time of the query (TT), transaction security level (TSL), and user making the query (USER). Consider the following sequence of

transactions:

- T₁. TT = 8; TSL = S; User = Mark.**
insert (NAME: [11,∞) JOHN; SALARY: [11,∞) 15K; DEPT: [11,∞) Toys)
with (Authorizer = Don; Reason = New Employee);
- T₂. TT = 40; TSL = TS; User = Ryne.**
modify (NAME: [11,∞) JOHN) to (DEPT: [11,44] Toys, [45,∞) shoes)
with (Authorizer = Don; Reason = Reassignment);
- T₃. TT = 42; TSL = S; User = Vance.**
Q1: What is John's salary?
- T₄. TT = 48; TSL = TS; User = Rick.**
insert (NAME: [48,∞) Doug; SALARY: [48,∞) 20K; DEPT: [48,∞) Auto)
with (Authorizer = Joe; Reason = New Employee);
- T₅. TT = 53; TSL = S; User = Dameon.**
modify (NAME: [11,∞) JOHN) to (SALARY: [11,49] 15K, [50,∞) 20K)
with (Authorizer = Don; Reason = Promotion);
- T₆. TT = 54; TSL = S; User = Andre.**
Q1: What is John's salary?
- T₇. TT = 55; TSL = S; User = Mitch.**
Q2: What is John's department?
- T₈. TT = 56; TSL = S; User = Don.**
Q3: Who made enquiries about John's salary?
- T₉. TT = 58; TSL = TS; User = Paul.**
Q2: What is John's department?

Corresponding to these transactions, the two-dimensional temporal relation is given in Figure 1. Figures 3 and 4 give the resulting shadow and query-log relations, respectively.

NAME	TT	TSL	AUTHORIZER	USER	REASON
John	8	S	Don	Mark	New Employee
John	40	TS	Don	Ryne	Reassignment
Doug	48	TS	Joe	Rick	New Employee
John	53	S	Don	Damon	Promotion

Figure 3. The shadow relation

QUERY	TT	TSL	USER
Q1: John's SALARY	42	S	Vance
Q1: John's SALARY	54	S	Andre
Q2: John's DEPT	55	S	Mitch
Q3: USER ID of Q1	56	S	Don
Q2: John's DEPT	58	TS	Paul

Figure 4. The query-log relation

It is easy to verify that that all the transactions can be completely restored using the three relations in our database activity model: The values of the key NAME in the two-dimensional temporal relation (Figure 1) and in the shadow relation (Figure 3) set up a logical correspondence between the tuples in the two relations. By including the transaction time, the logical correspondence can be refined to a one-to-one correspondence between all the updates to the temporal relation and the tuples in the shadow relation. As a result, the transactions T_1 , T_2 , T_4 , T_5 can be completely restored from the relations in Figures 1 and 3. The transactions T_3 , T_6 - T_9 can be completely restored from the relations in Figures 1 and 4. Finally, using the transaction time TT, we can order all nine transactions to obtain the original sequence of transactions.

6. RESTRICTING ACCESS TO THE DATABASE

Since our database activity model contains information about the complete activity in the database system, we next consider how we may restrict user access to the database.

6.1. SECURITY LEVEL OF TRANSACTIONS

The addition of the security level of the transaction during an operation on the database allows the incorporation of still further concepts of tagging the data with its security level at the object level. In our model, the data takes on its security level

because of the level of authority implied by the transaction creating or modifying the elements of the object. Thus, an object is itself not classified, but its parts carry different (possibly time-variant) security constraints. As a consequence, the response to a query or update transaction is dependent on the security level of the system user or the level at which the user is operating, in conjunction with the *security policy* that applies at the time of the operation.

In our example, this means that Andre's query about John's department (see transaction T_7) will not succeed (since security level of T_7 is secret (S), while the transaction T_4 which made the most recent update to the DEPT was at the top secret (TS) level). On the other hand, Paul who makes the same query in transaction T_9 will be given the right answer since security level of T_9 is same as that of T_4 .

The preceding paragraph raises an interesting policy question requiring careful examination. What should be the answer to the query T_7 ? Should the system give the last entry at the secret level, namely toys? Or should the system say "not available?" Another system response might be to say "no such person." Does this mean the system can or cannot give John's current salary to Andre which was updated at the secret level by transaction T_5 ? This interaction between the query language, the time, and the security policy appears to be complex and needs further examination.

6.2. THE USER HIERARCHY

A second way we limit access of a user is by filtering the database through a user hierarchy defined as follows.

The Master user. The master user has access to the whole database (e.g. Figure 1) and enjoys the power to query errors and updates in the database.

The History user. This user only sees information filtered through the time domain $now \times [0, \infty)$, and thus has access to only the most up-to-date knowledge of the history of objects (e.g. the relation in Figure 5). Errors which have been corrected are not available to the historical user. Such a user can ask questions of historical nature such as "When did Tom's salary increase?"

NAME	SALARY	DEPT
[11,∞) John	[11,49] 15K [50,∞) 20K	[11,44] Toys [45, ∞) Shoes
[48,∞) Doug	[48,∞) 20K	[48,∞) Auto

Figure 5. The two-dimensional temporal relation as seen by a history user

The Snapshot user. The filter in this case is the time domain $now \times now$, and this user sees precisely what is available to a traditional user in databases (e.g. see Figure 6 below). In our framework such a user lies at the lowest level of the user hierarchy.

NAME	SALARY	DEPT
John	20K	Shoes
Doug	20K	Auto

Figure 6. The two-dimensional temporal relation as seen by a snapshot user

The Audit user. This user sees the data filtered through the time domain $\{(t,t') : t' \leq t\}$, does not have the concept of future (since $t \leq \text{now}$), and only sees the actions taken by the organization. Since the rest of the world is only affected by the actions that have been taken by the organization, this user can deal with the public relations and legal aspects of the enterprise.

The Rollback Snapshot User. This user sees the data filtered through the time domain $t \times t'$ for fixed values of t and t' . Like the snapshot user, this user sees a snapshot relation, with the difference that this user can see a snapshot at any point in the past, present, or the future. As an example, a rollback snapshot user will be given the snapshot in Figure 7 for the time domain 8×11 and the snapshot in Figure 8 for the time domain 48×50 .

NAME	SALARY	DEPT
John	15K	Toys

Figure 7. The two-dimensional temporal relation filtered through the time domain 8×11

NAME	SALARY	DEPT
John	15K	Shoes
Doug	20K	Auto

Figure 8. The two-dimensional temporal relation filtered through the time domain 48×50

7. SUMMARY AND CHALLENGES

Our new model seems to be a building block for a general-purpose database system that holds the promise for a perfect logical organization of past, present, and future data. This aspect of our model can be exploited further to allow automatic regeneration and review of events that occur in the database system to detect and

discourage security violations. Furthermore, our model allows access to the database via transactions which maps quite well to the model proposed recently by Clark and Wilson [5], and it appears that there would be a high degree of compatibility between our model and the object-oriented data model. To harness the promise of our model, further theoretical advances have to be made, and then a prototype has to be developed. It is presumed that this will lead to further investigation of the audit trail capabilities in answering questions dealing with possible breaches and other problems.

References

1. Department of Defense Trusted Computer System Evaluation Criteria, August 15, 1983.
2. Report of the Invitational Workshop on Integrity Policy in Computer Information Systems, October 27-29, 1987.
3. A Guide to Understanding Audit in Trusted Systems, National Computer Security Center, June 1, 1988.
4. L. A. Bjork, Jr., "Generalized audit trail requirements and concepts for database applications," *IBM Systems Journal*, vol. 14, no. 3, pp. 229-245, 1975.
5. David D. Clark and David R. Wilson, "A comparison of commercial and Military computer security policies," *Proc. IEEE Symp. on Security and Privacy*, pp. 184-194., 1987.
6. Shashi K. Gadia, "A homogeneous relational model and query languages for temporal databases," *ACM Tran. on Database Systems*, vol. 13, no. 4, pp. 418-448, December 1988.
7. Shashi K. Gadia and Chuen-Sing Yeung, "A generalized model for a relational temporal database," *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pp. 251-259, June 1988.
8. V. Lum, P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner, and J. Woodfil, "Designing DBMS support for the temporal dimension," *Proc. ACM SIGMOD Int'l. Conf. on Management of Data*, pp. 115-130, May 1987.
9. Richard Snodgrass and Ilsoo Ahn, "Temporal databases," *IEEE Computer*, vol. 19, no. 3, pp. 35-42, September 1986.
10. Richard Snodgrass, "The temporal query language TQuel," *ACM Trans. on Database Systems*, vol. 12, no. 2, pp. 247-298, June 1987.

Secure DBMS Auditor: Audit in Trusted Database Management System Environments

Trusted Information Systems is pleased to respond to the Task Order Request from Rome Air Development Center for Statement of Work PR NO. B-9-3610, dated 15 February 1989, "Secure DBMS Auditor".

INTRODUCTION

Currently, there are few database management systems or prototypes that provide *all* of the trust features and assurances associated with any one class of DOD 5200.28-STD, the *Department of Defense Trusted Computer System Evaluation Criteria (TCSEC)* or of its derivative interpretations and guidelines. Some requirements in these documents tend to go considerably beyond current industry standards and practice for database management systems. Others are unique to multilevel security applications, an area where there is very little published DBMS experience.

However, this is not to say that there is *no* experience in the field. Many commercial database management systems implement features and mechanisms to limit and control access by authorized and unauthorized users. These features and mechanisms have traditionally included limitations on modes of access (e.g., constraints on queries) by users to defined subsets (views) of databases. These constraints, largely employing *discretionary* rather than *mandatory* access controls, have not just addressed which users may observe or modify portions of a database or its definition, but have additionally addressed issues such as *semantic integrity, data consistency, and transactional atomicity*.

In addition, legal and operational requirements have often necessitated the implementation of recording, logging and journaling mechanisms. These mechanisms have often served dual interests: on the one hand they have provided a means of associating a date and time, and often a designated person, with designated database transactions (especially with every modification or update to the database); on the other hand, the mechanisms have provided means for selective rollback and recovery from various malformed transactions and database updates, as well as diagnosis of the causes of certain losses of database integrity.

To many database administrators, the traditional database management system recording, logging and journaling mechanisms produce the data that would be of interest in an audit trail. However, there is little collected experience in the use of such transaction recording data for the express purpose of detecting attempted or successful violations of access control policies and constraints.

The 18 November 1988 draft of the *Trusted DBMS Interpretation of the TCSEC (TDI)* does not prescribe the specific data structures or user actions¹ that need to be audited, nor is it particularly enlightening in its interpretation of database audit tool requirements or of the techniques needed for the analysis of audit data. Rather, after reproducing the base *TCSEC* audit requirement, it states (for class A1):

The objects referred to are the DBMS objects specified in the security policy. Events, in the case of a DBMS, are interpreted as being individual operations performed by the DBMS (e.g., updates, retrievals, inserts) and not just the invocation of the DBMS. Individual operations performed by the TCB, if totally transparent to the user, need not be audited.

Where the TCB subset concept is employed, a TCB subset may maintain its own security audit log, distinct from that maintained by more privileged TCB subsets, or it may utilize an audit interface provided by a different TCB subset allowing the audit records it generates to be processed by that TCB subset.

Where the TCB subset concept is employed, responsibility for auditing events that may be used to exploit covert channels need not be allocated to the TCB subset containing the channel, but must be allocated to that subset or a more privileged subset provided that the criterion is met for the composite TCB as a whole.

If the TCB subset uses different user identifications than a more-privileged subset, there shall be a means to associate audit records generated by different TCB subsets for the same individual with each other, either at the time they are generated or at some later time.

Any TCB subset wherein events may occur that require notification of the security administrator shall be able to: 1) detect the occurrence of these events; 2) initiate the recording of the audit trail entry; and 3) initiate the notification of the security administrator. The ability to terminate these events may be provided in either the TCB subset within which they occur or in the TCB subset(s) where actions that lead to the event were initiated.

The above interpretation is based on the definition of 'object' that is given in the trusted DBMS (TDBMS) policy model, subject to the detailed discussion in Appendix I of the draft *TDI*. The draft *TDI*'s interpretation of 'object' and audit may not be sufficiently well-defined to capture the significance of context to the sensitivity of information as they apply to the database management context. In part, this may be because the draft *TDI* only associates sensitivities and classifications with static *containers of data* rather than with dynamic attributes of derived *information*.²

¹The draft *TDI*, like the *TCSEC*, imposes individual accountability requirements that require the ability to associate security-relevant events with the specific user on behalf of whom they occur. However, the draft *TDI* also hints that certain server-based architectures *may* be acceptable, even though strict compliance with the individual accountability requirement may not be possible. The issue is identified briefly in the Internal Servers Guideline.

²To be fair, it should be pointed out that the draft *TDI* allows for the treatment of a view definition as a classified object. A view definition is an association between data elements and many database management systems use view definitions as a means of controlling access to information. At present, there is controversy over whether view definitions should be treated as classified objects and, if so, what the relationship should be between their classification and that of the base data elements that comprise the view. However, it appears that a subject would be constrained from observing the contents of a view unless the subject were permitted to observe the individual data entities that would comprise the view. This latter constraint could conceivably have different properties for dynamically computed views than for the retrieval of statically-defined views, and may be of interest both for imposing consistency on actual access control mediation and for completeness of audit information.

In traditional *TCSEC* Division A and Division B systems it is important that all access paths to a specific object be mediated consistently. While this should be the case with respect to database objects, it need be noted that every view is equivalent to the evaluation of some query. It appears that the draft *TDI* would treat the classification of a query as the least upper bound of the classifications of the containers of data structures referenced in the evaluation of the query, while there remains the possibility that the equivalent defined view may be found to be of a different classification. The draft *TDI* does not particularly address the classifications or sensitivities of *semantic interrelationships* between the objects of a database or database view. Some would claim that it is from these *interrelationships* between database elements, rather than from the *containers* of the referenced data that is derived the actual sensitivities or security classifications of the referenced *information*.

Because of these issues, it appears that there may be justification for associating the text of each query with the other audit trail data generated by the trusted database management system's evaluation of the query.

It is often the case that the interrelationships are the basis for entries in the backup and recovery logs. The *TDI*'s audit rationale appears to leave as optional the decision of whether the security audit and integrity recovery logs be integrated or maintained separately:

....The emphasis of the audit criterion is to provide individual accountability for actions by users. The goal is not the same as that for a conventional DBMS log supporting backup and recovery. The security audit log need not, therefore, be integrated with a conventional DBMS audit log, although the tCB may provide mechanisms that are useful for the purposes of backup and recovery as well....

Hence, it appears that the goals of associating accountable actions with identified users and those of preserving the semantic integrity of databases are not necessarily at odds, and there is no explicit restriction on seeking to extend the audit and logging technologies used in current DBMS practice. However, it also seems that the questions of *how* and *what* to audit are unsettled research issues.

The ROLES Facility

Bill Maimone
Oracle Corporation

ABSTRACT

The **ROLE** facility extends the SQL data language to simplify the management of system and object privileges in user applications. A **ROLE** defines a group of privileges which can be assigned to users and other roles. The syntax and semantics of the **ROLE** facility are given along with an explanation of design decisions and open issues.

I. Introduction

The security facilities described in the ANSI SQL database language require users to maintain a record of all privileges needed to run each application so that they can be granted when users become authorized for an application, or revoked when the authorization is withdrawn. They also require that privileges on each database table be granted or revoked individually. This presents a significant management burden if a database supports even a modest number of users, applications, and tables. In practice, this burden often results in poor management and hence poor security. Roles are designed to make the process of privilege allocation more manageable and hence more secure.

Roles also support the reassignment of responsibility for security management without the significant complications that currently exist. If a user is responsible for database security and is replaced by a new user, the current SQL security facilities require that all privileges granted by the old user be granted again by the new user and then revoked by the old user. Again, this presents a significant burden which can result in poor security management.. This paper describes a roles facility as a solution to these problems.

A role can be considered a group of privileges. For example, all of the database privileges needed by an application can be granted to a role. To authorize a user to run the application, all that need be done is grant the role to the user. Similarly, to withdraw this authorization the role would simply be revoked from the user.

In addition to supporting the granting of roles to users, roles may be granted to other roles. This enables, for example, an `Accounting_Clerk` role to be defined which can contain the roles `Accounts_Payable_Clerk`, `Accounts_Receivable_Clerk`, and `General_Ledger_Clerk`.

The remainder of this paper describes the proposed syntax and semantics of the role facility proposed to ANSI SQL (X3H2) committee in X3H2-89-171 and planned for the next release of the ORACLE RDBMS.

II. CREATING ROLES

The `CREATE ROLE` command is used to create a role as follows:

```
CREATE ROLE rolename
```

This command defines the name of a new role, owned by the user who creates it, with no object or system privileges. Each role name must be different from all other role or user names in the systems. Since role names and user names share a database-wide name space, a user must have a specific system privilege to create a role.

```
CREATE ROLE MANAGER  
CREATE ROLE PAYROLL_CLERK
```

III. Granting Privileges to Roles

Once a role has been created, the `GRANT` command is used to assign system and object privileges to it. The syntax is similar to that of the existing `GRANT` to users, except that the `WITH GRANT OPTION` cannot be used. When granting to a user, the `WITH GRANT OPTION` allows the grantee to propagate privileges to other users. The `GRANT OPTION`

for roles is not provided because it would significantly complicate the algorithm for REVOKE while adding little marginal benefit.

```
GRANT UPDATE ON EMPLOYEES TO PAYROLL_CLERK
GRANT ALL ON EMPLOYEES TO MANAGER, DIVISION_MANAGER
GRANT CREATE TABLE TO MANAGER
```

The user does not have to own or have been GRANTED the role to assign privileges to it, but must have GRANT OPTION on any privilege in order to grant the privilege to a role.

IV. Granting Roles to Users and Roles

Once a role has been created, it can be assigned to users other than the owner with a GRANT command. The GRANT command can also be used to "nest" roles within other roles, so that creating the privileges for a job that combines the requirements of several job functions can be a simple one-step process. The following is the syntax for this form of the GRANT command to grant the named roles to a group of other roles and/or users.

```
GRANT ROLE(S) rolename [,rolename]...
TO [user | role] [, user | role]...
[WITH ADMIN OPTION]
```

If the user is not the owner of the role, he previously must have been granted a role WITH ADMIN OPTION in order to grant it to another user. The user need not have been assigned the role receiving the GRANT. In other words, to GRANT role A to role B, you must own or have ADMIN OPTION on role A, but not necessarily on role B.

The user granting a role may either have been granted the ADMIN OPTION directly as a user, or have been granted a role that contains this option. For example, suppose that user Carla owns a role called AP_CLERK that includes SELECT, INSERT, and UPDATE privileges on the EMPLOYEE table. User Bill CREATES a role called GEN_CLERK, but lacks any privileges on the EMPLOYEES table. Carla can assign AP_CLERK to GEN_CLERK as follows:

```
GRANT ROLE AP_CLERK TO GEN_CLERK WITH ADMIN OPTION
```

Through this role, Bill now has the privileges that he lacked as a user. He may also grant the role AP_CLERK, but not its individual privileges, to other users. Circularity of role assignments is not permitted. For example, Bill would get an error if he tried the following command:

```
GRANT ROLE GEN_CLERK TO AP_CLERK
```

A role GRANTED to another role is its subset, and two roles cannot both be subsets of one another.

V. Revoking Roles

The REVOKE command is used to remove privileges or roles from users or roles. The syntax used to remove privileges from roles is:

```
REVOKE object_priv [,object_priv] ...
ON [owner.]object
FROM ROLE[S] rolename [,rolename]
```

In order to REVOKE privileges from a role, a user must either own the object for which those privileges are defined, or have the privileges on the object via the WITH GRANT OPTION. The named privileges will immediately be removed from the role wherever it is assigned. To continue with the preceding example, Carla can decide to REVOKE UPDATE and INSERT from the role AP_CLERK.

```
REVOKE UPDATE, INSERT ON EMPLOYEES FROM ROLE AP_CLERK
```

Bill's role of GEN_CLERK would automatically lose these privileges.

The syntax to REVOKE a role from a user is similar:

```
REVOKE ROLE[S] rolename [,rolename]...
```

FROM rolename | user [, rolename | user]...

A user must either own or have ADMIN OPTION on the role being revoked, but not on the role(s) being revoked from. Unlike GRANT OPTION, ADMIN OPTION has no hierarchy based upon who granted privileges to whom; with ADMIN OPTION, a user with ADMIN on a role can REVOKE that role from anybody except its owner.

Only "top layer" roles may be revoked, e.g. if user A has been granted role B and role B has been granted role C, role B can be revoked from user A, implicitly revoking role C. Role C can be revoked from role B; this will apply not just to user A but to all users in the system with role B. Role C, however, cannot be directly revoked from user A independently of role B.

VI. Dropping Roles

Roles can be eliminated by using the DROP command. A user must own a role to DROP it. When a role is dropped, all users and other roles to which it has been assigned will lose access to it, and its definition will be removed from the system. The syntax is simply:

DROP ROLE rolename

VII. Enabling and Disabling Roles

The roles assigned to a user can be enabled or disabled for a given session or part of a session. This means that a role which is required for a given task can be "turned on" at the commencement of the task and "turned off" when it is finished. This feature allows tighter control of the usage of role privileges. The syntax to enable or disable a roles is:

ALTER SESSION [ENABLE | DISABLE] ROLE[S] [role_list | ALL]

This command applies to the user issuing it for the duration of the current logon session. The user must previously have been GRANTED or own the role in order to enable it; the role must currently be enabled in order to be disabled. A role is automatically enabled for the owner when created. The CREATE USER and ALTER USER commands can define the default enabled roles for a given user.

VIII. Viewing Roles Status

The user can determine his current roles and privileges through four new non-updatable data dictionary views.

The view USER_ROLES displays all roles created by the user. This view is parameterized so that each user sees only his own roles.

The view ALL_ROLES displays all roles present in the database. This view is available only to a database administrator.

The view SESSION_ROLES displays all roles granted to the user, and indicates whether each is currently enabled or disabled in the current session.

The view SESSION_PRIVILEGES displays all system privileges currently available to the user. This is the union of all system privileges granted to the user and all currently enabled roles.

Other views allow a user to display privileges on his objects granted to roles and users.

IX. Open Questions

Should roles own objects? If a role represents an application, it follows that a role might sensibly own all of the objects used in that application. It is unclear what privileges allow a user to create an object in a role or even a sub-role.

X. Summary

The roles facility simplifies the administration of system and object privileges in the SQL database language by grouping sets of privileges into a new object. Roles can be used to group together the set of privileges required to run an application so that the administrator can more easily manage both users and applications. The simplicity and flexibility offered by the roles facility facilitates better control over privileges and therefore leads to a more secure system.

DAC Mechanisms in Trusted Database Management Systems

Ronda R. Henning
Harris Corporation
P.O. Box 98000
Melbourne, FL 32902

July 2, 1989

Abstract

Discretionary access controls have often been considered a very weak security mechanism. However, database management systems use discretionary access controls as an integral part of their data integrity mechanisms and as an important flexibility feature. In a discussion at the Workshop on operating system/database management system dependencies, the issues involved in high assurance discretionary controls were addressed. This paper summarizes the key points of this discussion.

1 Introduction

Discretionary access controls in secure operating systems are considered easily circumventable and, as a result, relatively untrustworthy. Yet they do provide the notion of "ownership" of data in a trusted system. In a database management system, the ownership of data becomes a more critical issue. Individual databases may be owned by separate departments. Within those departments, only certain individuals may be granted the capability to modify, delete, create, or read data. These capabilities may be expressed as view specifications, relation, record, or data item access controls, or as content-dependent constraints.

Are such mechanisms sufficient, or do additional mechanisms need to be employed to provide the granularity of protection desired and the functionality to support user-defined applications?

2 Basic Operating System Controls

The basic discretionary access control mechanisms in conventional trusted operating systems can be defined as access control lists, group definitions, and role definitions. A brief overview of each is included for completeness.

2.1 Discretionary Access Control Lists

Discretionary access control lists are collections of users or groups of users and their associated access modes. Most trusted operating systems use the access modes of read/write/execute, where the modes are defined as:

- Read - access to contents of a file, without the capability to modify them. Read access allows a user to examine the data, but does not allow data modifications.
- Execute - access to a file, without the capability to examine or modify the contents.
- Write - access to a file, with the capability to append data or to modify the present contents.

There are variations on these basic access control modes. Some systems support a control access mode which is used to limit propagation of access rights. If a system supports control mode, a user can only grant his access modes to another user if he has been granted control mode access. Other systems support access modes that allow for modification of existing files, and creation or deletion of files.

2.2 Group Definitions

Users performing a similar function or working on a given project are often logically grouped together on a computer system. This simplifies the job of the system administrator when it is necessary to add/delete users from the system. Simply removing a user from a given group that has been established may remove that user's capability to access the group's data. (If the user has been granted access to the data from various groups, all such access rights must be traced and revoked.)

2.3 Role Definitions

Roles can be used to define a given user's privileges and authorizations. A role can be as all-inclusive as system administrator or security officer, or as narrow as "read-only user of non-sensitive personnel data". A user assumes a given role either by virtue of being in a particular group or by being granted specific access rights and privileges by the system security officer or system administrator.

3 Are These Mechanisms Sufficient?

Is judicious use of the discretionary access control mechanisms of an operating system sufficient for the discretionary access control requirements of a database management system? The consensus answer was no. There are several deficiencies in operating system discretionary access control mechanisms that make them insufficient for a trusted database management system. The problems with operating system security mechanisms in a database management environment are that they do not provide a fine granularity of control over data and they do not currently incorporate robust, flexible discretionary security mechanisms.

4 Defining DBMS DAC Requirements

Since operating system discretionary access controls are insufficient, what comprises the set of DAC mechanisms in a trusted database management system? The group discussed the DAC mechanisms considered sufficient for a DBMS. These included access modes, auditing, content-dependent controls, and view specifications.

4.1 Access Modes

The group consensus was that access modes as supported in SQL are the minimal discretionary access control mechanisms required in a trusted database management system. These modes include:

- insert, the ability to add data;
- delete, the ability to remove data;
- select, the ability to retrieve data;
- update, the ability to modify existing data;
- grant, the ability to propagate access modes to other users;
- revoke, the ability to rescind access modes of other users.

In a database management system, the operating system's discretionary access modes may be used to provide an interface to the file system. They do not provide sufficient granularity of control for a database management system to effectively restrict user access. The additional flexibility of SQL access modes is necessary to allow a finer degree of control.

4.2 Audit Modes

Transaction journals for rollback/recovery in database management systems usually do not include the detailed data access activity information required for security auditing activities. Assuming that trusted database management systems improve their auditing capabilities, it will be necessary to examine the audit logs for possible compromises. These audit examination techniques must be considered as part of the discretionary access control mechanisms in the database management system.

All users must have write access to the audit logs to record their security-relevant activities. Security officers and system administrators require read access to the data for examination purposes. The volume of data which must be examined is defined by the granularity of the audit as specified by the security officer. Audit reduction techniques must be developed to allow the security officer to select items of interest.

Another concern with audit data is delegation of audit responsibilities. In this scenario, reduction of the audit log would allow a given group's administrator to be responsible for monitoring of his group's audit records. This approach assumes that the security officer would delegate this responsibility to the group administrators, who in turn would be responsible for reporting problems to the security officer for further examination.

4.3 Content-dependent DAC

Discretionary controls that are content-dependent appear to offer a good mechanism to control some types of information. For example, if only one value for a given data item is considered sensitive, it would be a simple matter to write a contents check to ensure that the sensitive value is not inadvertently compromised.

This approach, however, does provide a very large covert channel opportunity because a filtering technique is used to produce the final query result. One must search the database and select the requested information, considering it as an intermediate result. The intermediate result must be scanned to determine if a given data element is sensitive, and if so, remove it from the final result returned to the user. The filter mechanism can easily be exploited by cooperating users. The preferred approach to content-dependent discretionary access control is a view mechanism. This avoids the problems associated with filters, and ensures that the retrieved data contains only what was requested by the user and allowed by the security policy of the database.

4.4 View Specification

Views have been considered one of the more flexible security mechanisms in a database management system. They have been proposed as an alternative mechanism for content-dependent security. The current view definition languages do not support security constraints. One of the areas for additional research is whether or not view definition languages need to support security specifications.

5 Assurance and Discretionary Mechanisms

There has been considerable debate as to whether the assurance requirements for discretionary access controls evolve upward through the Trusted Computer System Evaluation Criteria's (1) levels of trust. Beyond the C2-level, there are no new assurance requirements for discretionary access controls.

Since this is the case there is a question as to whether a model must be generated for discretionary access controls. If discretionary access control mechanisms demonstrate no information flow between mandatory sensitivity levels or trusted path communication that would effect sensitivity labels, they should not need a formal model. The extent of a model for discretionary access control mechanisms should be limited to an informal model. Predicted results must be explicitly stated. The model must state the expected result of discretionary access control operations and describe exceptions precisely.

6 The Role of DAC

The last point in the discussion was the proper place for discretionary access control mechanisms in the context of trusted database management. To date, discretionary access controls are not considered a covert channel in a trusted system. The discussion concluded that discretionary access control mechanisms should be treated as a component of object disclosure, and that it should be sufficient to demonstrate that a discretionary access control check has been completed with the expected results. To enforce greater assurance requirements on discretionary access control mechanisms would be excessive due to the inherently untrustworthy nature of discretionary mechanisms.

7 Conclusion

The discretionary access controls in conventional operating systems are not sufficient for database management systems. Additional mechanisms are required to provide a finer granularity of access control and the more flexible mechanisms desired for user-specified data management applications. However, to enforce a high degree of assurance on these mechanisms is not appropriate due to the inherent vulnerabilities of discretionary security mechanisms.

This paper has presented a brief summary of the discussion of discretionary access control mechanisms in database management systems. More research is required to determine the appropriate balance between the mandatory access control mechanisms and the discretionary access control mechanisms in a multilevel trusted database management system.

8 Bibliography

- (1). Department of Defense, Trusted Computer System Evaluation Criteria, DOD-STD-5200.28, December 1985.

Operating System Support of Multilevel Applications

Catherine Meadows

Judith Froscher

Center for Secure Information Technology

Code 5540

Naval Research Laboratory

Washington, DC 20375

The approach to DBMS security currently followed by most of those designing multi-level applications is to build a DBMS on top of a multilevel secure operating system enforcing a separation security policy. Although some trusted code may be implemented as part of the application, in general the application operates within the confines of the operating system policy, and relies on the operating system to enforce that policy. The benefit of this approach is that the application designers do not have to go through the expensive process of designing and implementing multi-level access controls; these are already provided by the operating system. However, the fact that the designers are constrained by the underlying operating system policy means that, in cases where the application policy does not agree with the operating system policy, the operating system controls must be largely circumvented, reimplemented, or used in ways that do not support the operating system security policy. The degree to which they can be used selectively is limited.

This is not surprising, since the needs of a secure DBMS and a secure operating system, although similar, also differ in many ways. It has been pointed out by Spooner [17] that a DBMS deals with (and hence a secure DBMS must protect) logical objects, while an operating system deals with physical objects. Thus the designers of a secure DBMS face the problem of mapping the logical protection objects of the DBMS down to the physical protection objects of the operating system. As would be expected, many features of the DBMS objects do not map very well. These include the smaller granularity and greater interrelatedness of the protected objects, which create needs for such functions as aggregation management, multilevel updates, and integrity constraints across security levels; protection based on information content as well as physical location in the system; and the need for auditing of complex events.

Some of these features can be built independently of the underlying operating system. This is the approach behind what is usually referred to as "TCB subsetting" [15]. One example is the approach taken to discretionary and mandatory security by the SeaView project [4]: mandatory access control is provided by the underlying operating system, while discretionary control is provided by the trusted DBMS. Other examples are provided by the various inference detection and classification guidance systems that have been proposed or are under development [6,9,19]. These

features provide guidance in the enforcement of a security policy, but are not responsible for the implementation of it, and thus can be considered independently of the underlying operating system.

Other functions can be forced to operate within the confines of the operating system policy. An example of this would be a secure DBMS whose discretionary access controls were only applied to relations and enforced by the underlying operating system, thus gaining assurance and ease of implementation at the sacrifice of functionality. Other, more sophisticated examples, are provided by the Hinke-Schaefer model [7] and the SeaView secure DBMS [3], in which multilevel relations are decomposed into single-level relations, which are then protected by the underlying operation system. In this case it is performance, not functionality, that is affected. Although all these solutions have the property that the mapping of DBMS protection objects to operating system protection objects is done at a cost, either to performance or functionality, they do show that DBMS and operating system security policies can be made to correspond at least to a limited extent.

Many features, however, seem to be neither independent of the underlying operating system or implementable within its confines. These include functions that would be managed in part by the user interface, which is usually considered too complex to design and verify according to the requirements of multilevel security. Such functions include trusted display and manipulation of multilevel data and tools for facilitating multi-level updates. Since these involve code that operates over several different levels of data, the underlying secure operating system, whose main purpose is to grant or deny access to objects based on security labels, does little to assist in its management. For the same reason they cannot operate independently of the operating system security policy. Thus the current paradigm of multilevel security requires the designer to model and implement such code as trusted subjects and verify that it does not violate the security policy of the system. Not surprisingly, the task of building such complex functions as trusted subjects is considered beyond the state of the art. Yet the fine granularity and greater interdependence of DBMS objects may make such features necessary. This may become even more the case when complex labeling and aggregation and inference policies are enforced.

Another set of features that may come into conflict with an underlying operating system security policy are those associated with complex data structures such as may be used in an object-oriented DBMS. For example, in [18] Spooner points out the difficulties that may arise when inheritance is used in a secure object-oriented database management system, both when high-level objects inherit from low-level objects, and low-level objects inherit from high-level objects. The classification of data in complex objects may also cause problems. If data in a complex object can only be accessed via some other data in that object, and the data used for access is classified at a higher level than the data being accessed, the security policy of the underlying operating system may be violated. Such a problem was encountered in

the modeling of containers for the NRL Secure Military Message System described by Meadows and Landwehr in [11]. In some cases, a container may be classified at a higher level than the data it contains, but data may be conveniently accessed only via that container. This would be the case, for example, in a Secret message that contains some unclassified paragraphs. Finally, the methods associated with objects in an object-oriented system create the new problem of mapping to subjects in a secure operating system. In a relational DBMS, operations on data are defined in terms of such actions as "read", "insert", and "update", which map in a straightforward way to the actions taken by subjects in a secure operating system. The actions taken by methods in an object-oriented system may be too complex to map to the "read", "write", and "execute" by which subject actions are defined in most secure operating systems.*

It is too early to claim that object-oriented systems cannot be implemented on top of secure operating systems as they exist today. Indeed, work is already being done in developing mappings of the requirements of secure object-oriented DBMSs to policies enforced by secure operating systems [10]. However, the problems we have listed are causes for concern.

It is unrealistic to expect that operating systems can be changed so that the security policy for a multi-level DBMS can be implemented completely by the underlying operating system. The fine granularity and the logical interdependencies of the data managed by the DBMS makes this impossible. However, we believe that it is realistic to expect that secure operating systems can be designed so that they provide greater support for DBMS security, and so that DBMS security requirements, although they may not be enforced by the underlying operating system, at least do not violate its security policy.

The chief cause of conflict appears to be that secure operating systems are built to enforce *separation* of data. Any application that requires sharing of data at different security levels must be treated as a trusted subject; thus the operating system provides little assistance in its management. However, secure DBMSs require, not only separation, but *controlled sharing* of data. Thus we need operating systems to provide better support of controlled sharing.

Some support of controlled sharing can be gained by providing closer management of code that is trusted to write down or across security levels or provide security-relevant functions. Designers of secure operating systems have to some extent been moving in this direction. One example is the assured pipelines of the LOCK operating system [2]. An assured pipeline may allow subjects to write across security levels, but it restricts the paths along which information can travel. In [2] an example is given of a secure labeler implemented as such a pipeline. The pipeline

* We are grateful to Sushil Jajodia for pointing this out to us.

consists of two object types, labeled and unlabeled data, and two modules, the labeler and the output module. The labeler accepts unlabeled data only and produces labeled data that cannot be modified, while the output module will accept labeled data only. More complex pipelines have been used to implement various security-relevant parts of the LOCK Data Views system [5], a secure DBMS intended to run on top of the LOCK operating system. These include pipelines that handle responses to queries and updates, and manage metadata.

A more specialized example of operating systems providing closer management of trusted code is the auditing subsystem provided by the MITRE Compartmented Mode Workstation (CMW) [13]. The auditing subsystem not only provides an auditing capability for the CMW, but also manages and stores audit trails of applications (such as DBMSs) that are trusted to do their own auditing.

A degree of control similar to that of LOCK may be achieved in the GEMSOS operating system, although by a more circuitous route. In GEMSOS [14], subjects are given two labels, one which specifies the highest level at which it can read, and the other which specifies the lowest level at which it can write, with untrusted subjects having read labels equal to write labels, and trusted subjects having read labels that strictly dominate the write labels.* This allows one to give trusted code permission to write across no more security levels than is necessary for it to do its job. By creating a large range of security and integrity levels, one can use such double labels to achieve a fine grain of control. For example, Lee [8] and Shockley [16] show how such labels can be used to enforce the Clark/Wilson integrity policy. Note, however, that this degree of control is achieved at the cost of using secrecy and Biba integrity labels for purposes other than they were intended. This means that some extraneous features are included (for example, the lattice hierarchy) that may make evaluation more difficult.

Another area in which operating systems can provide better support for database security is in the provision of primitive trusted functions that can be used to manage sharing of data at different security levels. Some of the functions that could be provided and that would have an impact on security are discussed by Spooner in [17]. These include process control, memory management, file management, correct physical I/O, buffer management, recovery and transaction management, interrupt handling, and network services. These functions will often be responsible for managing processes or data at different security levels. Some work, but not enough, has been done on the provision of such functions in secure operating systems. For example, in [7] Hinke and Schaefer provide algorithms for secure process synchronization. These are presented in the context of a secure DBMS, but there is no reason that they could not be provided by the operating system. For example, GEMSOS

* A similar labeling system is introduced by Bell in [1].

provides the eventcounts and sequencers that could be used to implement these functions, although it does not provide the functions themselves. As another example [12], Parenty discusses the means by which multilevel inter-process communication via shared files could be implemented in a multi-level Unix system. Such a function would be needed by a secure DBMS that builds a multilevel data structure out of single-level objects, but is not provided by most secure Unix systems under development.

The reason such trusted functions are not provided by most secure operating systems is not because we do not know how to implement them. For example, the techniques presented in [7] have been available for the last fifteen years. Rather, the lack of such functions seems to be due to the fact that their inclusion would expand the TCB beyond the limits of what is considered possible to evaluate. Thus what is required is, not only more research into the design of trusted functions and the management of trusted code, but more research into means of designing and evaluating TCBs so that trusted functions can be inserted without making the evaluation process intractable.

At present, all evaluated products enforce essentially the same security policy, and none support a methodology for adding and evaluating trusted functions. But attempts to produce trusted applications, SDBMSs in particular, have revealed, both that the enforcement of a simple, generic security policy does not satisfy the more complex security requirements of most applications, and that trusted functions are required for an application to do its job. Thus, what we appear to need is not a simple operating system that rigorously enforces a single security policy, but a system that provides a set of trusted functions that can be combined and extended so that the application builder can verify that the result enforces the security policy of the application. In order to achieve this goal, we need ways of implementing and evaluating operating system components that could be combined to form a TCB that enforces an application system security policy. Additionally, a sound theory must be developed to allow designers to rigorously demonstrate that the system TCB built out of those components enforces the application security policy. In other words, what we need is not only TCB subsets, but TCB "slices".

In conclusion, we believe that not all security requirements of a multilevel database management system can be satisfied by relying on the security mechanisms of an underlying operating system. The nature of the objects protected differs too widely. However, we do believe that secure operating systems can provide more support than they do now, both by providing more trusted functions for the management of data at different security levels, and by providing more operating system management and control of trusted functions supplied by the application designer. Moreover, the implementation of these features is currently within the state of the art; the area in which research still needs to be done is in how they can be independently implemented and evaluated so that they can be composed to form a TCB that

demonstrably enforces an application-dependent security policy.

References

1. D. E. Bell, "Secure Computer Systems: A Network Interpretation," in *Proceedings of the Second Aerospace Computer Security Conference: Protecting Intellectual Property*, pp. 32-39, AIAA, Washington, DC, 1986.
2. W. E. Boebert and R. Y. Kain, "A Practical Alternative to Hierarchical Integrity Policies," in *Proceedings of the 8th Annual National Computer Security Conference*, pp. 18-27, NBS/NCSC, 1985.
3. D. E. Denning, T. F. Lunt, R. R. Schell, M. Heckman, and W. Shockley, "A Multilevel Relational Data Model," in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pp. 220-234, IEEE Computer Society Press, Washington, DC, 1987.
4. D. E. Denning, T. F. Lunt, R. R. Schell, W. R. Shockley, and M. Heckman, "The SeaView Security Model," in *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pp. 218-233, IEEE Computer Society Press, Washington, DC, 1988.
5. J. T. Haig., P. D. Stachour, P. A. Dwyer, E. O. Onuegbe, and B. M. Thuraishingham, "Secure Distributed Data Views: Final Technical Report for a Database Management System, Volume 5: Implementation Specification," Report A005 on RADC contract F30602-86-C-0003, Honeywell Systems and Research Center, May 1989.
6. T. Hinke, "Inference Aggregation Detection in Database Management Systems," in *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pp. 96-106, IEEE Computer Society Press, Washington, DC.
7. T. H. Hinke and M. Schaefer, "Secure Data Management System," RADC-TR-75-266, Rome Air Development Center, November 1975.
8. T. M. P. Lee, "Using Mandatory Integrity to Enforce "Commercial" Security," in *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, pp. 140-146, IEEE Computer Society Press, Washington, DC, 1988.
9. T. F. Lunt, R. R. Schell, W. R. Shockley, and D. Warren, "Toward a Multilevel Relational Data Language," in *Fourth Aerospace Computer Security Applications Conference*, pp. 72-79, IEEE Computer Society, Washington, DC, 1988.
10. T. F. Lunt, "Secure Distributed Data Views: Identification of Deficiencies and Directions for Future Research." A007: Final Report. Vol. 4. SRI International, Menlo Park, CA, January 31, 1989.
11. C. A. Meadows and C. E. Landwehr, "Designing a Trusted Application Using an Object-Oriented Data Model," in *Research Directions in Database Security*, ed. T. F. Lunt, to appear.

12. T. J. Parenty, "The Incorporation of Multi-Level IPC into UNIX," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pp. 94-99, IEEE Computer Society Press, Washington, DC, 1989.
13. J. Picciotto, "The Design of an Effective Auditing Subsystem," in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pp. 13-22, IEEE Computer Society Press, Washington, DC, 1987.
14. R. R. Schell, T. F. Tao, and M. Heckman, "Designing the GEMSOS Kernel for Security and Performance," in *Proceedings of the 8th National Computer Security Conference*, pp. 108-119, NBS/NCSC, 1985.
15. W. R. Shockley and R. R. Schell, "TCB Subsets for Incremental Evaluation," in *Proceedings of the Third Aerospace Computer Security Conference: Applying Technology to Systems*, pp. 131-139, AIAA, 1987.
16. W. R. Shockley, "Implementing the Clark/Wilson Integrity Policy Using Current Technology," in *Proceedings of the 11th National Computer Security Conference*, pp. 29-37, NBS/NCSC, Baltimore, MD, 1988.
17. D. L. Spooner, "Relationships Between Database Security and Operating System Security," in *Database Security: Status and Prospects*, ed. C. E. Landwehr, pp. 149-158, North-Holland, Amsterdam, 1988.
18. D. L. Spooner, "The Impact of Inheritance on Security in Object-Oriented Database Systems," in *Database Security II, Status and Prospects*, ed. C. E. Landwehr, pp. 141-150, North-Holland, Amsterdam, 1989.
19. M. B. Thuraisingham, "Security Checking in Relational Database Management Systems Augmented with Inference Engines," *Computers & Security*, vol. 6, no. 6, pp. 479-492, December 1987.

AN INTERIM REPORT ON THE DEVELOPMENT OF SECURE DATABASE
PROTOTYPES AT THE NATIONAL COMPUTER SECURITY CENTER

John R. Campbell
National Computer Security Center
Office of Research and Development
98000 Savage Road
Fort George G. Meade, Maryland 20755-6000
301-859-4488

INTRODUCTION

This paper describes work that is being done by the National Computer Security Center with two database vendors to build a series of secure, full-functioned, commercial quality, database management system (dbms) prototypes. The security being added is consistent with the Trusted Computer System Evaluation Criteria (TCSEC or "Orange Book") [3]. A goal is to add Orange Book security to existing, widely-used state-of-art database management systems. A high level of integrity is also required. Specifically covered here are results to-date obtained from the initial task of the projects, the development of two secure database management system prototypes at the class C2 (Controlled Discretionary Access Security Level).

WHY THESE PROTOTYPES

The Trusted Database Interpretation (TDI) [4] is an interpretation of the TCSEC for database management systems much in the same way as the recently published Trusted Network Interpretation (TNI) [8] described the TCSEC relationship to networks. The prototypes provide research to support the development of the TDI. Systems ratable at the B2 and A1 levels had been built during the writing of the TCSEC to show that the TCSEC was implementable. Similarly, these prototypes are being developed to demonstrate that the requirements of the TDI are implementable. As was pointed out in a February, 1989 Software Technology Service Newsletter, the building of the prototypes allows us to develop the TDI in a real, live situation. Further, the DOD-vendor community at large will have a tested criterion instead of a policy that is just a paper standard [7].

The second reason why these prototypes are being built is to show that not only are the interpretation of the TCSEC requirements implementable but that they are realistic for full-blown commercial quality systems. Many papers have been written

and basic proof-of-concept prototypes have been built. However, there still remains a number of unanswered questions. Can you, for example, install security in a full database management system and still maintain adequate performance and high integrity?

System structure affects organization structure. How should organizations be restructured so that secure database management systems, with added functionality, can be effectively and advantageously used by the organization? What new roles must be created? What must be audited? The likelihood of being overwhelmed with huge amounts of audit information is certain, unless the audit system is carefully planned and unless adequate tools are provided to examine and reduce this information. Dr. Carl Landwehr indicated in [5] that the most secure database system in the world will not be effective in practice, if it places unreasonable constraints or demands upon the systems' operators and users or if it unacceptably degrades system performance.

The third reason why these prototypes are being built is that NSA, DoD and the Intelligence Community use many commercial database management systems and would greatly benefit from being able to purchase commercial off-the-shelf secure systems. The alternative is to support the expensive process of inhouse system development and maintenance of one-of-a-kind applications. Also, commercial systems are built for user ease of use. Specialized systems, while doing the job, are often difficult to use and require unique training.

DATABASE MACHINES AND HOST-BASED DBMSs

The prototypes are broken into two type of architectures, namely, database machines and host-based database management systems. In each, prototypes are or will be built which we believe are dbms implementations of the C2, B1, B3 and A1 levels of the TCSEC. In addition to these security requirements, we are addressing data integrity, including referential integrity, inference, aggregation and denial of service. We are looking at what the Strategic Defense Initiative calls "Security *", that is, control of data access, data integrity and system availability.

The first prototype approach using database machines is of interest because the physical isolation lends itself to enhanced database security. These are computers are dedicated to database management system functions. The disks containing the database

are hung off these computers and host computers access the database computer or database machine to obtain data. Only identification, authentication and query information need to be regularly sent from a host to the database machine and only answers need to be returned. Because of this physical isolation, the operating system of a host does not have the opportunity to attack the database management system as it could if both database and operating system coexisted on one computer. Also, since the database machine has only the database function, its operating system assumes fewer functions and can therefore be smaller and more efficient. Large amounts of data can be handled as some machines have parallel architectures. Some are fault tolerant. Some are modular, which means that their capacity and performance can easily be expanded and increased, without the need for changing system software. In the database machine we selected, up to 1024 processors can be connected together for linear performance. Whether six or 1024 are used, the same software is used, thus giving the user great flexibility.

The second experimental approach, host based database management systems is also interesting. These are the general purpose computers, the IBM PC/AT, the VAX or the IBM 3090. These database systems are widely used, with over 90% of mainframes, and most mini and microcomputers using them. Some of these database systems are distributed; data can be stored in multiple locations and yet appear to be local to the user. Also, some database systems can use the power of parallel processors and other advanced architectures.

STATUS

Work on both prototypes is underway. Teradata, Corp., with the National Computer Security Center, is using the database machine approach to develop "C2" and "B1" level prototypes. Changes have been made in the areas of authentication, audit, system I/O privileges and documentation. A "C2" prototype was delivered in August of 1988 and, after it passed extensive testing, was accepted as a contract deliverable. Teradata has elected to incorporate these security modifications into its standard release 4.0 of their commercial product. They have scheduled release of this product sometime this summer. Deficiencies preventing a "B1" level of trust have been defined and research is now being conducted to propose how these deficiencies can be corrected.

Oracle Corporation is using the host-based database management system approach to develop both "C2" and "B1"

prototypes. A deficiencies report enumerating what must be changed to bring the DBMS into compliance with the TCSEC and TDI for the first prototype has been delivered. This prototype, at the "C2" level, is expected in May, 1989 and will be on a VAX using the VMS operating system. A second discretionary prototype is scheduled for delivery in July, 1989 and will run on the GEMSOS operating system on a Gemini Computer. Developing two prototypes gives us the opportunity to explore how differing hardware architectures and operating systems affect secure dbms development. Also, both will be used as initial steps in the development of the "B1" prototypes. The VMS prototype will be used as a basis for SE/VMS prototype and the "C2" GEMSOS prototype will be used as a starting point for the "B1" GEMSOS prototype. The first "B1" prototype is expected to be delivered will be delivered in April of 1990, and will be integrated into SE/VMS. The final mandatory prototype using GEMSOS will be delivered in June of 1990.

We believe that the Oracle "B1" prototype will actually be in the B1-B3 range, if the current balanced assurance theory is accepted in the TDI and on the further study of the structure of the Oracle code. In any case, in all prototypes we are developing, the underlying operating system must be at least the same level of certification as the database system.

OBSERVATIONS

Deficiencies

Current commercial versions of both DBMS approaches were examined to improve security. A number of potential security problems were found in each and these have been or are being corrected by the vendors. It is interesting to note that both current commercial versions had security deficiencies due to system routines. These routines, because of ease-of-use or performance considerations, bypassed security procedures [1,2]. One routine was a bulk data loading routine. The second routine enabled a system programmer to fix a system that had crashed. It is suspected that most commercial DBMSs have such commands and therefore these commands would be easy tools to compromise data stored in the database management systems.

SQL Modifications

SQL, the language, is used by both vendors to access the databases. Both the number of SQL keywords and SQL functionality

was necessarily expanded to support the secure versions. SQL command extensions that support group access controls, that support the ability to selectively audit actions (based on the identity of the user performing the action), and that support inter-record dependency constraints, (i.e., referential integrity), have been added.

Teradata introduced the BEGIN LOGGING and END LOGGING commands to permit the system administrator to control auditable actions. The "Action-name" or "ALL" commands specify the type of accesses that will be logged and gives the system administrator control over the depth of auditing.. These are the same functions on which an access right can be granted. The action-names are given for completeness:

CHECKPOINT	DROP DATABASE	INSERT
CREATE DATABASE	DROP MACRO	MACRO
CREATE MACRO	DROP TABLE	RESTORE
CREATE TABLE	DROP USER	SELECT
CREATE USER	DROP VIEW	TABLE
CREATE VIEW	DUMP	UPDATE
DATABASE	EXECUTE	USER
DELETE	GRANT	VIEW

A "WITH TEXT" option saves the text of the command which caused the audit log entry. FIRST, LAST and EACH commands specify the frequency with which log entries will be made. On an END LOGGING statement, if the object name is a database or user and is followed by ".*", all logging rules on all objects in the database or user will be modified as specified.

To restrict a user to a subset of the hosts connected to the database machine, new SQL statements were added:

GRANT LOGON ON ALL (or Hostid,Hostid,...) TO, and
REVOKE LOGON ON ALL (or Hostid,Hostid,...) TO

An optional WITH NULL PASSWORD phrase is available. This permits logons without a password string from a specified community (host).

Oracle modifications include added support for group access controls by establishing the "role" mechanism. Privileges can be granted to roles and then roles can be granted to users or other roles. SQL statements are provided to CREATE and DROP roles. Two forms of GRANT commands are specified: one to assign privileges to roles, the other to assign roles to individuals or other roles. Two REVOKE commands are provided to revoke assignments made by the GRANT commands. Finally, additions are made to the AUDIT commands to permit auditing of the CREATE ROLE, DROP ROLE, GRANT and REVOKE commands. The AUDIT commands

themselves may be audited and NOAUDIT used to turn off the audit function.

A new AUDIT command (Form III) was created to support auditing of all actions by individual users. This command also supports the auditing of specific actions by individuals, for example, only GRANT and REVOKE commands. This feature reduces the amount of audit data collected. A NOAUDIT (Form III) command is used to partially or completely reverse the effect of a previous AUDIT (Form III) command.

Inter-record Dependency Constraints are supported by implementing referential integrity as defined in the proposed ANSI SQL standard x3.135.1-198s, "Database Language SQL - Addendum I". A CREATE TABLE command is used to define the table constraints that specify referential integrity.

One of the requirements of Audit is the need to move data from, say, disk to tape, when the disk becomes full. When this happens we must record the fact that it has happened. Therefore, to satisfy the need to audit the deletion of an audit trail, a new SQL command called PURGE AUDIT TRAIL that users with DBA privilege can use to remove data from the audit trail, was added. This command copies records from the audit trail into another table, deletes the records from the audit trail, and then adds a new audit trail entry to record that this command was evoked. Thus the act of deleting the audit trail is of itself an auditable action and is logged (audited) after successful deletion of the audit trail. Assurance that these controls work must be at a sufficiently high level.

It is suspected that multilevel control will require a substantial expansion of SQL keywords and/or the functionality listed here.

Audit

Even where the object is a record, the number of audit records generated under mandatory access control can be very large. Therefore the use of summary records is suggested for each subject/table/hierarchical level/compartment combination.

Discretionary Access Control

As described above, Oracle has proposed the use of "roles" in the enforcement of Discretionary Access Control (DAC). A "role" has assigned to it a set of capabilities such as read and/or write access to one or more tables. Users are then assigned to one or more roles.

Development Strategy

Development-Release Cycles

Both vendors have integrated security developments into their regular product release cycles. Security enhancements are targeted for specific releases. One vendor schedules all work on a particular module, whether security or not, at a particular time and by a particular team.

Subsetting v. Monolith Design Approaches

Teradata is using a monolith approach where database management system and operating system will be evaluated as one entity. For a database machine, where the operating system is designed to only support the database system, and therefore can be of minimal size, this approach is prudent.

Oracle is using a TCB subsets approach, where access control is shared between the operating and database systems. Specifically, the operating system is providing mandatory access control and both operating and database systems are providing discretionary access control. The subsetting approach is reasonable a host-based system design. The underlying operating system, once certified, does not have to be recertified until it is "changed". Oracle interfaces with many systems. It would be very costly, both in time and other resources, for a dbms vendor to obtain certification and recertification for the many dbms/operating system combinations possible.

Performance

Performance of the first Teradata prototype, when contrasted with the former standard version, appears to be little affected except for audit. If all auditing mechanisms are turned on, and in a worst case scenario consisting primarily of read accesses, the overhead due to auditing could be high. The use of summary audit records, especially under mandatory access control, to decrease this cost, has been suggested. Using this procedure, one record would be generated for each subject and relation/hierarchical level/compartiment combination of an object. This record would keep an access-attempted count.

Future

Both projects are working on "B1"-level prototypes. Future research will include the distributed database environment in which DBMSs must "support local ownership and management of data; not require users to know where the data is physically stored; not require users to know if data relations are fragmented or

duplicated somewhere in the network; support distributed query processing and distributed transaction management; permit hardware, operating system, and network independence, and... permit transparent access of data between dissimilar DBMSs on a network. This last rule means vendors first would have to agree on a standard version of SQL " [6]. To do all this, securely, and with data integrity, is a challenging, yet interesting, task.

SUMMARY

This paper briefly describes the initial work being done to secure two series of database management system prototypes. Why full-functioned, commercial quality DBMSs, the architecture of the DBMSs being examined, development plans, and progress and observations made are discussed. A likely place to look for problems, SQL changes being made for the "C2" versions and suggested modifications to the audit record, are described. Development strategies, performance observations and future plans were then outlined. A great deal of progress has been made, future progress is likely and success is expected.

REFERENCES

1. "CDRL A013: The Discretionary Modification Plan and Cost Assessment", Contract #MDA904-88-C-6009.
2. "Deliverable A006, The Final Security Modification Plan/Pre-Assessment Report", Contract #MDA904-87-C-6009.
3. Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, December 1985.
4. Draft "Trusted DBMS Interpretation of the DOD Trusted Computer System Evaluation Criteria", National Computer Security Center, November 18, 1988.
5. Landwehr, C.E., "Database Security: Where Are We?", Database Security: Status and Prospects, C. E. Landwehr, ed., North-Holland, 1988.
6. "Look Ahead", Datamation, March 1, 1989, p. 11.
7. Software Technology Service, February, 1989.
8. Trusted Network Interpretation of the Trusted Computer

System Evaluation Criteria, NCSC-TG-005, July 31, 1987.

Version 890406

SOME REMARKS ON INFERENCE CONTROLLERS

T. Y. LIN

Department of Computer Science
California State University, Northridge
Northridge, California 91330

1. INTRODUCTION

In this paper we examine three approaches to the Inference and Aggregation problems in multilevel relational database system. The first approach that we will examine is Morgenstern's approach [Morg87], [Morg88]; call it Method 1. The second approach is Ford Aerospace's proposal; call it Method 2 [Buck89a-c]. The third is studied by present author; call it Method 3 [Lin89a-b].

2. METHOD 1

2.1 Goal

The theory is designed to protect a rather general inference problems on multilevel database:

- (1) Inference as logical process of proving and deriving conclusions from some given facts and rules.
- (2) Inference in the information theoretical sense.

2.2 Quantitative Inference Model

2.2.1 Universe of Discourse:

All entities, attributes, relationships, constraints, and rules that are of interests. The information involved are not only from the stored database but also include user's knowledge that may not be stored in the database. The traditional database schema is a subset of this universe [Morg87].

2.2.2. Inference Function INFER is expressed by entropy. The function is denoted by $\text{INFER}(x \text{ ---} \rightarrow y)$, where x is an aggregate and y represents information that can be inferred from x .

2.2.3. Core and SOI: A collection of primitive data is called core and all the information that can be inferred from the core is called SOI, Sphere of Inference.

2.2.4. Inference Channel: An inference channel is an aggregate of core from which an inference is made. An inference channel exits

if the security classifications of all objects in the core are less than or non-comparable with that of SOI.

2.2.5. The solutions:

1. If the inference channel consists of one object, then simply raise the classification level of this object.

2. The data object within this aggregate would be released in a controlled manner.

3. Introduce the noisy to the data that is released from this aggregates.

4. Monitor the channel.

2.3 Discussions

1. The notion of SOI in 2.2.3 is too general. For example, let core be the axioms of Euclidean Geometry, then some mathematicians may still work on SOI(core) [Any theorems in geometry is in SOI]. The actual determination of SOI may take an impractical length of time. However, in [Morg87] but not in [Morg88], Morgenstern did provide an algorithm for finding SOI at schema level. We believe that some limit on the territory of inference should be included in the model. In this respect [Morg87] (presume that schema level is the only territory of SOI? or some generalization) is better than [Morg88]. Basically this is theoretical model. It is more interested in the existence problem of inference channels than the actual identification and removal of such channels.

2. In correcting an inference channel Method 1 may raise the classification level of some objects. This may send "ripples across" the whole model. This problem is mentioned [Morg98, pp. 245], but no solution is offered in the model.

3. METHOD 2

This approach is a modification of Method 1 with emphasis on the real world problem. The multilevel database is assumed to be the SeaView developed by SRI [Denn86a-b], [Denn87a-c], [Denn88a-b], [Lunt88a-c], [Lunt89].

3.1 Goal

The system is designed to protect following types of inferences:

- (1) Related Data and
- (2) Indirect Access

According to Hinke, there are four types of inferences: missing

data, vanishing data, indirect access and related data [Hink88]. Since SeaView has protection on first two problems, so Method 2 focus on the last two inferences.

3.2 Probabilistic Knowledge Model

3.2.1 Universe of Discourse:

- (1) Information store in the multilevel database.
- (2) Information that is not presented in the database:
 - (2.1) Knowledge that an unclassified user possesses.
 - (2.2) Knowledge that is to be protected [Bucz89c,pp.20].

3.2.2. Semantic Data Model:

The database is represented by Semantic Model. It is used to find the indirect access inference and to assist in building the probabilistic inference networks by providing ranges for the core parameters.

3.2.3 Probabilistic Inference Networks

The development of the network consists of two phases:

- (1) Experts identify information that is to be protected: The security classification policy is elaborated by top-down fashion. The top-level categories are used as starting point to identify parameters relevant to the security classification policy.
- (2) The system update the probability based on probabilities supplied by experts and inference propagation equation.

3.2.4 Knowledge Base System

(1) Knowledge base:

- (1.1) A basic semantic model to represent the database.
- (1.2) Probability inference networks are used to relate the objects of the underlying database to object to be protected under the system security policy.

(2) Rule base:

- Rules for knowledge acquisition and storage of knowledge
- Rules for detecting indirect access inferences
- Rules for detecting related data inferences
- Rules for communicating results back to user

(3) Inference Engine

3.3 Discussions

1. The idea is excellent, however, the proposed framework do not guarantee that probability inference network always construc-

table. Current theories on probability inference networks usually needs some restriction or conditions on the structure of networks. For example, J. Pearls developed his network on acyclic networks [Pear86]. PROSPECTOR's assumption is basically equivalent to that the network is a tree [DuHaNi76], [YaVaPeHoKe86]. Since this project is an application, the special circumstance surrounding the particular application may imply that all probability inference networks are constructable; however, no such proof is supplied. The paper exhibits such constructions by some examples and, however, has no general theorems to support such general constructions.

This presentation is just an opposite to Method 1. The scheme is very practical, however, the exposition does not provide a solid theoretical foundation on its methodology. Without a theoretical foundation, each project has to prove its own existence theorem on the probabilistic inference networks; constructability is not apparent to the readers.

2. As remarked in Method 1, reclassification of security level of an object may have ripple effect through whole the networks; no discussion is provided for this side effect.

3. If Method 2 has no defects, it actually can solve more than the stated goals: indirect access and related data. In fact, it is rather general, it can solve the general inference problems.

4. The universe of discourse is basically the same as Method 1. However, it may worthwhile to remark here that one should not be too ambitious about user's knowledge, for example, his inferred knowledge may be more than what the system can infer.

4. METHOD 3

4.1 Goal

The system is designed to

- (4.1) Detect Aggregation and
- (4.2) Protect the Inference on Aggregation

Note the aggregation here is more than usual aggregation; it is an algebraic aggregation. Hence it includes some indirect access and related data channels. For some systems security algebra may include "security join" which related to indirect access and some related data channels.

4.2 Aggregation

For a multilevel database, every data in the database has been assigned a security classification which reflects the real world

security classification; such security class is called real-world security class. Some data are derived data, some are primitive data (atomic facts). We will assign a second security class to these data; they are called algebraic security class. First, we will assign the algebraic security class of primitive data the real-world security class. Then we will assign the algebraic security class of a derived data the class that is derived from the security algebra. Note that every derived data is defined by an expression in the relational algebra, hence its security class can be computed by the corresponding expression in the security algebra [Lin89b].

Let us set up some notations. Let E a derived data (that represents a view or an entity). Then, E can be expressed by

$$E = f(E_1, \dots, E_k)$$

where E_1, E_2, \dots, E_k are primitive data. If the real world security class of E is not agree with the algebraic security class computed by security algebra, then the pair $(E; E_1, E_2, \dots, E_k)$ is an aggregation (see [Lin89b]). An aggregation is called simple, if none of its subexpressions are aggregations.

4.3 Solutions

4.3.1 Simple Aggregation

The solutions of such aggregations have been well discussed in the [Dill88], [Hink88], [Lunt89a], [Sayd87]. Basically the solutions for simple aggregations can be classified into two types [Lin89]:

- (1) The original aggregation problem

$$E = f(E_1, \dots, E_k)$$

can be transformed to a new expression

$$E = g(F_1, F_2, \dots, F_h)$$

so that the pair $E; F_1, F_2, \dots, F_h$ is not an aggregation. Such simple aggregation is called a **fake aggregation**. Examples are "context dependent" aggregation which can be removed by appropriate database design, see [Lunt89].

The primitive data E_1, E_2, \dots, E_k in the original database are removed and replaced by F_1, F_2, \dots, F_h . In other words, part of the core is changed -- The new set of primitive data which includes $F_i, i=1, 2, \dots, h$ but has no $E_i, i=1, \dots, k$ is a result of intelligent data design [Lunt89].

- (2) This is the type of simple aggregation that no transformation

exists; it is a true aggregation problem.

Let $E = f(E_1, E_2, \dots, E_k)$ be a true simple aggregation. Then the solution of E is to

- (2.1) Add the database a new primitive fact E
- (2.2) Remove E_1, E_2, \dots, E_k from the original primitive data set
- (2.3) The release of any of E_1, E_2, \dots, E_k are handled by a special procedure which is processed by SSO or a special mechanism.

4.3.2 General Solution

The general aggregation is solved by induction on the "level" of aggregation [Lin89b].

5. COMPARISONS

5.1 Method 1:

- (1) Theory: A complete general theory.
- (2) Application: Impractical in applications.
- (3) Scope: Very General.
- (4) Solution: Inference channels are detected, however, no solution is guaranteed.

5.2 Method 2:

- (1) Theory: An incomplete theory. The constructability of probabilistic inference network for its applications are demonstrated by examples, but not in general theory.
- (2) Application: If the probability inference networks exists, then it is a beautiful frameworks for applications.
- (3) Scope: Together with SeaView, solve all the inference channels identified by Hinke [Hink88]. In fact Method 2 can identify more general inference channels than the indirect access and related data channels.
- (4) Solution: Inference channels are detected, however, no solution is guaranteed.

5.3 Method 3:

- (1) Theory: The theory provides a complete solution to the algebraic aggregation. Algebraic aggregation includes the some indirect access [Lin89b, Example 2.3] and related data channels [Lin89b, Example 5.6]. The theory has potential to solve the two inference channels completely; more work needs to be done.

- (2) Application: Applicable to any multilevel database system.
- (3) Scope: It is more general than classical aggregation and inference problem. The algebraic aggregation includes some indirect access and related data. It has a highly potential to solve all the inference channels identified by [Hink88].
- (4) Solution: A complete solution for inference on algebraic aggregation is presented.

6. CONCLUSIONS

Defending against inference attacks is one of the most fascinating area of computer security. However, one should realize its limit. For example, many mathematical theorems, which do not employ mathematical induction in the proofs, are inferred from the axioms of that branch of mathematics. So, it is unreasonable to build a defense against such type of inference. A reasonable approach is to identify some "immediate/easy inference" problems such as identified by [Hink88] and then build a defense system against such inference attack. Method 1 is too general. Method 2 has identified their limited targets, however, the actual mechanism (if it works) is applicable to a fairly general type of inference problems which is beyond their stated targets; so it might be too general. On the other hands, the idea of merging the uncertainty reasoning with Database Security itself is an exciting idea; it will attract a lots of attention. Method 3 limits itself to a special type of inference problem, so it has a strong result. Note that the mechanism is built on algebraic aggregation (it is more general than usual aggregation problem). Method 3 with proper generalization, such as "Semantic Aggregation" has a high potential to give a complete solution to the related data and indirect inference problems; it seems one of the most effective approaches on the inference channels identified by [Hink88].

ACKNOWLEDGE

This research was supported by the U. S. Air Force, Rome Air Development Center (RADC), under SRI's Subcontract C-12537. I am indebted to both RADC and SRI for making this work possible.

REFERENCES

- [Buck89a] L. J. Buczkowski, E. L. Ferry, and D. H. Lee. Database Inference Controller -- Draft Top-level design, July 1989, Ford Aerospace Corp.
- [Buck89b] L. J. Buczkowski. Database Inference Controller, Proceedings of IFIP WG11.3 Workshop on Database Security Septem-

ber 5-7, 1989

[Buck89c] L. J. Buczkowski, E. L. Perry, and D. H. Lee. Database Inference Controller--Final Technical Report, September 1989, Ford Aerospace Corp.

[Denn76] D. E. Denning. A Lattice Model of Secure Information Flow, Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 236 - 243.

[Denn86a] D. E. Denning. The Inference Problem in Multilevel Database systems, In the Proceeding of the National Computer Security Center Invitational Workshop on Database Management Security, June 1986.

[Denn86b] D.E. Denning, S.G. Akl, M. Heckman. T.F. Lunt, M. Morgenstern, P.G. Neumann, and R.R. Schell. View for Multilevel Database Security, Proc. IEEE Symposium on Security and Privacy, 1986.

[Denn87a] D.E. Denning, T.F. Lunt, R.R. Schell, M. Heckman and W.R. Shockley, A Multilevel Relational Data Model, Proc. 1987 IEEE Symposium on Security and Privacy, 1987.

[Denn87b] D.E. Denning, T.F. Lunt, R.R. Schell, M. Heckman and W.R. Shockley, "The SeaView Formal Security Policy Model", Computer Science Laboratory, SRI International, July, 1987.

[Denn88a] D.E. Denning, T.F. Lunt, R.R. Schell, W.R. Shockley, M. Heckman, The SeaView Security Model, Proc. 1988 IEEE Symposium on Security and Privacy, 1988.

[Denn88b] D.E. Denning, T.F. Lunt, P.G. Neumann, R.R. Schell, M. Heckman and W.R. Shockley, "Security Policy and Policy Interpretation for a Class A1 Multilevel Secure Relational Database System", Computer Science Laboratory, SRI International, Aug. 1988.

[Dill88] Dillaway et al. Security Policy Extensions for a Database Management System. Interim report A002. Honeywell Systems Research Center and Corporate System Development Division, May 1988.

[DuHaNi76] R. O. Duda, P. E. Hart and N. L. Nilsson. Subjective Bayesian methods for a rule-base inference system. In Proceeding of 1976 National Computer conference, vol 45, pp.1075-1082, 1976

[Hink88] T. H. Hinke. Inference Aggregation Detection in Database management systems, In Proceedings of the 1988 IEEE Symposium on Security and Privacy, April 1988.

[Lin89a] T. Y. Lin, L. Kerschberg and R. Trueblood. Security Al-

gebra and Formal Models, Proceedings of IFIP WG11.3 Workshop on Database Security September 5-7, 1989

[Lin89b] T. Y. Lin, Commuative Security Algebra and Aggregation, Proceedings of Second RADC Workshop on Database Security, 1989

[Lunt88a] T.F. Lunt, R.A. Whitehurst, The SeaView Formal Top Level Specifications, Computer Science Laboratory, SRI International, Feb. 1988.

[Lunt88b] T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman and W. R. Shockley, Element-Level Classification with AI Assurance, Computers & Security, Vol. 7, No. 1, 1988, pages 73-82.

[Lunt88c] T. F. Lunt, R. R. Schell, W. R. Shockley, M. Heckman, D. Warren, A Near-Term Design for the SeaView Multilevel Database System, Proceedings 1988 IEEE Symposium on Security and Privacy, 1988.

[Lunt89] T. F. Lunt. Aggregation and Inference: Facts and Fallacies, Proceedings 1989 IEEE Symposium on Security and Privacy, 1989.

[Morg87] Mathew Morgenstern. Security and Inference in Multilevel Database and Knowledge-base systems, ACM International conference on Management of Data (SIGMOD-87), May 1987

[Morg88] Mathew Morgenstern. Controlling Logical Inference in Multilevel Database Systems, Proceedings 1988 IEEE Symposium on Security and Privacy, 1988.

[Pear86] Judea Pearl. Fusion, Propagation, and Structuring in Belief Networks, Artificial Intelligence 29(1986), 241-288

[Sayd87] O. S. Saydjari, J. M. Beckman and J. R. Leaman, Locking Computers Securely, Proc. 10th National Computer Security Conference, Sept. 1987, National Bureau of Standards/ National Computer Security Center, pp. 129-141, 1987. Advances in Computer System Security, Vol. 3, edited by Rein Turn, pp. 207- 219, 1988.

[Yag86] R. M. Yadrick, D. S. Vaughan, B. M. Perrin, P. D. Holden, K. G. Kempf. Evaluation of uncertain inference models I: PROSPECTOR. IN Lemmer and L. N. Kanal Uncertainty in Artificial Intelligence, pp.333-338, Elsevier Science Publishers, 1988.

COMMUTATIVE SECURITY ALGEBRA AND AGGREGATION

T. Y. Lin

Department of Computer Science
California State University
Northridge, California 91330

ABSTRACT

One of the difficulty in aggregation problems is that there is no proper framework to express the problems properly. The security algebra, introduced by his colleagues and present author, provides a succinct theory to express the aggregation problems in database security. Aggregation has been phrased vaguely as a collection (set theoretical?) of atomic facts. In terms of security algebras, aggregation is defined to be an entity described by an algebraic expression (in the relational algebra) of individual data. The theory indicates that the efforts in various projects have been devoted to solve only the "simple" types of aggregation problem; simple is used in the sense of structure theory of algebra. Some complex aggregations are constructed for illustrations. In this paper, the aggregation problems (simple and complex) are formalized. In terms of the formalism, a complete solution of the aggregation problems are presented. In this paper, only the content dimension is addressed. Moreover, non-commutative aggregation problems are not discussed (the usual aggregation problems are all belonged to the commutative type aggregation problems. See forthcoming papers.

1. INTRODUCTION

Inference in general and aggregation in particular have attracted considerable interests [Denn86a-b], [Denn87a-c], [Denn88a-b] [Dill87], [Hink88], [Lunt89a], [Stac88], [SuOz87], [SuOz89]. One of the difficulty is that there is no proper framework to express the problems properly. For example, aggregates have been referred vaguely as a collection; is it set theoretical collection? Most likely not; in [Lunt89a], Lunt considered the problem of joint attributes. A proper formulation of a problem is almost always a necessary step toward any significant progress. The security algebra, introduced by his colleagues and present author [Lin89b], provides a succinct theory to formulate the aggregation problems in database security. Aggregation, intuitively, is an "algebraic collection", more precisely, an entity E defined by an algebraic expression (of a relational algebra) operating on a collection of data. The aggregation problem then is the problem

of dealing with the possibility of inferring the information of an aggregate from that of individual data.

A multilevel relational database system is a system that contains information with different levels of sensitivity. In such a relational database system every entity (object, view) specified by a query should have a sensitivity level or security classification. Since every query in database can be described by a relational algebra, we can use the corresponding security algebra to compute the security classification of the entity. Now, the problem arises if the computed classification is different the actual classification assigned to this entity. In fact, this is one of the criteria that are used to detect the existence of aggregation problem. The central theme of this paper is to study and solve the usual aggregation problem through commutative security algebras. Non-commutative type of aggregation problems, which have not been discussed in the literature, will be reported in the forthcoming papers.

Security algebras are "derived" algebras of (extended) relational algebra. A relational algebra is called an extended relational algebra if the BUILT-IN functions, such as SUM(ADDITION), COUNT, MULTIPLICATION, and some other mathematical and string functions as part of the relational algebraic operations on the data. One of the novelties of current paper is that we view data as generators of the relational algebra. Every single data is regarded as a view (or relation) with single attribute and single value; it is called primitive data. Therefore any view (virtual table) is generated by these primitive data through the operations UNION and PRODUCT - See Proposition 2.4. Using this, a security algebra is constructed from the relational algebra using UNION and PRODUCT (by duality the INTERSECTION). In other words, we construct a security algebra which is (1) a subset of complete lattice, and (2) there is an algebraic homomorphism from the relational algebra (with operations: UNION and PRODUCT) to the lattice (with operation: l.u.b); the operations UNION and PRODUCT are "mapped" to g.l.b.; by duality, the INTERSECTION is "mapped" to g.l.b.. Depending on individual applications, this homomorphism [StMc77] may be able to extended to include the "mapping" of

- (1) BUILT-IN functions to operations in Security Algebra,
- (2) JOIN to multiplication (an operation in m-lattice), or
- (3) All operations to g.l.b.; This is Denning's lattice model.

Case (2) is deferred to future work. Note that the multiplication may not be commutative, so this is in the territory of (non-commutative) security algebra. See forthcoming paper.

From this formalism, we inductively revise the set of primitive data, and solve all the aggregation problems completely for the multilevel database.

This author would like express his sincere thank to Teresa Lunt for her encouragement in this investigation and the arrangement of the research grant.

2. EXTENDED RELATIONAL ALGEBRA

A relational database consists of a collection of primitive data (e.g., values of attributes). Intuitively, a sub-collection (e.g., a tuple, a sub-relation or a view) of these primitive data usually represents an entity, which is either a real-world entity (e.g., a department of a corporation) or an association among entities (e.g., between manager and employees). Any query in a relational database represents an entity. The data (i.e., virtual table) that describe the entity is called a view. We will use entity or view interchangeably. A query is definable via relational algebra, so an entity or a view is an expression of a (extended) relational algebra operating on the primitive data. (See examples below).

The usual relational operations are [Date86]

- (1) SELECT,
- (2) PROJECT,
- (3) PRODUCT,
- (4) JOIN,
- (5) DIVIDE,
- (6) UNION, INTERSECTION, and DIFFERENCE,

In most commercial products, the SQL always support some mathematical or string functions, such as maximum, sum, string length and etc. [Date86]. We would like to include all these built-in functions as part of relational operations whenever appropriate. So we include

- (7) BUILT-IN FUNCTIONS.

We shall call such algebra as Extended Relational Algebra.

The set of information represented by a database can be regarded as the set of all entities (views) describable by queries or relational algebra. Using the notation of [Lin89], such set will be denoted by CO. Note that the set CO contains elements, tuples, relations, sub-relations and, in general views. In other words, CO is the set of all views definable by (extended)relational algebra operating on the primitive data (atomic facts); we will refer to this set CO as the set of informations, the set of entities or the set of views.

Let E be an entity and A1, A2, ..., An be the primitive data that define the entity. Then E can be expressed by

$$(2.1) \quad E = f(A_1, A_2, \dots, A_n)$$

where f is an expression of extended relational algebra.

Example 2.1 Let the entity E be a committee represented by the relation COMMITTEE. Let A_1, A_2, \dots, A_n (Smith, Anderson,..) be its members represented by tuples in the relation. Then

$$E = A_1 \cup A_2 \cup \dots \cup A_n$$

where \cup is the UNION of the extended relational algebra.

COMMITTEE

member_name	city	status	classification
Smith	Rome	100	SECRET	
Anderson	Athens	500	SECRET	
Johnson	London	200	CONFIDENTIAL
Kennedy	Berlin	499	TOP_SECRET	
Hoover	Paris	400	TOP_SECRET	
		:		
		:		

Example 2.2 Let the entity E be the TOTAL number of US-TROOPS in Europe. Let A_1, A_2, \dots, A_n be the numbers of troops (atomic facts) in the cities X_1, X_2, \dots, X_n . Then

$$E = A_1 + A_2 + \dots + A_n$$

(2331 = 755 + 345 + 231 + 500 + 500)

where $+$ is the SUM in the extended relational algebra [Lin89a].

US-TROOPS

city_name	number_troops	classification
Rome	755	SECRET	
Athens	345	SECRET	
London	231	CONFIDENTIAL
Berlin	500	TOP_SECRET	
Paris	500	TOP_SECRET	
TOTAL(in Europe)	2331		

Example 2.3 Let E be a relation $E(\text{emp\#, name, address, salary, } \dots)$. Let $A_1 = \text{NAME}(\text{emp\#, name})$, $A_2 = \text{ADDRESS}(\text{emp\#, address})$, $A_3 = \text{EMP_SALARIES}(\text{emp\#, s\#})$, $A_4 = \text{SALARIES}(\text{s\#, salary})$,

$$E = A_1 \# A_2 \# A_3 \# A_4 \# \dots \# A_n$$

where $\#$ is the JOIN of the relational algebra.

Although there are six types of operations in the relational algebra, they are not the "minimal set of operations". Here, minimal set of operations means the minimum types of operations that are required to generate all possible information CO from primitive data (elements) in a relational database.

Proposition 2.4 The minimum set of relational operations are

- (1) PRODUCT
- (2) UNION

Proof: SELECT is a description of a subset of a relation; we can use the UNION of tuples to describe the subset. PROJECT is a description of a "sub-columns" of a relation that can be obtained by taking a Cartesian product on a subset of original attributes. JOIN and DIVIDE are both describable by SELECT and PROJECT. INTERSECTION and DIFFERENCE are methods of describing subsets of a relation; they can be expressed by the UNION of tuples.

Remark: Obviously, to get the minimum set of extended relational operations

- (3) BUILT-IN FUNCTIONS

are needed to be included.

Example 2.5 Let E be a relation

```

-----
file_name  city_name  number_troops  classification  .....
-----
Rome_file  Rome       755            SECRET
Athens_file Athens     345            SECRET
London_file London     231            CONFIDENTIAL  .....
Berlin_file Berlin     500            TOP_SECRET
Paris_file  Paris     500            TOP_SECRET
EUROPE_file Europe    2331            ***

```

Then E can be constructed from the elements as follows:

Each of the following four elements: Rome_file, Rome, 755, SECRET are four informations in CO. The product of this four elements form a tuple of E, i.e.,

tuple1 = Rome_file X Rome X 755 X SECRET

which is an information in CO.

Similarly, we can form

tuple2 = Athens_file X Athens X 345 X SECRET

```

tuple3 = London_file X London X 231 X CONFIDENTIAL
      :
      :
      :

```

The UNION of these tuples form the relation E.

This illustrates how a relation or view can be built from primitive data via minimum set of operations.

Definition 2.6 An entity (view) can be represented by an expression

$$E = f(A_1, A_2, \dots, A_n)$$

where f is an expression of minimum set of operations and A_1, A_2, \dots, A_n are primitive data. The expression is called a defining expression of E and the data a defining primitive data.

3 SECURITY CLASSIFICATION MAPS

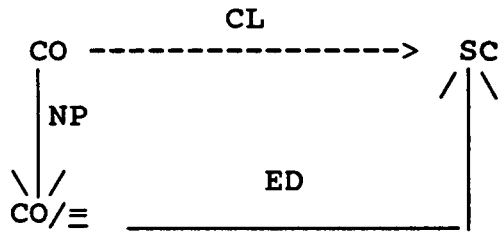
In this section, we will investigate two security classifications maps: one is the real world security classification. The other is derived from the primitive data and the structure of a commutative security algebra.

3.1 Real World Security Classifications

In a multilevel database system, each entity (atomic fact, view) is assigned a security classification. The goal is to assign the security classes to all entities so that they reflect the real-world classifications. Let us assume that such a security classifications has been assigned to the multilevel database system under consideration.

As in [Lin89b], let the set of all information stored in the multilevel database system be denoted by CO. We partition the set of information CO into disjoint equivalence classes according to their security classification: Two data d_1 and d_2 are equivalent, denoted by $d_1 \equiv d_2$, if they have the same security classification (i.e., $CL(d_1) = CL(d_2)$). This equivalence relation partitions CO into disjoint classes, and the set of these disjoint classes is denoted by CO/\equiv . Let SC be the poset of security classifications and NP be the natural projection that maps the element to its equivalent class. The security Classification map CL induces an one-to-one map ED (Embedding) from CO/\equiv to SC. That is, we have the following "commutative diagram":

[Two ways of traveling from CO to SC will give us the same result: $CL(co) = ED(NP(co))$, where co is a typical member of the set CO.]



For future references, let us summarize our notations as follows:

- (CL-1) CL is the security map which assigns every entity a security class to represent the real world security class of an entity.
- (CL-2) Let x be an entity, then the security class $\text{CL}(x)$ is denoted by $[[x]]$, i.e., $\text{CL}(x) = [[x]]$
- (CL-3) SC is the poset of security classes and is identified with CO/\equiv .

3.2 Algebraic Security Classifications

In the real world security classification, the security class of an entity may or may not reflect the algebraic structure of the relational algebra. In this subsection, we will base on the real world security classification of primitive data to construct a new security classification on CO, which reflects the algebraic structure of the relational algebra. Namely, we will define another security classification map, which is a homomorphism [StMc77] from the relational algebra to the security algebra. Note that given a relational algebra, there may have more than one security algebras. Our approach is applicable to any security algebra.

First, we need to extend the poset SC to a lattice, by the standard embedding theorem. Let us quote from [Birk60, pp. 58, Theorem 12] the following:

Theorem 3.1 Any poset can be embedded in a complete lattice, so that inclusion is preserved, together with all g.l.b. and l.u.b. existing in the poset.

Remark: This lattice structure is not the lattice of [Denn76]; it is a pure mathematical theorem. We will call this lattice LSC the enveloping lattice of the poset SC.

Let LSC be the complete lattice in Theorem 3.1. Let the new security map ACL be called Algebraic Classification Map. Recall that the real world security class is denoted by double brackets $[[\]]$ and Algebraic security class by single bracket $[\]$.

First we define the map ACL on primitive data.

$$\text{ACL}(A_i) = [[A_i]] \text{ for all primitive data } A_i.$$

that is $[A_i] = [[A_i]]$ for all primitive data.

Proposition 3.2 If the ACL can be extended to a homomorphism (its existence is exhibit in Section 4)

$$\text{ACL: CO} \text{ ----> LSC,}$$

then the new security class of a general entity E,

$$E = f(A_1, A_2, \dots, A_n),$$

can be defined by

$$\text{ACL}(E) = [f]([A_1], [A_2], \dots, [A_n])$$

where f is an expression in relational algebra and $[f]$ is the corresponding expression of f in LSC.

Proof: This follows immediately from the meaning of homomorphism.

Remark: There can be more than one such ACL, depending on how much structure is imposing on CO.

Let us summarize our notations:

- (ACL-1) ACL is the security map which assigns
 - (1) every primitive entity a security class of real world security class.
 - (2) every general entity by its algebraic structure.
(see Proposition 3.2)
- (ACL-2) Let x be an entity, then the security class $\text{ACL}(x)$ is denoted by $[x]$, i.e., $\text{ACL}(x) = [x]$
- (ACL-3) LSC is the enveloping lattice of SC.

4. SECURITY ALGEBRA

4.1 Security Algebra

In [Lin89], we show that a relational operation induces an operation on SC if the security map CL satisfies Denning's axiom for that operation. The induced algebraic structure on SC is called security algebra.

A security algebra consists of three objects:

- (1) A relational algebra RA which is the set of information CO together with certain operations, denoted by $*r*$ (Usually, it is a subset of the extended relational operations).
- (2) A security algebra SA which is a poset SC or its extension together with certain operations, denoted by $*s*$ (e.g.,

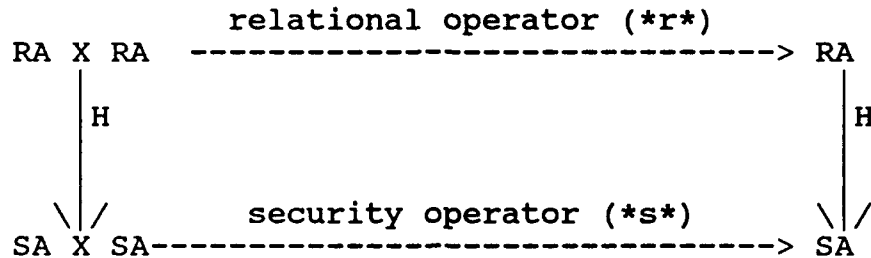
l.u.b. and g.l.b.)

(3) A homomorphism $H: RA \dashrightarrow SA$, which is a security classification map satisfying the following property:

$$[d1 *r* d2] = [d1] *s* [d2], \text{ for all } *r* \text{ in } RA$$

where d_i is in RA and $H(d_i) = [d_i]$ is in SA , for $i = 1, 2$; $*r*$ and $*s*$ represent the corresponding pair of operations in RA and SA respectively.

In other word, the following diagram holds:



where $*r*$ represents every operation in the particular subset (under consideration) of the (extended) relational operations.

Definition 4.1 A security algebra is said to be commutative if $d1 *s* d2 = d2 *s* d1$.

Remark: There are real needs for non-commutative case. See forthcoming paper.

There are 3 common commutative security algebras in usual relational database (More precisely, there are six; in each case, either one uses relational algebra or extended relational algebra).

Case 1. For general systems, the minimum set of operations, namely, UNION and PRODUCT are mapped to l.u.b. of LSC; by duality INTERSECTION is mapped to g.l.b. (See Section 4 for the construction of this security algebra).

For extended relational algebra, the BUILT-IN FUNCTIONS, SUM (ADDITION) and MULTIPLICATION are mapped to l.u.b..

Case 2. For some systems, the relational operations are mapped into a multiplicative lattice LSC, where UNION and PRODUCT are mapped to l.u.b. and JOIN is mapped to a multiplication; by duality INTERSECTION is mapped to g.l.b. -- This case is differed to later papers.

Case 3. For some systems, we impose Denning's axiom on every extended relational operations. Namely, UNION, PRODUCT, JOIN, SUM (ADDITION) and MULTIPLICATION including all BUILT-IN FUNCTIONS are all mapped to g.l.b.; this is the classical lattice model of Denning.

4.2 Case 1 -- A Commutative Security Algebra

In this section we will construct a security algebra for relational database. Namely the Case 1. in Section 4.1.

From 4.1, we need to construct RA, SA and H.

RA is the set CO of informations together with the minimum set of operations, UNION, PRODUCT and INTERSECTION. SA, a subset of LSC, and H will be constructed together.

Let the security class of all the primitive data be defined by real world classifications: For all primitive data A_i ,

$$H(A_i) = [A_i] = [[A_i]].$$

Let E be an entity (view) which is defined by the primitive data A_1, A_2, \dots, A_n via minimal set of operations (see 2.4):

$$E = f(A_1, A_2, \dots, A_n)$$

Then, the security class of the expression $f(A_1, A_2, \dots, A_n)$ can be defined by

$$[f()] = [f]([A_1], [A_2], \dots, [A_n])$$

where $[f]$ is the expression obtained from f by replacing

- (1) A_i by $[A_i]$, and
- (2) UNION and PRODUCT by l.u.b., #, of LSC.
- (3) INTERSECTION by g.l.b., *, of LSC.

Obviously $[f]$ is an expression in LSC. So $[f()]$ is an element of LSC.

Proposition 4.2 $[f]([A_1], [A_2], \dots, [A_n]) = \text{l.u.b.}\{[A_1], [A_2], \dots, [A_n]\}$

Proof: The expression f is a union of all the tuples in the view (virtual table) E. Each tuple, say (t_1, t_2, \dots, t_k) is mapped to

$$[t_1] \# [t_2] \# \dots \# [t_k]$$

Note that UNION is also mapped to #. Therefore

$$f(A_1, A_2, \dots, A_n)$$

is mapped to

$$[A1]#[A2]#\dots#[An] = \text{l.u.b.}\{[A1],[A2],\dots[An]\}$$

This proved the proposition.

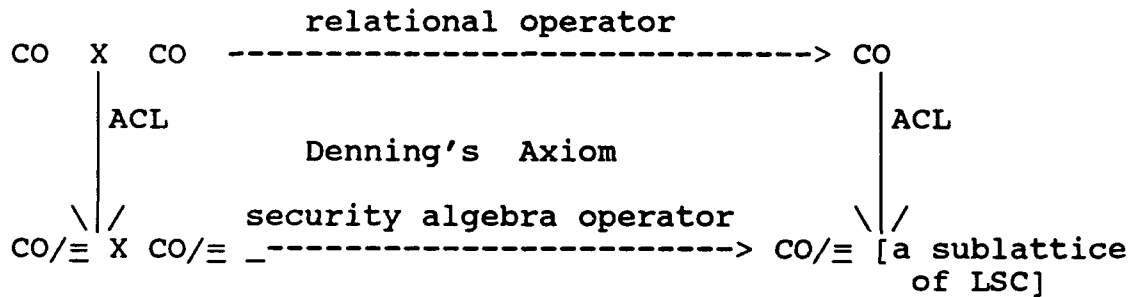
Definition 4.3 The homomorphism H is defined by

$$H(E) = [E] = \text{g.l.b.}\{[f()]: \text{for all } f \text{ which defines } E\}.$$

As in Proposition 3.2 H(E) exists, since LSC is a complete lattice.

As in Section 3, we will use ACL to denote H.

Proposition 4.4 The map $ACL: CO \dashrightarrow LSC$ satisfies Denning's axiom for the operations in the minimum set of operation, that is, the following diagram is commutative:



where relational operator is in the minimum set of operations namely, UNION and PRODUCT.

Proof: Let E1 and E2 be two entities in CO. Let g be a relational operation in the minimum set. Then the new entity E3

$$E3 = g(E1, E2)$$

has the new security class

$$ACL(E3) = [E3] = \text{g.l.b.}\{[f()]: \text{for all } f \text{ defines } E3\} \leq [g()], \text{ since } g \text{ is a particular } f$$

On the other hand $ACL(E3) = [E3] \geq [E1]$ and $[E2]$, since all the defining primitive data of E3 contains the defining primitive data of E1 and E2.

Therefore $[E3] = \text{l.u.b.}\{[E1], [E2]\}$. In other words, for any relational operator in the minimum set, the induced security operator is the least upper bound operator of the lattice LSC. Therefore the diagram above is commutative.

In fact, the proof give us more than Denning's axiom. It give us

the following theorem.

Theorem 4.5 Let LSC be the enveloping lattice of the security poset SC. Let CO be the set of all entities (views) in a relational database. Let ACL be the security map, then the security algebra induced from the minimum set of operations, is a sublattice of LSC.

Proof: In the proof of previous proposition, we show that $[E3] = [g(E1, E2)] = \text{l.u.b.}\{[E1], [E2]\}$, where g is in the minimum set operations. This shows that the operation in the security algebra CO/\equiv is the lattice operation. QED.

Although in this theorem only the operations in the minimum set of operations (i.e., UNION and PRODUCT) are mapped into security algebra; by the duality of lattice, INTERSECTION is mapped to the greatest lower bound in LSC.

Under this general ACL, JOIN may not satisfy Denning's axiom. If for some system JOIN does satisfy Denning's axiom, then, as we remarked in [Lin89], JOIN may induce a new operations in LSC, which gives LSC the multiplicative lattice structure (or simply m-lattice) [Birk60, pp.200]. Since the algebraic nature of m-lattice is not well presented for computer science, we will not impose the m-lattice structure into LSC at this time, and defer the study to later papers.

Example 4.6 Let R be a relation which is a full set of Cartesian product of three attributes domains. Let $R = D1 \times D2 \times D3$, which is a relational expression. Apply the ACL to R, we have

$$[R] = [D1] \# [D2] \# [D3]$$

This is an expression in security algebra.

Example 4.7 Let E be the following relation (see Example 2.5)

file_name	city_name	number_troops	classification
Rome_file	Rome	755	SECRET	
Athens_file	Athens	345	SECRET	
London_file	London	231	CONFIDENTIAL
Berlin_file	Berlin	500	TOP_SECRET	
Paris_file	Paris	500	TOP_SECRET	
EUROPE_file	Europe	2331	***	

We will assume the value in the classification field indicates the security class of the tuple, we also assume that every field in the tuple has same classification of the tuple. For example

$$[Rome_file] = [Rome] = [755] = [SECRET] = SECRET$$

Then, we can compute the security class of E by the defining expression:

```
E = tuple1 U tuple2 U ..... U tuple6

    = Rome_file X Rome X 755 X SECRET
    U Athens_file X Athens X 345 X SECRET
    U London_file X London X 231 X CONFIDENTIAL
      :
      :
```

Now, the expression is mapped to the following expression in the security algebra:

```
[E] = [Rome_file] # [Rome] # [755] # [SECRET]
      # [Athens_file] # [Athens] # [345] # [SECRET]
      # [London_file] # [London] # [231] # [CONFIDENTIAL]
      :
      :
      = SECRET # SECRET # SECRET # SECRET
      # CONFIDENTIAL # CONFIDENTIAL # CONFIDENTIAL # CONFIDENTIAL
      # TOP_SECRET # TOP_SECRET # TOP_SECRET # TOP_SECRET
      :
      = TOP_SECRET
```

5. AGGREGATION

5.1 What is Aggregation ?

According to [Lunt89],

(1) "The aggregation problem arises whenever some collection of facts has a classification strictly greater than that of the individual facts forming the aggregate".

(2) "To qualify as an aggregation problem, it must be the case that the aggregate class strictly dominates the class of every subset of the aggregate"

Our notion of aggregation is basically the same as hers, however, the formulation is slightly different. We consider aggregation as **a pair, the collection and the individual facts**. Roughly, the aggregation problem exist if **and only if** the classification of the collection is strictly greater than that of the individual facts. Moreover, a collection here is not merely the set theoretical collection, in fact, any algebraic expression is an aggregate -- See below.

An aggregation that satisfies (2) is called **simple aggregation**; here **simple** conforms the common usages in abstract algebra; no sub-aggregate exists. (e.g., A simple abelian group is an abelian

group that has no abelian subgroup). Lunt and many others have good strategies to solve these simple aggregation problems [Dill88], [Hink88], [Lunt89a], [Sayd87]. If we incorporate their solutions to our procedure, then we solve all the aggregation problems in a relational database.

Let E be the collection of A_1, A_2, \dots, A_n . Then the pair

$$(5.1) \quad E; A_1, A_2, \dots, A_n$$

is an aggregation.

Suppose B , a sub-collection of E , consists of A_1, A_2, \dots, A_k and $k < n$. Then, the entity can be expressed by

$$E = B \cup A_{(k+1)} \cup \dots \cup A_n.$$

The consideration of B induces two new aggregations; one of them is the pair

$$(5.2) \quad E; B, A_{(k+1)}, \dots, A_n.$$

and another is

$$(5.3) \quad B; A_1, A_2, \dots, A_k$$

In [Lunt89a], Lunt considered (5.1) and (5.3) as one problem. In our approach (5.1), (5.2) and (5.3) are three individual problems, however, the procedure will naturally take care of three problems at the same time.

Roughly, every non-primitive entity is an aggregation; the entity and the defining primitive data (see 2.6) is the pair for an aggregation.

Definition 5.1 Let E be an entity and $E = f(E_1, E_2, \dots, E_k)$ is an expression in the extended relational algebra. Then E is a relative aggregation problem if and only if

$$[[E]] \neq [f]([[E_1]], [[E_2]], \dots, [[E_k]])$$

where \neq means not equal, $[f]$ is the induced expression of f in the security algebra contained in LSC.

Recall that $[[E]]$ is the security classification reflecting the real world data. The security classification map is CL , i.e., $CL(E) = [[E]]$. See $CL-1$, $CL-2$ and $CL-3$ in Section 3.1. Note that for primitive data $[[A_i]] = [A_i]$.

If we apply the Definition 5.1 to the defining expression of E , then we have

Definition 5.2 Let E be an entity. Then $E = f(A_1, A_2, \dots, A_n)$ is an absolute aggregation problem if and only if

$$[[E]] \langle \rangle [E] = [f]([A_1], [A_2], \dots, [A_n]).$$

Definition 5.3 Let E be an entity. Then $E = f(E_1, E_2, \dots, E_n)$ is a **simple (relative) aggregation** problem if and only if

- (1) E is an aggregation
- (2) All sub-entities $E' = f'(E_j, \dots, E_k)$, which are entities represented by proper sub-expressions of f, are not aggregation problems, in other words,

$$[[E']] = [f']([E_j], \dots, [E_k]),$$

for all proper sub-expressions f' of f.

Here "simple" conforms the usages of abstract algebra; for example, a simple abelian group means no abelian subgroups.

Example 5.4 Let R be a relation which is a whole set of Cartesian product of two attributes domains. Let $R = \text{NAME} \times \text{SALARY}$, which is a relational expression. Apply the ACL to R, we have

$$[R] = [\text{NAME}] \# [\text{SALARY}]$$

The security classes of individual fields may be

$$[\text{NAME}] = [\text{SALARY}] = \text{UNCLASSIFIED}$$

By computation rules of # in LSC, we have

$$\begin{aligned} [R] &= \text{UNCLASSIFIED} \# \text{UNCLASSIFIED} \\ &= \text{UNCLASSIFIED} \end{aligned}$$

In reality, one may assigned R a higher classification, e.g., $[[R]] = \text{SECRET}$. Thus there is an aggregation problem. -- Merely upgrading R can only shut off the data flow but not the inference flow.

Example 5.5 (see Example 2.2) Let the entity E be the TOTAL number of US-TROOPS in Europe. Let A_1, A_2, \dots, A_n be the numbers of troops (atomic facts) in the cities X_1, X_2, \dots, X_n . Then

$$\begin{aligned} E &= A_1 + A_2 + \dots + A_n \\ (2331 &= 755 + 345 + 231 + 500 + 500) \end{aligned}$$

where + is the SUM in the extended relational algebra.

Apply ACL to the extended relational expression, we have

$$[E] = [A_1] + [A_2] + \dots + [A_n]$$

$$([2331] = [755] + [345] + [231] + [500] + [500])$$

Let us express the problem in terms of field names (the first field value as tuple name), we have

$$\text{EUROPE.troop_number} = \text{Rome.troop_number} + \text{Athens.troop_number} + \text{London.troop_number} + \text{Berlin.troop_number} + \text{Paris.troop_number}$$

Apply ACL to the expression, then the induced expression in security algebra is

(Security class of individual field is the same as its tuple)

$$\begin{aligned} [\text{EUROPE.troop_number}] &= [\text{Rome.troop_number}] \\ &\# [\text{Athens.troop_number}] \# [\text{London.troop_number}] \\ &\# [\text{Berlin.troop_number}] \# [\text{Paris.troop_number}] \\ &= \text{SECRET} \# \text{SECRET} \# \text{CONFIDENTIAL} \# \text{TOP_SECRET} \# \text{TOP_SECRET} \\ &= \text{TOP_SECRET, by properties of } \# \end{aligned}$$

However, in real life we may downgrade the EUROPE.troop_number to UNCLASSIFIED for news release.

Remark: This type of aggregation appears as an "inverse aggregation inference", it is an aggregation in our formulation. Let us consider the following situation: If the entity E is the pair (TOTAL, COUNT) of total number of US-troops and the number of locations, for example

$$E = (A_1, 1) + (A_2, 1) + \dots + (A_n, 1)$$

Then, issues queries and obtained the following answers:

$$\begin{aligned} (2331, 5) &= (755, 1) + (345, 1) + (231, 1) + (500, 1) + (500, 1) \\ (1831, 4) &= (755, 1) + (345, 1) + (231, 1) + (500, 1) \\ (1331, 3) &= (755, 1) + (345, 1) + (231, 1) \\ (1100, 2) &= (755, 1) + (345, 1) \\ (755, 1) &= (755, 1) \end{aligned}$$

One can find the individual number of right hand side by solving the system of simultaneous equations using only left hand side information. This an over simplified version of statistical inference.

This type of aggregation are important in statistical inference. For general discussions see [Ting89], [Mat189].

Example 5.6 Sensitive Association.

$$\begin{aligned} [[\text{Smith}]] &= [\text{Smith}] = \text{UNCLASSIFIED} \\ [[500000]] &= [500000] = \text{UNCLASSIFIED} \end{aligned}$$

The security class of sensitive association can be expressed in the security algebra by

```
[Smith X 500000] = [Smith] # [500000]
                 = UNCLASSIFIED # UNCLASSIFIED
                 = UNCLASSIFIED
```

However, in real world, one would like to assign

```
[[Smith X 500000]] = SECRET
```

So this is an aggregation problem.

It is important to observe here that we are not handling the product of name list and salary list. We are handling individual name and his/her salary. The solution to this problem is to give up the primitive data Smith and 500000 and to adopt a new primitive data, the pair Smith X 500000; so this pair is not a **derived** data of lower data [Smith or 500000 is no longer lower data]. The releasing of these two data is controlled by SSO (SeaView) or some other mechanism (LDV [Dill88], [Stac88]).

Example 5.7 "Complex" or "Multilevel" aggregation problem.

```
[[A1]]=[[A2]]=...[[An]]= UNCLASSIFIED
[[{A1, A2}]] = CONFIDENTIAL
[[{A1, A2, A3, A4}]] = SECRET
[[{A1, A2, A3, A4, A5, A6}]] = TOP_SECRET
```

Here {A1,A2} is a simple aggregation, {A1, A2, A3, A4} and {A1, A2, A3, A4, A5, A6} are "complex" aggregations. However, they can be transform to relative simple aggregations.

5.2 Solutions to Simple Aggregation Problems

Various proposals and efforts to settle the aggregation problems have been well discussed in the [Dill88], [Hink88], [Lunt89a], [Sayd87]. Basically the solutions to simple aggregations can be classified into two types:

(1) The original aggregation problem

$$E = f(E_1, \dots, E_k)$$

can be transformed to a new expression

$$E = g(F_1, F_2, \dots, F_h)$$

so that the pair E;F1,F2,...Fh is not an aggregation. We will call such simple aggregation a **fake aggregation**.

(1.1) The primitive data E1,E2,...Ek in the original database are removed and replaced by F1,F2,...Fh [Lunt89, pp. 104].

(2) This is the type of simple aggregation that no transformation exists; it is a true aggregation problem. The solution to this type of problems usually is one of the following

(2.1) Upgrade the primitive data and SSO controls the releasing of upgraded data. (SeaView)

(2.2) Arrange a special mechanism to control the release of primitive data (LDV).

Intrinsically, both approaches are the same from our point of view, LDV automates the sanitizing process, while SeaView entrust the control to SSO (They do have their differences which do not concern us [Lunt89]). So, we summarize the solution (SeaView and LDV) by formalizing it as follows:

Let $E = f(E_1, E_2, \dots, E_k)$ be a true simple aggregation. Then the solution of E is to

- (2.1) Add the database a new primitive fact E
- (2.2) Remove E_1, E_2, \dots, E_k from the original primitive data set
- (2.3) The release of any of E_1, E_2, \dots, E_k are handled by a special procedure which is processed by SSO or a special mechanism.

Remark: (2.3) need more study, current solution offered either by SeaView or LDV is not very satisfactory. Morgenstern's proposal may be useful here [Morg88].

5.3 Detecting and Solving Aggregation Problems

Let Q be a query to a relational database. Let E be the view that answers Q . The entity E can be expressed by

$$E = f(A_1, A_2, \dots, A_n)$$

where f is an expression of the extended relational algebra operating on the primitive data. The procedure for detecting and solving the aggregation problem will proceed inductively as follows:

We will present the induction rather informally; leave the rigorous proof to serious readers.

ASSUMPTION: All true simple aggregation problems are solved.
(See 5.2)

Let us set up some notations: Any proper sub-expression f' of f defines a sub-entity E' . Recall that real world security class is represented by $[[[]]$.

We can induct on the number of levels of aggregations.

Induction Assumption: All aggregations in which the level of the aggregation of its expression tree is less than k are all solved.

1. For the expression tree of f, the system examines all its subtrees bottom up. Note a subexpression corresponds to a subtree of the expression tree of f
2. If $[E'] = [[E']]$ for all proper sub-expressions, then E is a simple aggregation. By assumption, it is solved.
3. Let f' be the first sub-expression such that $[E'] \neq [[E']]$, then E' is a simple aggregation. By assumption it is solved. Namely, the primitive data of f', say A1, A2, ... Ak (this representation does not lose the generality) are removed and E' is added to the primitive data list.

4. So the new expression for E is

$$E = g(E', A(k+1), \dots, A_n)$$

where E' is the simple true sub-aggregation.

5. If $[[E]] = [g]([[E']], [[A(k+1)]], \dots, [[A_n]])$ then the original aggregation problem is solved.
6. If $[[E]] \neq [g]([[E']], [[A(k+1)]], \dots, [[A_n]])$ then the original aggregation problem is reduced to g.
7. By induction, g can be solved (the level of g is less than k).
8. This completes the induction.

Example 5.8 A1, A2, ... A6 have the following classifications:

$[A1] = [A2] = \dots [A_n] = [[A1]] = [A2]] = \dots [[A_n]] = \text{UNCLASSIFIED}$
 $[[\{A1, A2\}]] = \text{CONFIDENTIAL}$
 $[[\{A1, A2, A3, A4, A5, A6\}]] = \text{CONFIDENTIAL}$

Here {A1, A2} is a simple aggregation.

1. First evaluate $E_{12} = A1 \cup A2$ and we find

$$[[E_{12}]] \neq \text{g.l.b. } \{[A1], [A2]\}$$

where $[[E_{12}]] = \text{CONFIDENTIAL}$ and $[A1] = [A2] = \text{UNCLASSIFIED}$. So E12 is a true simple aggregation.

2. Apply the solution, we have

$$E = \{E_{12}, A3, A4, A5, A6\}$$

3. Evaluate $E = E12 \cup A3 \cup A4 \cup A5 \cup A6$ and we find

$$[[E]] = \text{g.l.b.} \{ [E12], [A3], [A4] \}$$

where $[[E]] = [E12] = \text{CONFIDENTIAL}$, $[A1] = [A2] = \text{UNCLASSIFIED}$.

So, E, with this new expression, is not an aggregation.

4. This solves the aggregation problem.

Example 5.9 The solution for Example 5.7:

$$\begin{aligned} [A1] = [A2] = \dots [An] &= [[A1]] = [[A2]] = \dots [[An]] = \text{UNCLASSIFIED} \\ [[\{A1, A2\}]] &= \text{CONFIDENTIAL} \\ [[\{A1, A2, A3, A4\}]] &= \text{SECRET} \\ [[\{A1, A2, A3, A4, A5, A6\}]] &= \text{TOP_SECRET} \end{aligned}$$

Here $\{A1, A2\}$ is a simple aggregation, $\{A1, A2, A3, A4\}$ and $\{A1, A2, A3, A4, A5, A6\}$ are "complex" aggregations.

1. First evaluate $E12 = A1 \cup A2$ and we find

$$[[E12]] \langle \rangle \text{g.l.b.} \{ [A1], [A2] \}$$

where $[[E12]] = \text{CONFIDENTIAL}$ and $[A1] = [A2] = \text{UNCLASSIFIED}$. So E12 is a true simple aggregation.

2. Apply the solution, we have

$$E = \{E12, A3, A4, A5, A6\}$$

3. Evaluate $E1234 = E12 \cup A3 \cup A4$ and we find

$$[[E1234]] \langle \rangle \text{g.l.b.} \{ [[E12]], [A3], [A4] \}$$

where $[[E1234]] = \text{SECRET}$, $[[E12]] = \text{CONFIDENTIAL}$, $[A1] = [A2] = \text{UNCLASSIFIED}$.

So E1234 is a true simple aggregation.

4. Apply the solution, we have

$$E = \{E1234, A5, A6\}$$

5. Evaluate $E = E1234 \cup A5 \cup A6$ and we find

$$[[E]] \langle \rangle \text{g.l.b.} \{ [E1234], [A5], [A6] \}$$

where $[[E]] = \text{TOP_SECRET}$, $[[E123]] = \text{SECRET}$, $[E12] = \text{CONFIDENTIAL}$, $[A1] = [A2] = \text{UNCLASSIFIED}$.

So E is a true simple aggregation.

The solution is that E is the only primitive data for E. All the sub-data can only be released by SSO or special mechanism. If we apply Lunt's solution, then

A1 and A2 will have three aggregation labels

for belonging to three aggregations, and

A3 and A4 will have two aggregation labels

for belonging to two aggregations, and

A5 and A6 will have one aggregation labels

for belonging to one aggregation.

It is not clear that LDV's mechanism can solve this problem.

6. CONCLUSIONS AND FUTURE WORKS

Based on the current solution proposed by various projects, a general procedure is developed to detect and solve the aggregation problems. The solution is complete in the sense all aggregations are detected and the problems are reduced to classical solutions.

However, the classical solution may not be very satisfactory in the light of Example 5.9. Moreover, the current aggregation problems have only addressed the commutative case. In a non-commutative case, the situation is more complex. The knowledge of individual data is not necessary allow one to infer the information of aggregation unless the individual data is in the "correct" order. For example, a text message of thousand words is not necessary compromised if an uncleared person knows every word of the message in alphabetical order. In the forthcoming paper we will report on this type of problem.

ACKNOWLEDGE

This research was supported by the U. S. Air Force, Rome Air Development Center (RADC), under SRI's Subcontract C-12537. I am indebted to both RADC and SRI for making this work possible.

REFERENCES

[Denn76] D. E. Denning. "A Lattice Model of Secure Information Flow", Communications of the ACM, Vol. 19, No. 5, May 1976, pp. 236 - 243.

[Denn86a] D. E. Denning. The Inference Problem in Multilevel Database systems, In the Proceeding of the National Computer Security Center Invitational Workshop on Database Management Security, June 1986.

[Denn86b] D.E. Denning, S.G. Akl, M. Heckman. T.F. Lunt, M. Morgenstern, P.G. Neumann, and R.R. Schell. View for Multilevel Database Security, Proc. IEEE Symposium on Security and Privacy, 1986.

[Denn87a] D.E. Denning, T.F. Lunt, R.R. Schell, M. Heckman and W.R. Shockley, A Multilevel Relational Data Model, Proc. 1987 IEEE Symposium on Security and Privacy, 1987.

[Denn87b] D.E. Denning, T.F. Lunt, R.R. Schell, M. Heckman and W.R. Shockley, "The SeaView Formal Security Policy Model", Computer Science Laboratory, SRI International, July, 1987.

[Denn88a] D.E. Denning, T.F. Lunt, R.R. Schell, W.R. Shockley, M. Heckman, The SeaView Security Model, Proc. 1988 IEEE Symposium on Security and Privacy, 1988.

[Denn88b] D.E. Denning, T.F. Lunt, P.G. Neumann, R.R. Schell, M. Heckman and W.R. Shockley, "Security Policy and Policy Interpretation for a Class A1 Multilevel Secure Relational Database System", Computer Science Laboratory, SRI International, Aug. 1988.

[Dill88] Dillaway et al. Security Policy Extensions for a Database Management System. Interim report A002. Honeywell Systems Research Center and Corporate System Development Division, may 1988.

[Gass88] M. Gasser, Building A Secure Computer System, Van Nostrand Reinhold Company, 1988.

[Gogu84] J. A. Goguen and J. Meseguer, Unwinding and Inference Control, Proc. 1984 IEEE Symposium on Security and Privacy, 1984.

[Hink88] T. H. Hinke. Inference Aggregation Detection in Database management systems, In Proceedings of the 1988 IEEE Symposium on Security and Privacy, April 1988.

[Lin89a] T. Y. Lin, A Generalized Information Flow Model and Role of System Security Officer, Database Security: Status and Prospects II, edited by C. E. Landwehr, North Holland, 1988.

[Lin89b] T. Y. Lin, L. Kerschberg and R. Trueblood. Security Al-

gebra and Formal Models, Proceedings of IFIP WG11.3 Workshop on Database Security September 5-7, 1989

[Lunt88a] T.F. Lunt, R.A. Whitehurst, The SeaView Formal Top Level Specifications, Computer Science Laboratory, SRI International, Feb. 1988.

[Lunt88b] T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman and W. R. Shockley, Element-Level Classification with AI Assurance, Computers & Security, Vol. 7, No. 1, 1988, pages 73-82.

[Lunt88c] T. F. Lunt, R. R. Schell, W. R. Shockley, M. Heckman, D. Warren, A Near-Term Design for the SeaView Multilevel Database System, Proceedings 1988 IEEE Symposium on Security and Privacy, 1988.

[Lunt89a] T. F. Lunt. Aggregation and Inference: Facts and Fallacies, Proceedings 1989 IEEE Symposium on Security and Privacy, 1989.

[Matl89] N. Matloff and P. Tendick, The "Curse of Dimensionality" in Database Security, Database Security: Status and Prospects II, edited by C. E. Landwehr, North Holland, 1988.

[Morg87] Mathew Morgenstern. Security and Inference in Multilevel Database and Knowledge-base systems, ACM International conference on Management of Data (SIGMOD-87), May 1987

[Morg88] Mathew Morgenstern. Controlling Logical Inference in Multilevel Database Systems, Proceedings 1988 IEEE Symposium on Security and Privacy, 1988.

[Sayd87] O. S. Saydjari, J. M. Beckman and J. R. Leaman, Locking Computers Securely, Proc. 10th National Computer Security Conference, Sept. 1987, National Bureau of Standards/ National Computer Security Center, pp. 129-141, 1987. Advances in Computer System Security, Vol. 3, edited by Rein Turn, pp. 207- 219, 1988.

[Stac88] Stachour et al, Secure Distributed Data Views -- Implementation Specifications. Interim Report A005. Honeywell Systems Research Center and Corporate system Development Division, May 1988

[StMc77] Donlod F. Stanat and David F. McAllister: Discrete Mathematics in Computer Science, Prentice-Hall, Inc., Englewood Cliffs, N. J. , 1977

[SuOz87] T. Su, and G. Ozsoyoglu, Data Dependencies and Inference Control in Multilevel Relational Database Systems, IEEE Symposium on Security on Security and Privacy, Oakland, CA, April 1987

[SuOz89] T. Su and G. Ozsoyoglu, Multivalued Dependency Inferences in Multilevel Relational Database Systems, Proceedings of IFIP WG11.3 Workshop on Database Security September 5-7, 1989

[Ting89] I. T. Leong and T. C. Ting. An analysis of Database Security with queries for High Order statistical Information, Database Security: Status and Prospects II, edited by C. E. Landwehr, North Holland, 1988.