COMMUNICATIVE DATABASES

Kwang-I Yu

Computer Science Department
California Institute of Technology

4527:TR:81

COMMUNICATIVE DATABASES

By

Kwang-I Yu

Technical Report #4527

Department of Computer Science

California Institute of Technology

Pasadena, California

1981

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

Acknowlegements

I wish to thank Frederick Thompson who has been my teacher. His ideas and philosophies will always be a part of me.

I wish to thank Carver Mead for his friendship and encouragement.

I wish to thank Mike Ullner with whom I had many rewarding discussions on my work.

Last but certainly not least, I wish to thank my parents and my wife for their love and support.

Abstract

A hierarchical organization stores its information in a large number of databases. These databases are interrelated, forming a closely-coupled database system. Traditional information systems and current database management systems do not have a means of expressing these relationships.

This thesis describes a model of the information structure of the hierarchical organization that identifies the nature of database relationships. It also describes the design and implementation of the Communicative Database Management System (CDMS).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The efficient management of information has always been crucial to the success of any human organization. As the modern organization becomes larger and more complex, its information requirements increase exponentially. Information generated or received from the outside at one department must be made available to other departments. No segment of the organization can set up an independent and self-sufficient information system.

In the 1960s organizations began to use computers to process information in isolated functional areas. These "applications-oriented" systems were generally successful in their limited ways, increasing the information processing capabilities of individual departments many fold. However, they could not account for the information dependencies among different departments and functional areas. For example, time cards processed for payroll must be processed separately for production control.

The decade of the 1970s saw the emergence of the database management system (DBMS) as the new paradigmatic approach to meeting the challenge of comprehensive information management. DBMS seeks to encompass all the data in an organization in a cohesive computerized system that satisfies the information requirements of all departments in all functional areas.

By and large, DBMS's have fallen far short of their stated objectives. Things have not improved much since R. C. Canning somewhat cynically listed the seven stages in a typical DBMS installation project [Canning 73]:

1. Uncritical acceptance
2. Wild enthusiasm
3. Dejected disillusionment
4. Total confusion
5. Search for the guilty
6. Punishment of the innocent
7. Promotion of non-participants

An important reason behind the failure of DBMS's is that their designers discarded the traditional information system all too quickly. A traditional system embodies the habits and experiences of the people of the organization; it reflects their organizational philosophy. While attempting to compensate for the inadequacies of the traditional system, DBMS designers frequently fail to retain its useful qualities. As a result DBMS's often seem "unnatural" to their users and unresponsive to their needs. The installation of such a system is predictably disruptive to the operation of the organization.

The object of this research is to design a database management system that is compatible to the traditional information system. Such a system must be flexible enough to accommodate the traditions of the organization, and powerful enough to correct their inadequacies. As Frederick Thompson wrote in "Design Fundamentals of Military Information Systems" [F. Thompson 61]:

> Although these traditions in themselves are no longer adequate for solving the severe informational problems of [an organization], they are an important part of its foundation and must be understood, improved, and strengthened, not ignored and replaced.

This chapter defines the problem area, outlines the general classes of DBMS's that are designed to solve it, and describes our own approach to seeking a solution.

1.1 Terminology

Since database management is a relatively young field, people tend to take some liberty in inventing new terms or using older ones. Before we launch into a discussion of the problem area, let us establish the terminology that will be used in our discussion.

An "information system" is that part of an organization which handles information processing and communication. It includes data, people, languages of discourse, and operational procedure.

Data, whether they are stored in loose-leaf binders, file cabinets, or computer storage media, are grouped into databases. A database is a collection of interrelated data that is distinguishable from all other databases by the following properties:

(1) Information: Each database contains a different class of information.

(2) Access rights: Each user enjoys different access rights to each database. There are some databases whose contents he can alter, others he can query without being allowed to update, and still others that he cannot access at all.

(3) Maintenance responsibilities: The responsibilities for keeping the contents of different databases current are assigned to different groups of users.

(4) Timeliness requirements: Since the cost of updating generally increases with the stringency of timeliness requirements, cost effectiveness determines how current a database is to be kept.

(5) Retention schedule: The time span over which specific data items remain relevant differ from database to database.

(6) Volatility: Some databases change more frequently than others.

(7) Interpretation and implicit qualifications: Each database carries with it an unstated context. One must be familiar with that context to interpret the meaning of the data correctly.

A self-contained, interrelated group of databases constitute a "database system". A database system forms the foundation of the organizational information system.

A DBMS is a computer system, consisting of a single installation or a network of connected installations, that encompasses and automates an organizational database system.

1.2 The Hierarchical Organization

Websters New Collegiate Dictionary defines "organization" as 1. association, society (as in charitable organization); 2. an administrative and functional structure (as a business or political party). This definition covers such diverse structures as Overeaters Anonymous, the International Olympic Committee, military commands, alumni associations, and commercial companies. The membership of these structures range from less than a hundred to many thousands. Their organizational complexities vary just as much. They have vastly different information requirements.

In this thesis we shall concentrate on an important subclass of organizations, namely the hierarchical organizations. A hierarchical organization has a multiple-leveled command structure. It consists of numerous divisions and subdivisions, each with its own special area of concern. The execution of a task requires the coordinated effort of many divisions. Examples of hierarchical organizations include corporations and military commands.

The hierarchical organization is usually represented by an organizational chart like the one shown in Figure 1.1. Each box on the chart represents a political division within the organization; the tree below it represents its subdivisions.

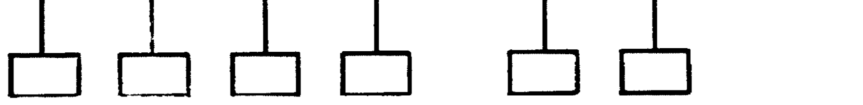

Figure 1.1 - An organizational chart.

The hierarchical organization rarely operates like the neat tree structure shown in Figure 1.1. Functional entities overlap and cut across the organizational chart, each collecting, generating, and processing information. Each functional entity stores information in its own databases, and communicates with other functional entities by accessing their databases.

It is inherent in the nature of the database system of the hierarchical organization that it encompasses many distinct databases. These databases are interrelated, just as the functional entities that access them form a single cohesive structure. The goal of the DBMS is to computerize this database system to provide current, accurate, and relevant information to the functional entities where it is needed.

The next three sections present an overview of current DBMS. Rather than concentrating on one or two specific examples, we shall describe the general categories of DBMS and discuss the merits and drawbacks of their approaches to the problem.

1.3 Integrated Database Management Systems

Research on DBMS has centered on three broad categories - integrated database management systems, distributed database management systems, and management information systems. These categories are not clearly distinguishable. A particular DBMS often carries the characteristics of more than one category. Integrated and distributed database management systems represent two approaches of configuring or transposing real world database systems to fit computer technologies. Management information system is a much abused name for any DBMS that assists in the making of management decisions by providing statistical or filtered information analyzed from the data.

Let us pretend for the moment that these terms have universally accepted definitions, so that we can gain some insight into our problem area by examining the merits and shortcomings of each category of systems.

The emergence of the integrated database management system was made possible by the introduction of third generation computers with large-capacity direct access secondary storage devices. An integrated DBMS manages a small number of independent "integrated" databases created by consolidating the larger number of databases in the traditional database system. This consolidation process is known as database integration.

There are two principal motivations behind database integration:

(1) Reduction of data redundancy.

(2) Creation of a cohesive, consistent, comprehensive database system.

In a traditional database system, the same information is often stored redundantly in several databases. This is done for two reasons: First, the traditional information system lacks the capability of automatically cross-referencing multiple databases to generate desired information. So if a particular piece of information is needed in two functional areas, it is stored redundantly in two databases. Second, a traditional information system lacks the capability of remote data retrieval, so if a particular piece of information is needed at several scattered locations, a copy must be stored at each location to keep access time down.

Controlled redundancy of stable data, such as manufacturers' product lists or automobile spare part specifications, is generally tolerable. Keeping multiple copies of dynamic data updated and consistent, however, is an expensive proposition and increases the probability of errors.

The following example shows how database integration reduces data redundancy: Two databases A and B, with different user groups, each contain the same item of information d. The logical data structures that represent d in A and in B may be different.

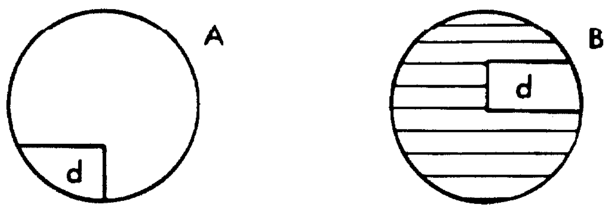

Figure 1.2.1 - Original databases A and B.

They are merged into a new integrated database AB in which d is no longer duplicated. This integration process forces a restructuring of the logical organization of not only d but also the rest of A and B.

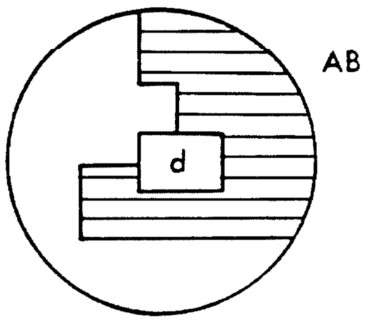

Figure 1.2.2 - Integrated database AB.

Each user group has to learn a new view, or subschema, of the logical data structures corresponding to its original database.

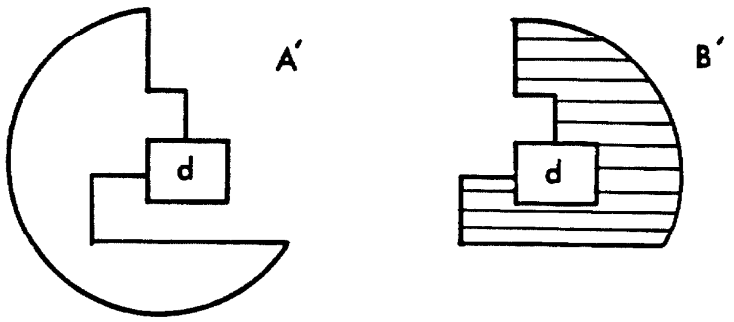

Figure 1.2.3 - Subschemata of AB.

The proponents of integrated DBMS technology recognize that the traditional databases in an organization are interrelated. Lacking the means to express the relationships between distinct databases, they seek to merge them into one integrated database in which the data organization itself reflects these relationships. Indeed, a stated goal of database integration is the creation of a database system consisting of a single comprehensive, non-redundant, computerized database in which all members of the organization can go fishing for information [Martin 76].

The motivations behind database integration are unquestionably valid. Yet actual installations of integrated DBMS have been plagued by difficulties. The following are some of the inherent problems behind this approach:

(1) The reconciliation of the diverse requirements of the many user groups of an integrated database is a costly and time consuming task. Techniques exist to automate part of the labor [e.g. Raver and Hubbard 77], such as identifying clashes in nomenclature, but the removal of inconsistencies results in unsatisfactory compromises.

(2) The integrated database system, consisting of one or at most a few large databases, is not modular. Any change in its logical organization brings the whole system to a halt.

(3) Data security is difficult to maintain in an integrated database. Nothing short of explicitly identifying the access rights to each data item can prevent a user from accessing parts of the database outside his own functional domain.

(4) Since an integrated database does not belong exclusively to any one user group, management control must be given to a database administrator who is not finely tuned to the working environment of each group.

(5) Finally, database integration affects organizational philosophy. As Figures 1.2.1 and 1.2.3 show symbolically, not only the names of data items but also thier relationships to one another may be changed. This wholesale restructuring of the database system puts management and staff alike on uncertain grounds, and disrupts organizational operations.

1.4 Distributed Database Management Systems

Distributed database management systems became a popular research area in the late 1970s, in part pulled by the inadequacies of integrated database management systems, and in part pushed by the availability of minicomputer and network technologies.

A distributed database management system comprises databases from a single information system distributed over multiple computing facilities. As the generality of the definition suggests, the term covers a broad spectrum of system organizations. Let us look at two types of distributed database management systems at opposing ends of the spectrum. We shall refer to them as homogeneous distributed DBMS and heterogeneous distributed DBMS.

A homogeneous distributed DBMS comprises a single database spread out over geographically distant installations, or nodes, connected by a telecommunication network. Some examples of homogeneous distributed DBMS are airline reservation systems, banking and financial systems, and stocks and commodities systems.

Figure 1.3 shows the basic structure of a homogeneous distributed DBMS.

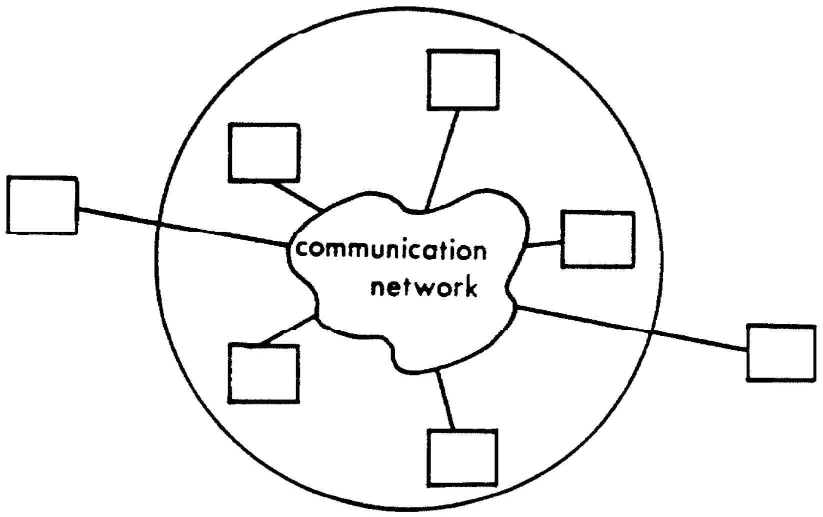

Figure 1.3 - A homogeneous distributed database system. The circle in the diagram represents a database. Boxes represent computer installations. Notice that nodes that do not contain a part of the database may receive information from it nevertheless.

The homogeneous distributed DBMS performs well in an important but restricted set of applications. The data organization within every node conforms to the overall distributed database structure, or schema. This schema is kept static to minimize the difficulty of coordinating multiple-site management. The amount of data that can be transmitted in each transaction is limited by the bandwidth of the communication network.

We are able to design and install successful homogeneous distributed DBMS because the problems involved are technical rather than managerial. Such systems deal with only one facet of the information requirements of the hierarchical organization.

A heterogeneous distributed DBMS comprises multiple databases in multiple nodes. Figure 1.4 shows its basic structure.

In theory, the databases in each node are constructed and maintained independently by the local user groups or their data processing staff. Varying amounts of organizational control are exerted over choice of hardware, software, and database organization to maintain compatibility among nodes in the system. Completed nodes are linked through a communication network.

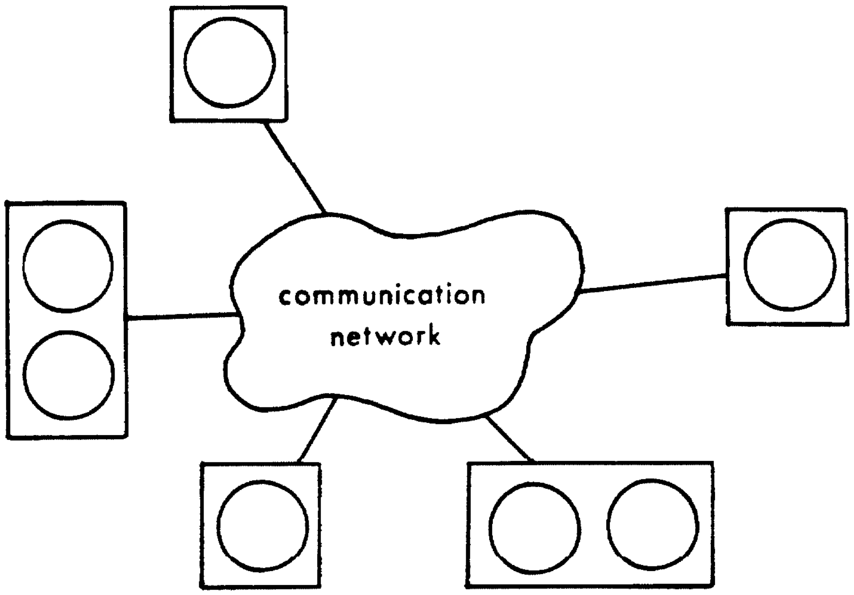

Figure 1.4 - A heterogeneous distributed database system. Circles represent databases. Boxes represent computer installations.

The availability of minicomputer and network technologies has made the heterogeneous distributed DBMS economically feasible, but the principal motivation behind the installation of such systems is the desire to overcome the flaws of integrated DBMS.

Since no merging of databases is required in the construction of a heterogeneous distributed database system, the problems of awkward compromises and complex tradeoffs in data structures and nomenclature do not arise. Since it is modular by nature, it can be installed gradually, spreading the cost over a period of time. Data security is improved with databases residing in physically separate installations. Management control of the databases is delegated to the user groups involved, rather than concentrated in the hands of a database administrator. Most important of all, organizational philosophy is unaffected.

Unfortunately, the heterogeneous distributed database management system evades the very issues which motivate database integration - namely the reduction of data redundancy, and the construction of a cohesive system explicitly reflecting the interdependencies among databases.

Generally, a heterogeneous distributed DBMS provides the technical means of communication between databases, whether they reside in the same node or in separate nodes. With remote data retrieval capabilities, this removes the necessity of maintaining multiple copies of the same database to keep access time down, unless the nodes involved are so far apart that network bandwidth restricts the amount of data that can be transmitted. However, it does not address the issue of how to consolidate data items stored redundantly in several databases that correspond to different functional areas.

The heterogeneous distributed DBMS does not incorporate a systematic means of expressing the interrelationships among its databases. Instead, individual system implementors and user groups are warned of the perils of uncoordinated, decentralized heterogeneous distributed database system construction, and left to their own devices [e.g.

Lorin 79]. The resulting database system is frequently anarchic, with interrelated databases in different nodes having incongruent structures.

1.5 Management Information Systems

Management information system (MIS) is a controversial term that occupies a prominent position in management science, even though it receives less attention in current computer literature. As its name suggests, such a system places its emphasis on the management aspects of data processing, especially in hierarchical organizations.

A management information system is a computer system that consists of several levels of databases; the contents of upper level databases, designed to answer non-routine queries for decision making, are made up of summary information from lower level databases, designed for routine operations.

The common objectives of management information systems - to provide managers with summary information for decision making, to spare them from an overload of irrelevant data, and to respond to non-standard queries that cannot be predicted - are certainly worthy. Yet few such systems have been installed, and those that were installed did not perform well.

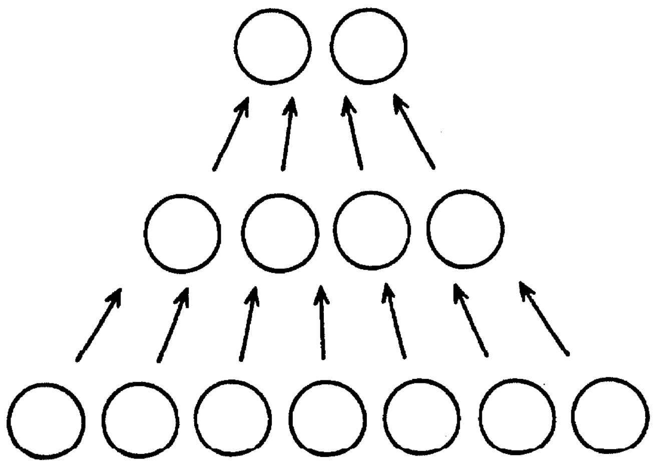

Figure 1.5 - A management information system. (Arrows represent information transfer operations.)

Management information systems on the whole fail to meet their stated objectives. It is difficult to keep databases consistent through different levels because discrete operations must be invoked to transfer information upwards. It is impossible for upper level databases to respond to non-standard queries because they are generated by standard data transfer operations. It is difficult for managers, who are often distrustful of purely summary information, and like to check the numbers for themselves from time to time, to access all the lower level databases.

That managers are distrustful of purely summary information is supported by the results of an experiment carried out at the University of Minnesota [Chervany and Dickson 74]. Twenty-two graduate business administration students were divided into two groups in a simulated decision making environment. One group was given only data that was summarized through the use of simple descriptive statistics. The other group was given only raw data. Decision makers who were given the summarized data (1) made better decisions than those receiving the same data in their raw form, (2) had less confidence in the quality of their decisions, and (3) took longer to make their decisions.

Even though existing management information systems are too static and inflexible to meet the demands of a modern organization, a multilevel database management system that digests information in steps is an accurate reflection of the hierarchical nature of the organizational information system.

1.6 Some Goals for a Database Management System

From the discussion of current DBMS, we have gained some insight into the desired properties of a database management system for a hierarchical organization from a system designer's perspective. Let us summarize those properties:

1. Elimination of data redundancy.

2. Maintenance of data security.
3. Modularity in database system structure.
4. Ability to summarize and analyze information.
5. Versatility to respond to non-standard queries.
6. Flexibility to accommodate changes.
7. Reflection of organizational philosophy.
8. Incorporation of a systematic means of expressing natural database interrelationships.

These properties contribute to give the user a responsive database system. From the user's point of view, the desired properties of a database management system are expressed somewhat differently. He wants to be able to obtain required information quickly. He wants that information to be concise, current, accurate, and presented in a familiar form. He wants to be able to manipulate and analyze that information.

In addition to having the above-mentioned properties, we would like design a database management system that is generally applicable hierarchical organizations. Since the information system structures these organizations are highly individualistic, we must provid a flexible set of tools for the construction of database systems r er than a pre-packaged system ready for installation.

Before we can design such a database management system, we have to have a good understanding of the nature of the organizational information system. After some initial work on developing a model that embodies the database interrelationships, it became apparent that first hand experience would be needed.

1.7 A Case Study

In the spring of 1978 I arranged for a series of visits to the Chemicals Plastics Research Division (CPR) of the Upjohn Company in Torrance, California. With the help of Robert Chess, who took part in the project for undergraduate research units and whose father was a manager at CPR, we interviewed twenty people and took copies of a large quantity of non-classified data. The people we interviewed included secretaries, staff from each department, and all the managers. In exchange for these interviews, we submitted a report to CPR advising them on the development of their computer system [Yu, Chess 78].

CPR, with 150 employees, manufactures a wide product line that includes styrofoam, foam rubber, and urethane compounds. It was an independent company before its purchase by Upjohn. As such it functions much as an autonomous hierarchical organization, with its own administrative, research and development, production and quality control, financial, and marketing and sales departments. Yet it is also part of a larger organization, conforming to certain corporate-wide rules, and communicating with other divisions. Hence it was an ideal subject for our studies.

At the time of our visit CPR was suffering badly from a computerization project that had begun a year earlier. So we had the opportunity to observe the clash between traditions and database management system first hand. We listened to complaints from all levels of CPR, with the exception of the divisional vice president who had initiated the project. People complained that computer databases did not contain the information that they wanted or in the form that they wanted, that inventory records were a week behind time, and that terminal response times for routine queries ranged from a minute to half an hour.

Completely distrustful of computer records, the staff at CPR maintained traditional databases, mostly in the form of multiple-stenciled forms (called pencil copies) which were filled out by hand and collected in large loose-leaf binders. Computer records were updated after a time lag with data extracted from these pencil copies. Together with personal "bootleg" records, these traditional databases formed the actual sources of working information.

The only useful products of the computer system were batch listings of weekly and monthly reports, on which people penciled in corrections and updates. They constituted a poor return for a year's work by a computing department that consisted of a Honeywell 61/60 minicomputer, four disc drives, two tape drives, eight terminals, and a technical staff with five full time employees.

A model of the organizational information system, developed with the insights gained at CPR, is presented in Chapter 2.

CHAPTER 2

THE HIERARCHICAL ORGANIZATION MODEL

To design a database management system that is responsive to the informational needs of its intended users, we must first understand the environment in which they work. To that end, we shall describe a model of the hierarchical organization information system.

This model is based on equal parts of observation at CPR, analysis, and intuition. Like most models of human behavior, it is limited in scope and simplistic. Nevertheless, it provides us with a conceptual frame of reference against which system design alternatives can be evaluated.

A major inadequacy of the model is that it was created without the benefit of experimental experience. An ideal procedure for the development of such a model would have been iterative. A version 1 model would be used to design a commercial quality database management system. The installation and operation of that system in an actual hierarchical organization would provide the experience necessary to design a version 2 model, and so on. Such an undertaking is beyond the scope of university research. However, this inadequacy does not invalidate the usefulness of the model, or the motivations behind its creation.

2.1 Working Groups

The basic functional entity within a hierarchical organization is the "working group", or group for short. A working group consists of a number of people who work jointly to perform the tasks in a functional area. The following examples show how working groups are related to organizational structure.

Organizational structure is usually represented by the organizational chart. Simple organizational charts, or segments of organizational charts, will be used to illustrate our examples. In an organizational chart, each box represents an administrative unit as well as its manager. The tree below it represents its subdivisions.

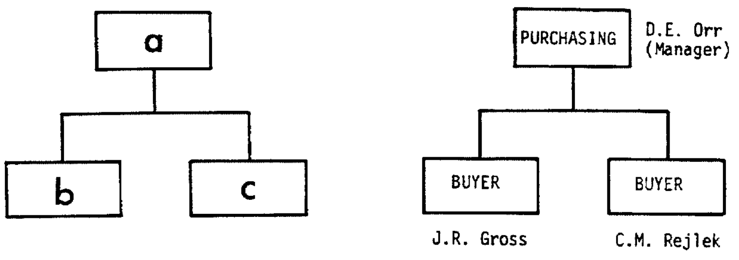

Figure 2.1 - Simple Organizational charts.

Our first example is the management group. It is a group of managers who meet together to discuss administrative and operational matters. In Figure 2.2, M is a management group consisting of the head of division a and the heads of three of its subdivisions. The manager of b belongs to M as well as the lower working group B comprising of members of his own subdivision. With an understanding of both the global considerations within division a, as well as the operational details of his own subdivision b, he forms the human interface between M and B.

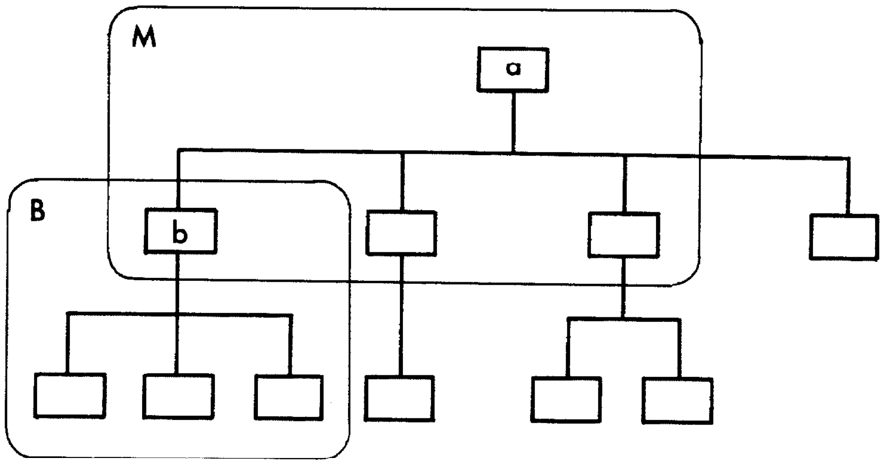

Figure 2.2 - A management group M intersects with a lower group B.

We may extrapolate the above example, imagining that the head of division a belongs also to a yet higher management group at the corporate level, or that the project supervisors who report to the manager of b oversee their own lower working groups. This hierarchy of intersecting working groups form the management chain of command that coordinates the activities within the organization.

Our second example is the subgroup. People who form a working group to perform a set of tasks further divide into subgroups to deal with different functional aspects of those tasks. For example, the Quality Assurance group at CPR has three subgroups - the Physical Testing group, the Chemical Analysis group, and the Inspection group.

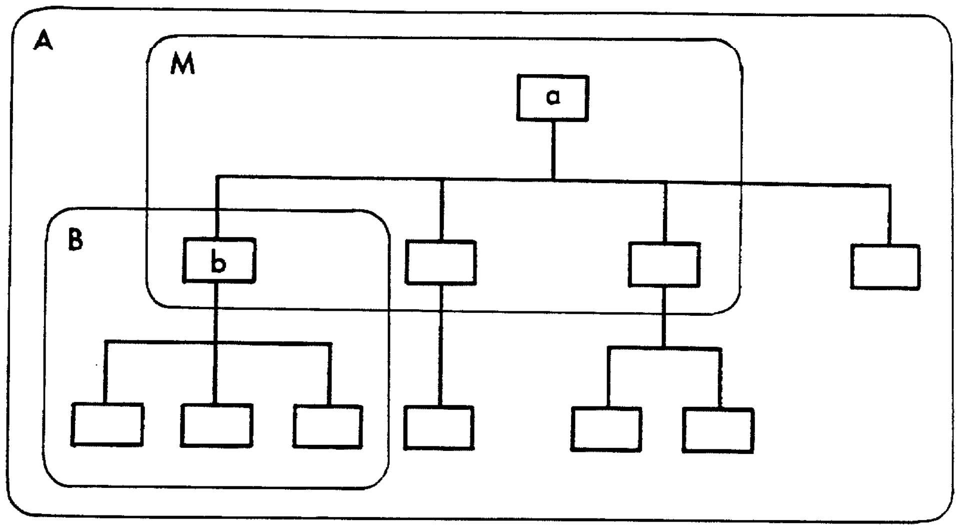

Figure 2.3 - M and B are subgroups of the group consisting of all members of division a.

The entire organization can be thought of as a working group. All other working groups are its subgroups. At the other extreme, each individual is a one-man working group that is a subgroup to a number of larger groups. Hence the working groups in a hierarchical organization form a partial ordering of subgroups. The size of the smallest group to which two people belong determines the closeness of their working relationship.

Our final example is the task group. A task group is formed to deal with a specific task, often temporary in nature, that does not belong to a routine functional area. The members of a task group are borrowed from other groups, either on a part-time or a full-time basis. The CPR committee formed to plan and oversee the installation of its database management system was an example of a task group.

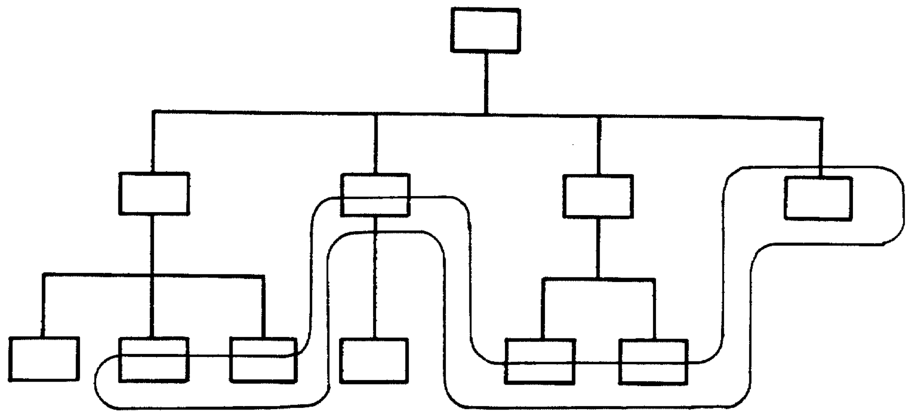

Figure 2.4 - A task group that cuts across the organizational chart.

2.2 Contexts

Each working group functions within a unique working environment, or "context". It has its own language of discourse, operational procedure, and a way of viewing the world for decision making. A newcomer must become familiar with this context before he is able to participate fully as a group member. Within this context, group members can operate efficiently through concise communication. Words take on special connotations, and details need not be spelled out for fear of misinterpretation.

Since working groups overlap, their contexts overlap also, forming a context hierarchy within which members of the organization communicate with one another. The context of a working group forms part of the context of each of its subgroups. The broadest and least specialized context underlies the entire organization; it is an integral part of every working group.

Figures 2.5.1, 2.5.2, and 2.5.3 show the correlation between working groups and their contexts in some simple situations. Contexts, delineated by dotted lines, are shaded variously to show where they intersect. Notice that communication between groups, represented by double arrows, occurs in the intersection of their contexts.

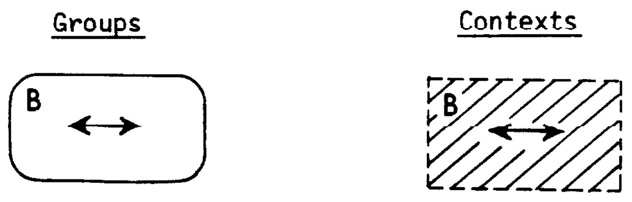

Figure 2.5.1 - Members of a group communicate within its context.

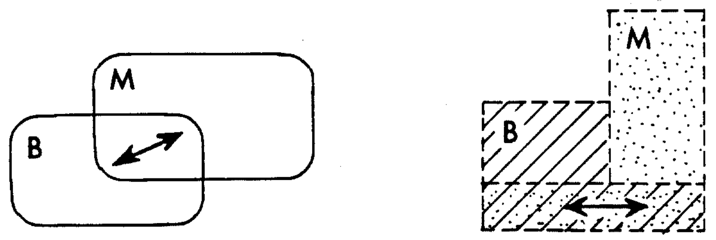

Figure 2.5.2 - Overlapping groups communicate within the intersection of their contexts.

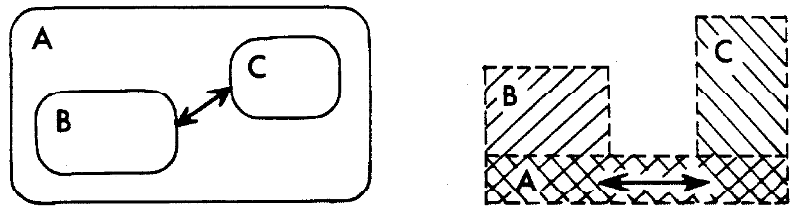

Figure 2.5.3 - A′s context forms the environment in which its subgroups communicate.

2.3 An Example

Figure 2.6 shows the working groups associated with a simple organizational structure. A, B, and C consist of members of a, and of its subdivisions, b and c. M is the management group comprising their managers.

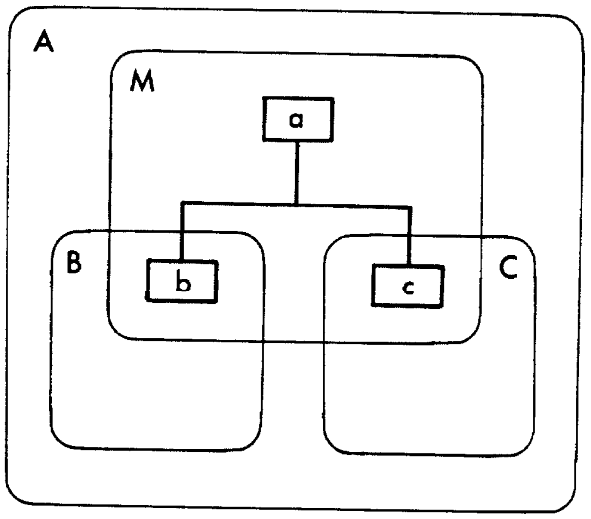

Figure 2.6 - Four working groups associated with a simple organizational structure.

Figures 2.7.1 and 2.7.2 show two ways to conceptualize the context hierarchy corresponding to the example above. Notice that the context of A is the intersection of the contexts of all its subgroups.

Let us introduce a new terminology. A "contextual region" corresponds to a single building block in Figure 2.7.2. For example, the context of A consists of a single contextual region, and the context of B consists of three separate contextual regions.

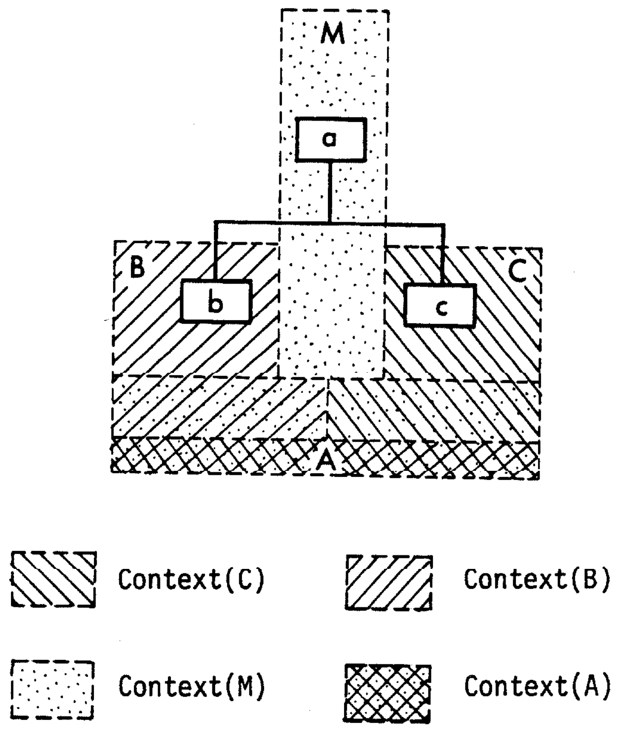

Figure 2.7.1 - The context hierarchy corresponding to the working groups shown in Figure 2.6.

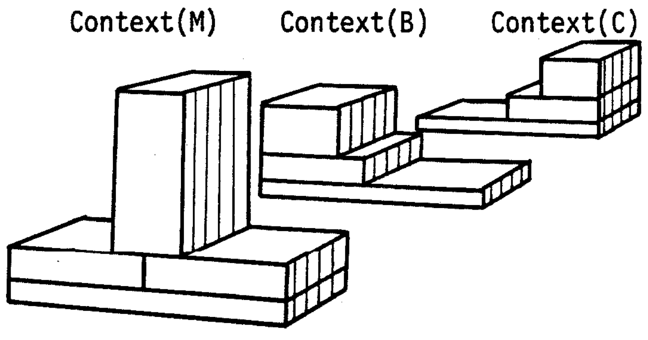

Figure 2.7.2 - The "building block" representation of the context hierarchy shown in Figure 2.7.1. The lowest building block represents the context of the working group A.

2.4 Characteristics of the Context Hierarchy

A context hierarchy is composed of distinct contextual regions that are "based" one on top of another; each one is an extension of the one below it. For instance, the common context that underlies the entire organization is the lowest contextual region in the hierarchy. It includes the operational rules and regulations that apply to everyone in the organization, the vernacular of the trade that they share, and such non-sensitive data as the personnel directory. The contextual region that is exclusive to the members of a division is based on this common context. Its operational procedures, language of discourse, and data are extensions of, and complies with, those of the common context.

Let us look at a simple graphical example. Figure 2.8 shows the building block representation of the context of the group B exactly as shown Figure 2.7.2. The three blocks or contextual regions are numbered for reference. The lowest contextual region, 1, corresponds to the context of the working group A. The middle contextual region, 2, is based on 1. It corresponds to the context that is shared by only the subgroup B and the management group M. The highest contextual region, 3, corresponding to the exclusive context of B, is in turn based on 2.

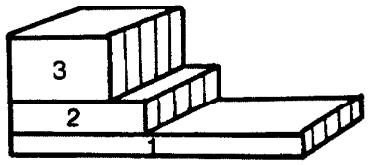

Figure 2.8 - A hierarchy of contextual regions.

When members of the working groups B and C communicate, they do so within contextual region 1. It covers the common aspects of their functional areas without including any operational details specific to either group. Compared to contextual regions 2 and 3, it is broader in scope, contains less details, and is more stable in structure.

When members of M and B communicate, they do so in both contextual regions 1 and 2. Contextual region 2 covers the functional area of B that is of concern to the management group M.

The context of B includes all three contextual regions. Contextual region 3 covers the operational details that are of concern only to members of B. Compared to contextual regions 1 and 2, it is narrower in scope, contains more details, and changes more often.

In the real world, hierarchical organizations have large organizational charts and numerous working groups whose memberships and reponsibilities change continually. Each organization has a unique and dynamic contextual structure that is orders of magnitude more complex than the simple example presented in the last section. However, they have the same essential qualities. We shall refer to them as "organizational context hierarchies" even though formally speaking they are partial orderings rather than hierarchies.

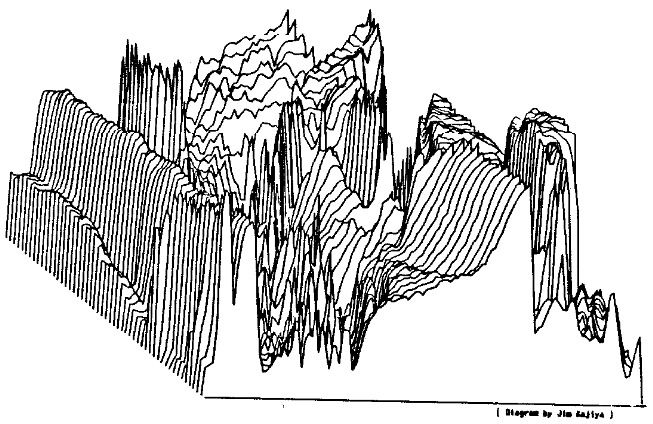

Figure 2.9 - An organizational context hierarchy.

2.5 The Database System

Databases are the tangible components of a working group context. In addition to being information depositories, they also define the functional area of the working group, and establish the responsibilities of individual group members.

The context of a working group generally includes more than one database. Figure 2.10 shows the lines of access from a working group M to several databases within its context, represented by circles.

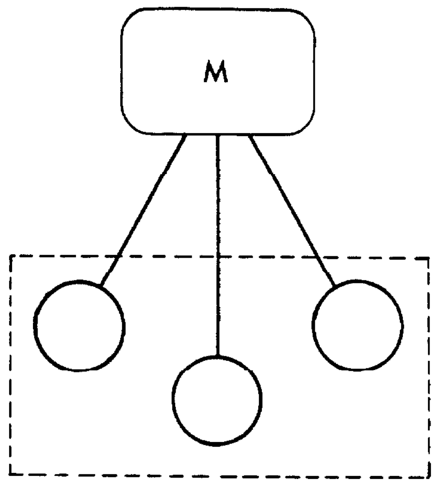

Figure 2.10 - Databases are the tangible components of a working group context.

To see how the structure of the database system is related to the organizational context hierarchy, let us go back to our earlier example. The dotted lines in Figure 2.11 mark the context hierarchy that was shown in Figure 2.7.1. Each contextual region contains a database, shaded to identify the working groups to whose contexts they belong.

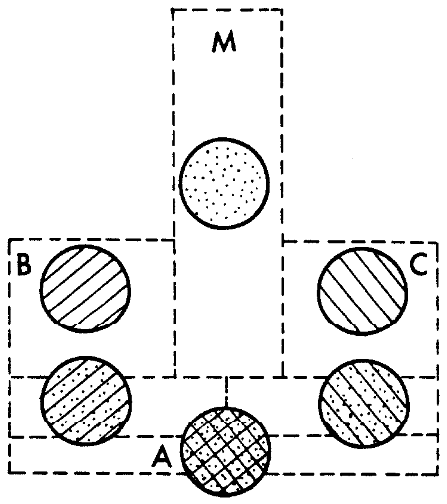

Figure 2.11 - Each contextual region in the hierarchy contains a database.

Notice that the access right to each database is determined by the contextual region to which it belongs. For instance, the lowest database belongs to the contextual region that is part of the contexts of three working groups - M, B, and C. Therefore, it may be accessed by any member of those three groups. By the same reasoning, the highest database is accessible only to members of M.

A database should not cut across two contextual regions. If it were to do so, then it would be divided into two parts with different sets of users, as shown in Figure 2.12. That would make it difficult to maintain security within the database. As we recall, this is one of the inherent problems of integrated database management systems.

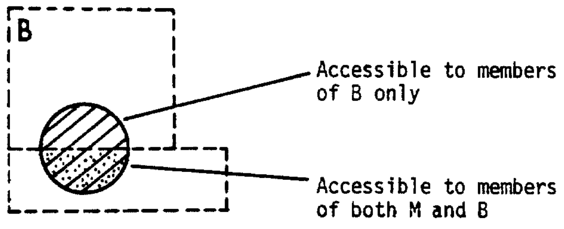
B
Accessible to members of B only
Accessible to members of both M and B

Figure 2.12 - A database that cuts across two contextual regions.

A contextual region may contain more than one database. In principle, it may also contain no database at all. However, as long as it does not affect the behavior of our model, let us assume that each contextual region contains exactly one database.

Figure 2.13 shows the lines of access from the working groups M, B, and C to the six databases in our example, without their shadings. The databases are named for later reference. Notice that each working group can access precisely those databases that lie within its context.

The situation shown in Figure 2.13 approximates the structure of the traditional database system. It can also represent the structure of the ideal heterogeneous distributed database system. The practical difference is that the latter automates information processing, and facilitates access to databases through remote computer terminals.

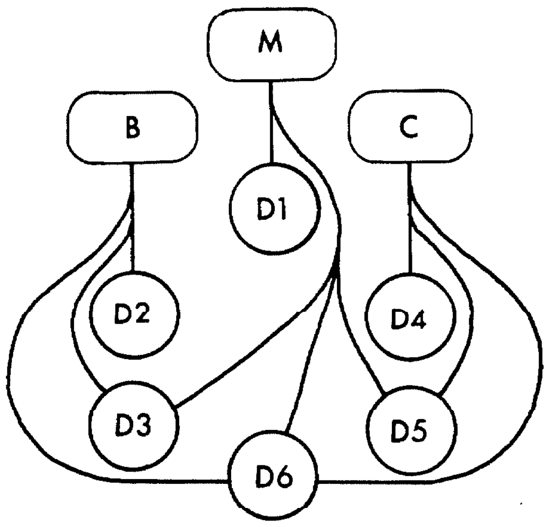

Figure 2.13 - Lines of access from the working groups M, B, and C to the databases shown in Figure 2.11.

Such a system structure has a major flaw. It does not account for the interrelated nature of the databases. Recall that a database is but the tangible part of a contextual region. As contextual regions are based on other contextual regions below them, so databases should be based on other databases. For instance, the contextual region to which the database D2 belongs is based on the contextual region to which the database D3 belongs. Hence, D2 may contain information that is an extension or refinement of the information in D3.

Let us look at a specific example. Suppose B is the Rigid Foam Development group at CPR. The database D3 contains such general information about rigid foam products as their flammability, which the management group M needs to know. D2, on the other hand, contains technical information such as the chemical composition of each product, which only concerns the chemists of Rigid Foam Development. Let us say that a chemist wants to know how flammability is related to the fluorocarbon contents of each product. He may ask the following question: "What is the average flammability of products whose fluorocarbon content is greater than 3%?". Given the system structure in Figure 2.13, he has three options:

(1) He can have someone write an application program to cross-reference the two databases. But then the system is rigid and unable to answer unanticipated questions.

(2) He can query D2 to obtain a list of all the products whose fluorocarbon contents exceeds 3%, write the list down on paper, and then query D3 to find out what is the flammability of each of those products. In other words, he cross-references the two databases manually. This is a laborious process.

(3) D2 and D3 can both contain data on the flammability of rigid foam products. He can then query D2 to obtain his answer. Thus, the information is stored redundantly.

To carry the example further, let us say that C is the Microcellular and Elastomer Development group at CPR, and that M is the management group that oversees research and development. When a member of M wants to compare the flammability of rigid foam and elastomer products, he is faced again with the same three unpalatable alternatives.

The root of this problem is that even though a working group's context is theoretically a single continuous entity, its information is stored in discrete, disconnected databases. Neither the traditional information system nor the heterogeneous distributed DBMS presents a consistent strategy to logically interconnect those databases.

2.6 Basing

How can we improve upon the system structure shown in Figure 2.13? The context hierarchy model suggests a solution.

As a contextual region is based on the contextual region immediately below it, so should its tangible component, the database, be based on the corresponding lower database. Figure 2.14 shows the resulting system structure. A quick comparison with Figure 2.11 shows the equivalence of the contextual region structure and this database structure.

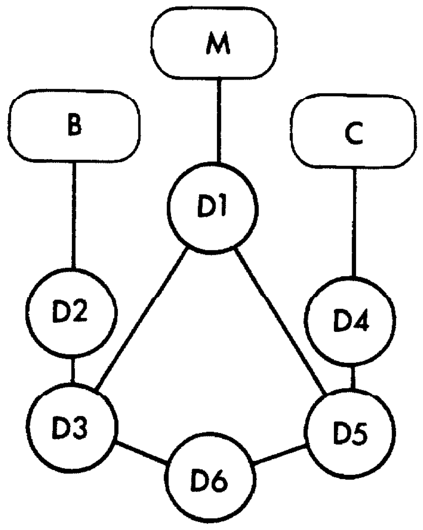

Figure 2.14 - The database basing hierarchy.

What does it mean when we base one database on another? We want the two databases to be logically connected just as they are conceptually related. When D2 is based on D3, we want the resulting union to be logically equivalent to a single, consistent, non-redundant database containing and relating all the information in D2 and D3.

Let us look at it from another angle. The context of the working group B contains three databases named D2, D3, and D6. D3 is based on D6, and D2 is based on D3. We want the resulting union of these databases to be a continuous entity, just as the context of B is a continuous entity.

Since D3 is within the shared context of M and B, it must not be affected by the basing of D2 upon it. The basing of D2 on D3 makes D2 the equivalent of their union, leaving D3 unchanged.

With the basing linkages thus established, the databases D1, D2, and D4 have become the logical equivalents of all the databases within the contexts of M, B, and C. Hence, each of those working groups needs only a single line of access into the database structure - one context, one line of access.

Let us bring back the specific example where B represents the Rigid Foam Development group at CPR. Now if D2 is based on D3, the chemist can obtain the answer to his question from D2, without the redundant storage of information in both D2 and D3.

A formal description of the properties of basing will be presented in the next chapter.

Figure 2.15 presents a more complete view of the basing structure in the larger environment of the organizational database system. D6, the common database of the division a, is based on lower databases whose information it shares with other divisions. Direct lines of access lead from the working groups B and C to the databases D3 and D5. A direct line of access indicates the right to alter the contents of the database.

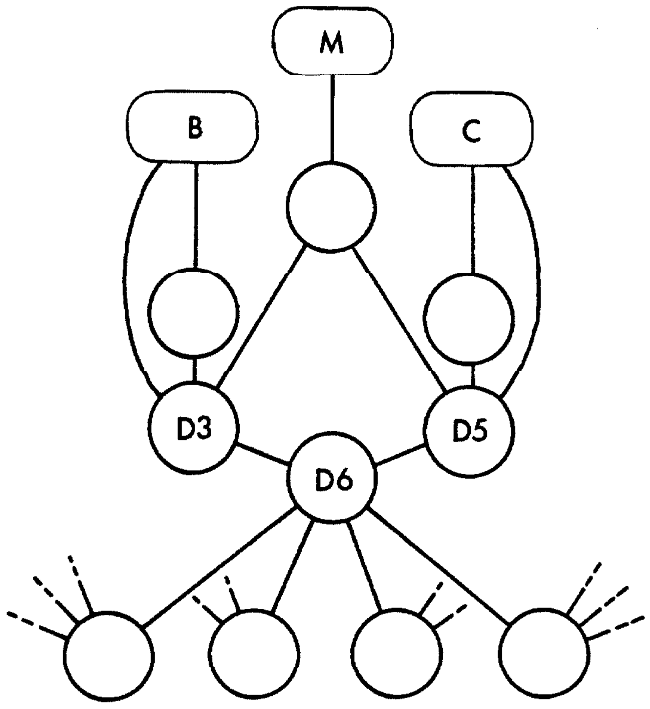

Figure 2.15 - The extended database basing hierarchy.

2.7 Lateral Communication

For the most part, instances of communication tend to fall within the organizational context hierarchy. However, there is an important class of exceptions.

Let us define "horizontal groups" to be working groups whose memberships come from different branches of the organizational chart. There are times when members of two horizontal groups have to exchange some specific items of information. Since the intersection of their contexts is small, such communication frequently occurs outside the normal context hierarchy. This kind of information exchange is called "lateral communication."

Figure 2.16 shows the contexts of two horizontal groups whose memberships come from subdivisions f and g. The double arrow indicates an instance of lateral communication.

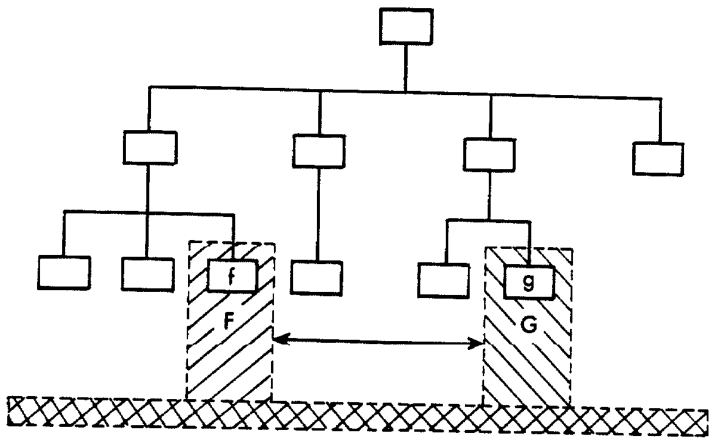

Figure 2.16 - Lateral communication between horizontal groups.

Let us look at a concrete example. The Customer Services group at CPR belongs to the Marketing and Sales Division. The Inventory Control group belongs to the Administrative Division. They are horizontal groups whose functional areas are largely separate.

When a salesperson from Customer Services receives a telephone order, the customer usually wants to know when the goods can be delivered. To answer that question, the salesperson has to find out from the inventory control group whether the order can be filled from existing stock, or the goods have to be manufactured. He is incapable of accessing the Inventory database and interpreting its contents without help because it lies outside his context.

Lateral communication has the following characteristics which distinguish it from in-context communication:

(1) It has a restricted subject matter because the common areas of interest shared by horizontal groups are generally limited.

(2) The information exchange occurs between two people rather than between a person and a database.

(3) The group that supplies the answer obtains the information from its own databases, and presents it to the questioner in a simplified form that the latter can understand. In other words, the supplier group serves an interpretive function.

Figure 2.17 represents the structure of lateral communication. In essence, G obtains information from F's database indirectly by enlisting F's help.

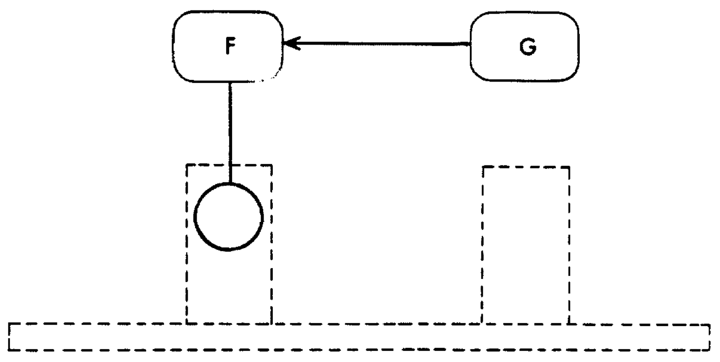

Figure 2.17 - The supplier group F serves as the interpreter of the contents of its database for the recipient group G.

2.8 Channeling

Lateral communication occurs within a narrowly defined context shared by the recipient group, the group that requests and receives the information, and the supplier group, the group that provides the information. For example, when the salesperson from Customer Services asks Inventory Control: "How much Urethane 210 do you have?", Inventory Control understands that he wants to find out how much of that product is available to fill a new order, and not how much of it sits in the warehouse.

This narrow context is called a channel. It is exclusive to the members of the horizontal groups involved. It does not form part of the general context of the working group of which these horizontal groups are subgroups.

Figure 2.18 shows a context hierarchy criss-crossed by channels of lateral communication. This is the essential structure of the hierarchical organization information system in our model.

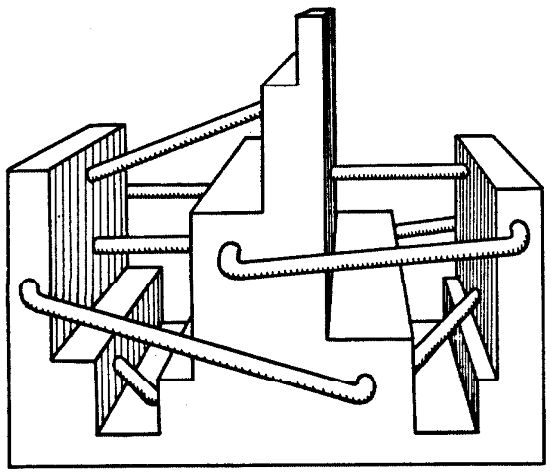

Figure 2.18 - Channels of lateral communication.

Lateral communication is a time-consuming process. How fast the salesperson at Customer Services can find out if an order can be filled from stock depends on several factors beyond his control. If he makes his inquiry by phone, it depends on whether the telephone extension at

Inventory Control is busy. If he walks over to Inventory Control, it depends on the distance he has to cover. In any case it depends on how busy the Inventory Control people are, or how much they like to talk.

We can speed up lateral communication many times by automating the database interpretation process and making it part of the database management system. Figure 2.19 shows how this is done.

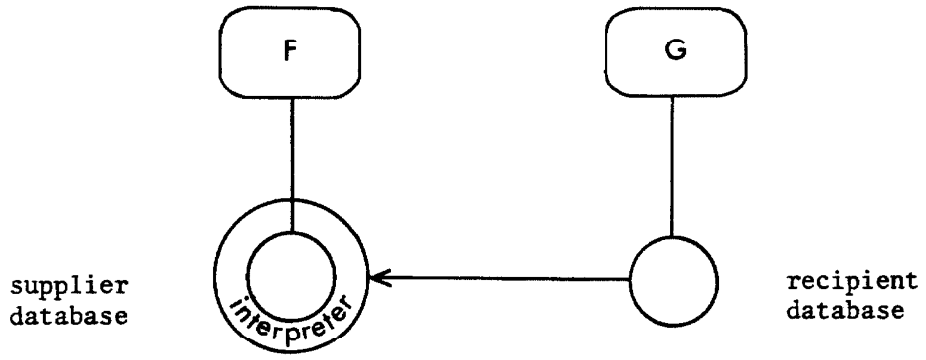

Figure 2.19 - Channeling.

Notice that we have replaced person-to-person information exchange with database-to-database information exchange. Since G cannot access F's database directly, a line of access is established from one of G's databases to the interpreter of F's database. This operation is called "channeling". We say that G's database has a channel to F's database. G can now access F's database indirectly through his own database rather than through the people in F.

2.9 Remarks

In this chapter we have designed a model of the hierarchical organization information system. This model abstracted the structure of the database system and the ways in which people obtain information from it. It also showed some of the defects of the traditional information system, and suggested ways of amending them.

All the defects of the traditional information system have the same root cause: its databases do not reflect the relationships within the context hierarchy. The context hierarchy - the environment in which members of the organization function - is a continuous entity whose parts are closely coupled. The traditional information system, and for that matter current database management systems, use the database as a concrete means of representing parcels of information within the context hierarchy. However, they do not offer any systematic means of representing the interrelated nature of these parcels of information. In short, we have databases that are related, but have no means of expressing those relationships.

The model showed two types of relationships between databases - the basing relationship and the channel of lateral communication. A database management system that incorporates these relationships would more accurately reflect the nature of the organizational context hierarchy. As we shall see in the following chapters, this affinity to the contextual structure of the organization results in a database management system that satisfies the criteria for a good DBMS that we have set in Chapter 1.

CHAPTER 3

COMMUNICATIVE DATABASE MANAGEMENT SYSTEM

The hierarchical organization model described in chapter 2 suggested a database system consisting of numerous distinct databases linked by two types of binary relationships. Databases are based on one another to form a tightly coupled hierarchy; channels for lateral communication cut across this hierarchy to link databases from different branches. The next step is the design of a database management system that accommodates these relationships. Because it allows databases to communicate with one another, it is called the Communicative Database Management System, or CDMS for short.

CDMS is designed to be a general purpose system able to meet the database management demands of any hierarchical organization. As each organization has its own unique informational structure, CDMS does not include a pre-packaged database system ready for instant installation and use. Instead, it provides the necessary tools for the modular construction of a computerized database system tailored to the individual organization.

CDMS consists of three parts: single database management capabilities, the basing operator, and the channeling operator. Single database management capabilities are provided by the REL/POL System of which CDMS is an extension. The next chapter describes the relevant features of REL/POL. This chapter deals with the basing and channeling operators. Together, these are called linkage operators, because the effect of invoking them is to link databases together.

All the examples are given in REL/POL English, the principal query language in the REL/POL System.

3.1 The Basing Operator

The basing operator establishes a basing relationship between a superior database A and a subordinate database B; we say that A is based on B. The operation gives A access to B's current informational structure. Once the basing relationship is set up, A becomes the conceptual equivalent of a single database A′ that is the union of both A and B.

Any changes in B's contents are reflected instantly in A. This contrasts with the standard management information system which requires discrete operations to transmit informational changes to upper level databases. Unlike database integration, the basing operation does not impose changes in the schema of the subordinate database B.

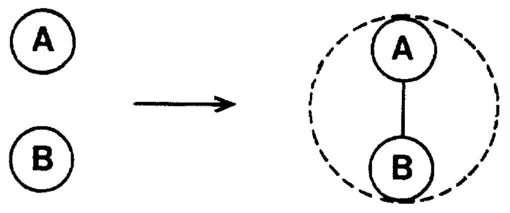

Figure 3.1 - The basing operation.

The following sequence of examples illustrate the usage and properties of the basing operator. Queries to the subordinate database B are listed on the right hand side of the page; queries to the superior database A are listed on the left hand side. Queries are interrupted by comments. To avoid repeatedly saying "the user of a database" or "a user working with the database", databases are treated as animate objects actively communicating with each other.

Example: A database B, containing information about a set of objects called ships, authorizes basing by another database A. A bases itself on B.

```
          (A)                            (B)

                              ENTER B
                              What are ships?
                                Kittyhawk
                              AUTHORIZE BASING BY A
BASE A ON B
ENTER A
What are ships?
  Kittyhawk
```

3.2 Basing - CDMS as a Management Information System

Basing allows the superior database A to ramify, extend, and analyze its subordinate's information. Continuing our example:

```
          (A)                            (B)

vessel:=CLASS
Ships are vessels.
                              Enterprise:=NAME
                              Enterprise is a ship.
What are vessels?
  Enterprise
  Kittyhawk
```

Notice that the new data structure "vessel" in A is dynamically dependent on the data structures in B. The ability to extend a subordinate's data structures and reflect its current informational content gives CDMS desirable management information system properties. More sophisticated analyses can be done using the REL/POL definitional capabilities:

```
         (A)                              (B)

                              The length of the Enterprise
                                is 2500 ft.
                              What are the lengths of ships?
                                Enterprise    2500 ft.
                                Kittyhawk     1925 ft.
DEF:long ship:ship whose
  length is greater than
  2000 ft.
What are long ships?
  Enterprise
                              The length of the Kittyhawk
                                is 2025 ft.
What are long ships?
  Enterprise
  Kittyhawk
```

3.3 Basing - One Way Flow of Information

A fundamental property of management information systems is that information flows from lower level databases to the upper levels, never the other way around. CDMS has this property. Even though the superior database apparently modifies and extends its subordinate's information, these modifications and extensions affect neither the structure nor the content of the latter. In other words, the basing relationship preserves the integrity of the subordinate.

```
      (A)                             (B)

Hornet:=NAME
Hornet is a ship.
What are vessels?
  Enterprise
  Hornet
  Kittyhawk
                              What are vessels?
                                eh?
                              What are ships?
                                Enterprise
                                Kittyhawk
```

3.4 Basing - Structural Stability of Subordinate Databases

The hierarchical database system rests on the structural stability of the lower level databases. As we have seen, CDMS allows the informational structures of the superior databases to be extensions of, and dependent upon, their subordinates, reflecting the tightly coupled nature of the organizational context hierarchy. This dependency imposes a responsibility upon the subordinate database.

Structural changes in a database, in contrast to changes of information content, reflect changes in the way people think. They affect not only the working group to which the database belongs, but also every other working group that receives information from it.

Structural changes fall into two categories: additions and deletions. To give the maximum degree of flexibility to the subordinate database, structural additions are allowed. These additions are not instantly reflected in the superior. They are tested in the subordinate to ascertain their usefulness and compatibility with existing data structures. The working groups who access the superior are informed of the pending changes. Eventually the changes are allowed to propagate upward by an explicit new basing operation.

Basing is a discrete operation during which the superior acquires knowledge of the subordinate's structure at that instant. Making basing a repeatable operation gives management groups control over when they would allow changes in the organization of lower levels to show up in their own databases.

There is no way to guarantee the integrity of a superior if we allow structural deletions in its subordinate. Deleting part or all of a subordinate database is analogous to removing a can from the bottom of a stack on a supermarket shelf - the entire structure wobbles or collapses. However, there needs be a mechanism for structural deletion short of wholesale reconstruction of the affected portions of the database system.

Fortunately, the necessity for structural deletions arises infrequently, so that efficiency is not a central concern. The mechanism is provided by an "unbasing" operator which reverses the effects of basing. Notice that unbasing A from B when A has superiors of its own is a specific case of structural deletion of a subordinate database and cannot be allowed. Therefore, to free a database at the lower levels of a basing hierarchy, a sequence of unbasing operations must be involved, starting from the top and going downward. Thus freed, the formerly subordinate database can then make the necessary structural changes.

After changes are made, the part of the basing hierarchy affected must be reconstructed by explicitly reinvoking the basing operator, once for each link, starting from the bottom. Thus the procedure for structural deletion is more involved when the database is lower in the hierarchy. This can be tolerated since the lower level databases tend to be more stable. When a lower level database changes its structure a large number of people are affected. The administrative task of informing everybody concerned overshadows the technical task of rebasing, which can be done automatically by a batch job run after the working hours.

The following is an example, using databases A and B, of the way a subordinate database can make structural deletions.

```
          (A)                           (B)

                              destination:=RELATION
                              Boston:=NAME
                              The destination of the
                                Enterprise is Boston.
What is the destination of
  each vessel?
  eh?
EXIT
BASE A ON B
What is the destination of
  each vessel?
  Enterprise     Boston
                              Delete destination.
                                Deletion not allowed

EXIT
UNBASE A FROM B
                              Delete destination.
                                Deleted

BASE A ON B
ENTER A
What are ships?
  Enterprise
  Hornet
  Kittyhawk
What is the destination of
  each vessel?
  eh?
```

3.5 Transitivity of the Basing Operator

To create a database hierarchy of more that two levels, the basing operator must be transitive. when a database A is based on another, B, it assimilates the apparent informational structure of B. In effect, A becomes the equivalent of a single database that is the union of itself, B, and the entire database hierarchy subordinate to B.

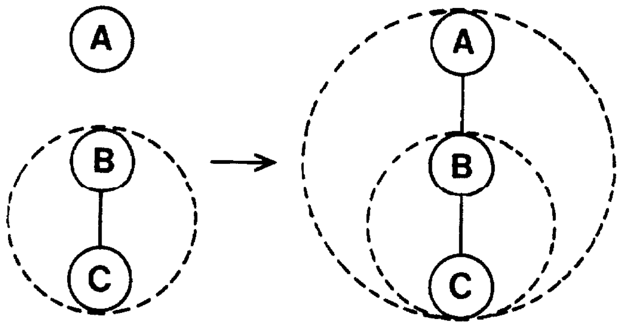

Figure 3.2 - Multiple level basing

The basing operator is not associative; the order of two basing operations influences their cumulative effect. The basing of a subordinate database B on yet another database C is a special case of structural addition to a subordinate. B′s superior A does not automatically acquire knowledge of C′s informational structure.

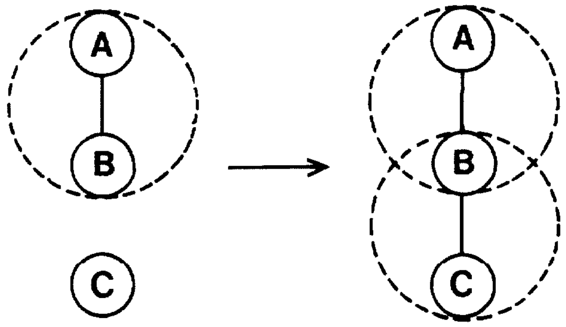

Figure 3.3 - Structural addition to a subordinate by basing

3.6 Basing – Multiple Superiors

A database may have more than one superior. This obvious property has a number of desirable effects:

(1) It allows many users to access shared information concurrently while preserving the integrity of the shared database;

(2) It eliminates data redundancy without forcing reconciliation of the differences in the schemata of all superior databases;

(3) The responsibility of maintaining the single copy of the subordinate can be clearly delegated, ensuring its accuracy and timeliness.

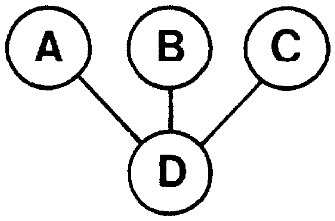

Figure 3.4 - A database with multiple superiors

3.7 Basing - Multiple Subordinates

A database may have more than one subordinate. A person working in a hierarchical organization requires information from many different sources. Basing his own working database on multiple subordinates gives him the equivalent of a single, private, sufficient database containing all the information he needs. This capability not only eliminates the necessity of explicitly accessing each subordinate database separately, it allows the user to combine information from different sources to generate meaningful results.

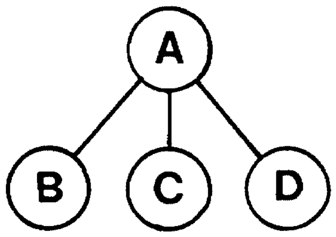

Figure 3.5 - A database with multiple subordinates.

Example: a university database system includes a "faculty" database and a "salary" database. Using a conventional database management system, the following sequence of queries generates the average annual salary of a particular group of professors.

```
ENTER faculty
Who are the associate professors in the Biology Division?
  Henry A. Lester
  John D. Pettigrew
  James H. Strauss
EXIT
ENTER salary
What is the average annual salary of Henry A. Lester,
John D. Pettigrew, and James H. Strauss.
  $29,520.00
EXIT
```

The query sequence would have been more complex had there been fifty associate professors instead of three in the Division of Biology.

Using the Communicative Database Management System, a user bases his personal database on both "faculty" and "salary". To generate the same information, he types

```
What is the average annual salary of associate professors in
the Division of Biology?
  $29,520.00
```

Thus defined, the basing operator allows the construction of arbitrarily complex database hierarchies that correspond to the information system structures of individual organizations. Since the basing linkages can be created, broken, and renewed, the modularity of the database system allows it to be constructed and modified with minimum disruption to organizational operations.

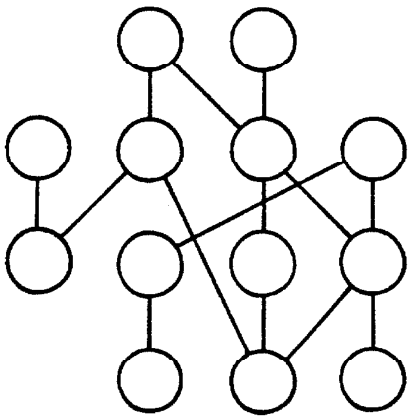

Figure 3.6 - **Example of a database basing** hierarchy.

3.8 The Channeling Operator

The channeling operator establishes a channel for lateral communication from a recipient database R to a supplier database S. We say that R has a channel to S. While basing creates a direct link from one database to another, a channel links the recipient to a protective envelope of the supplier called the "interpreter".

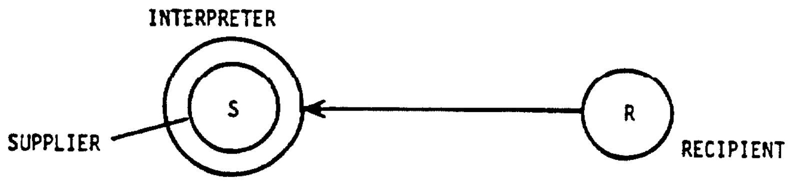

Figure 3.7 - A channel for lateral communication.

The interpreter has two functions:

(1) It interprets the recipient's query into a form that the supplier understands.

(2) It protects the supplier's data from unauthorized or unstructured access by the recipient.

The interpretive function allows the supplier to present a different apparent structure to each database across a channel; it resolves contextual conflicts between horizontal groups while preserving the natural structure of their databases.

An integrated database system, on the other hand, consists of a single database whose structure is described by a schema. Each user has an apparent view of it described by a subschema. These subschemata have to be structurally congruent because they are projections of the same integrated schema. This enforced congruency robs the database system of the individuality and dynamicity so necessary for efficient communication.

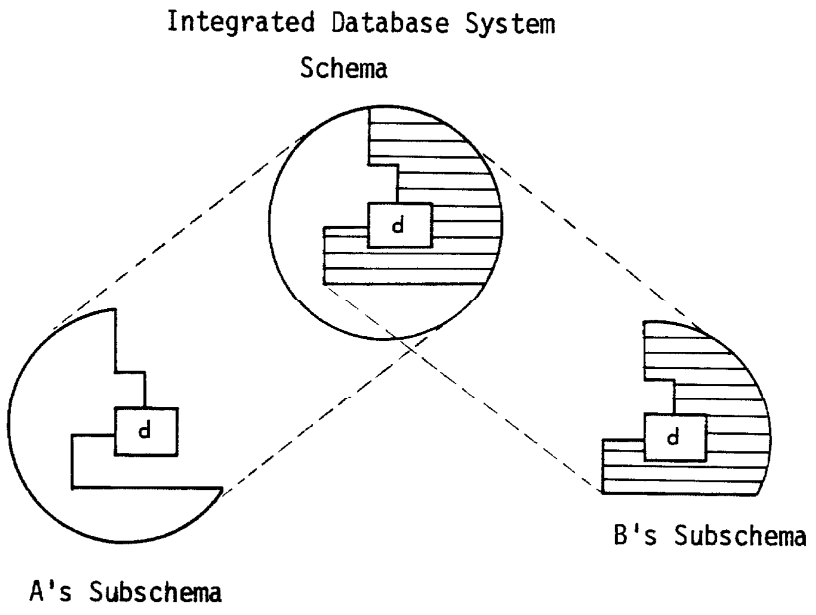

Figure 3.8 - An integrated database system forces its subschemata to be mutually congruent.

The structure of a communicative database system is described by a "map" consisting of the set of schemata of its constituent databases, which do not have to be congruent, and the set of linkages between them. The apparent structure of each database to its user is a consistent, sufficient "subschema" of this map. The channel interpreters allow these "subschemata" to be more than simple projections. Such a system incorporates the database dependencies while allowing constituent databases to retain their individuality and dynamicity.

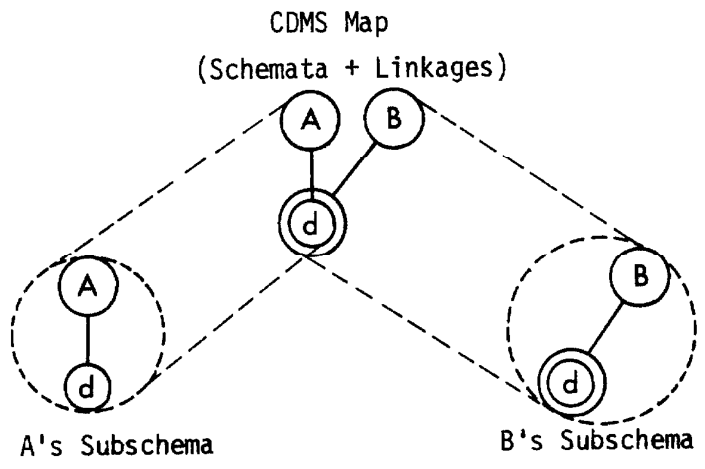

Figure 3.9 - A communicative database system allows its constituent databases to retain their natural structures.

The protective function allows the owners of a database to control what information each horizontal group can obtain from it. For example, a census database may contain detailed information on individuals, but only statistical information is made available. More on this later.

3.9 Channeling - An Example

The following example, mentioned in Chapter 2, was taken from CPR. It illustrates the usage and properties of the channeling operator. The inventory database at CPR is maintained by Irv Botvinik of Materials Management. It contains the following records:

```
LOT(LOT NUMBER, PRODUCT NAME, PRODUCTION DATE,
    QUANTITY ON HAND, ALLOCATED QUANTITY);
SHELF LIFE(PRODUCT NAME, INTEGER);
SHRINKAGE FACTOR(PRODUCT NAME, REAL);
```

Products are manufactured in "lots" of various quantities. "Quantity on hand" is whatever amount that has not yet been shipped. Out of that "allocated quantity" has been assigned to particular orders and is awaiting shipment. Each product has a "shelf life" beyond which it cannot be sold without chemical restoration, and a "shrinkage factor" accounting for product shrinkage.

When someone calls Customer Services to find out how soon 500 pounds of Urethane 210 can be delivered, the salesperson has to find out whether CPR has that amount of the foam mixture in stock. Using CDMS, a channel can be set up between the Customer Services′ working database R and the inventory database S to handle such routine inquiries.

The first step is consultation between Customer Services and Irv Botvinik to determine what information is needed, what can be provided, and the exact language to describe that information. Next the channel is set up through the following sequence of commands to the two databases:

```
(Irv Botvinik:)

ENTER S

DEF:age of lot:number of days between production date of lot
and today

DEF:fresh lot of "Urethane 210":lot whose product name is
"Urethane 210" and whose age is less than the shelf life
of "Urethane 210"

DEF FOR R:available quantity of "Urethane 210":sum of
((1-shrinkage factor)*age*(quantity on hand-allocated quantity))
of fresh lot of "Urethane 210"

EXIT


(Customer Services:)

ENTER R

CHANNEL TO S
```

The channel is set up. Customer Services can now ask about the availability of any product from their database R.

(Customer Services:)

```
What is the available quantity of Isonate 459 spray?
  1852 lbs.
```

3.10 Channeling - Structured Access to Information

A channel provides narrower but more structured access to information than the basing relationship. Basing restricts access to entire databases, while channeling can provide several levels of increasingly severe access restrictions to specific information, as shown in the following sequence of examples.

The supplier database S contains information on aircraft - domestic, foreign, commercial, military; their speeds, seating capacities, physical characteristics, etc. The examples show definitions by the supplier.

(1) Channels can limit access to specific data items:

```
DEF FOR R:commercial aircraft:commercial aircraft
```

The recipient database R can ask about commercial aircraft but not about aircraft in general.

(2) Channels can limit access to specific combinations of data items:

```
DEF FOR R:speed of commercial aircraft:speed of commercial aircraft
```

R cannot ask about the speed of aircraft in general, nor about the seating capacity of commercial aircraft.

(3) Channels can limit access to specific statistical results:

```
DEF FOR R:average speed of commercial aircraft:average speed/
of commercial aircraft
```

(4) Channels can hide the actual data structures from the recipient by making the definiens distinct from the definiendum. Irv's definition of available quantity is an example.

Thus, channels provide high level information security without attaching authorization lists to individual data items. Without the appropriate channel definitions, there is no way to access any part of a horizontal database, so security checks are made unnecessary.

Database security is not the central concern in this thesis, however, and CDMS does not contain defenses against malicious access mechanisms such as the "tracker" [Denning, et al 79]. Nevertheless it has the proper structure for the development of a truly secure system.

3.11 Other Properties of Channeling

So far we have dwelt on the differences between basing and channeling, but really they are both linkage operators, having many common properties. The properties that channeling shares with basing are listed below. Since they are useful in similar ways to their basing counterparts, they will not be discussed in detail.

(1) Transitivity: the channels of a recipient database form part of its informational structure, from which it can define channels for other databases, acting now as the supplier.

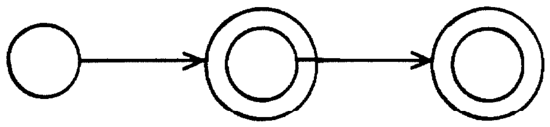

Figure 3.10 - Multiple stage channeling.

(2) Multiple recipients: a database may supply channeled information to any number of recipient databases. In general it presents a different apparent structure to each recipient.

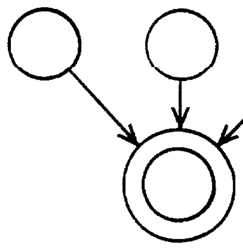

Figure 3.11 - A database with multiple recipients.

(3) Multiple suppliers: a database may have channels to any number of supplier databases.

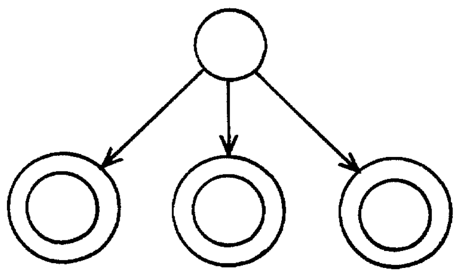

Figure 3.12 - A database with multiple suppliers.

(4) Channeling is a reversible and renewable operation:

Since the intersection of the contexts of horizontal groups is small, channels are a looser form of database linkage than basing. The supplier database can add to its own structure, and have these structural additions reflect across channels by redefining them. The recipient databases need not be notified as long as they continue to receive the information they have contracted for.

Example: The Canadian Sales Department of a T-shirt manufacturer supplies sales figures in U.S. Dollars to the Marketing Department.

```
DEF FOR Marketing Database:price of "tennis shirt":wholesale
price of "tennis shirt"
```

Suppose Canadian Sales Department now decides that internally it should use the Canadian Dollar as the basic monetary unit for accounting. It can then redefine the Marketing Department channel:

```
DEF:Canadian exchange rate:0.9102

REDEF FOR marketing database:price of "tennis shirt":
Canadian exchange rate*wholesale price of "tennis shirt"
```

The channel is now redefined; it still supplies the same information, and the Marketing Department is none the wiser. "Canadian exchange rate" is redefined daily to reflect going money market conditions.

The same restrictions on structural deletion that applies to basing also applies to channeling. The recipient database invokes the "detach" operator to severe the channel and free its supplier for structural deletion. The channeling process can then be repeated to re-establish linkage.

3.12 How to Use the Linkage Operators - A Simple Example

Figure 3.13 illustrates a simplified version of the part of the CPR database system used by the Customer Services Department.

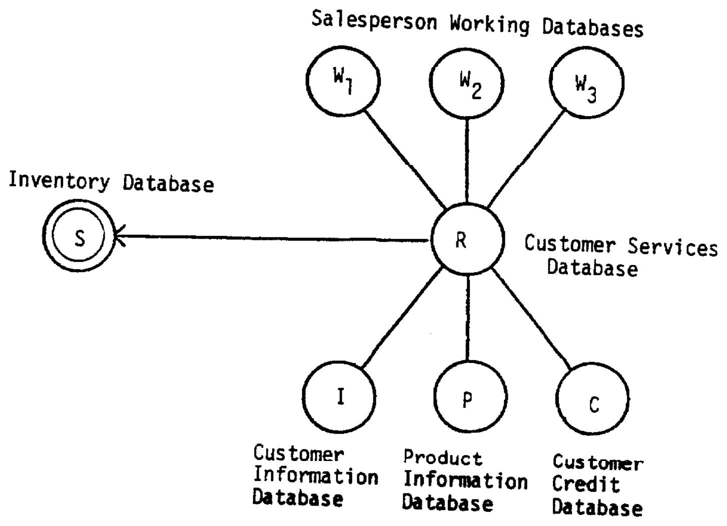
Salesperson Working Databases
W_1
W_2
W_3
Inventory Database
S
R
Customer Services Database
I
P
C
Customer Information Database
Product Information Database
Customer Credit Database

Figure 3.13 - CPR Customer Services Database Subsystem

The Customer Information Database I is maintained by Customer Services. It contains the customers' shipping and billing addresses, their product use records, etc. The Product Information Database P is maintained by the Marketing Department, and contains the technical and pricing information of each product. The Customer Credit Database C is maintained by the Credit and Collections Department. It contains customer credit information. The Inventory Database S is maintained by Irv Botvinik. As we know, it contains information concerning the products such as lot, shelf life and shrinkage factor.

The Customer Services Database R is the common database for the whole department. It is based on I, P, and C, and has a channel to S that includes "available quantity". It may contain further information that concerns only the people at Customer Services. It may also contain summary and analytical operators.

The personal working database Wi of each salesperson is in turn based on R. The apparent structure of Wi encompasses all the information that he might need.

3.13 Communicative Database System Integrity

Multiple user database systems use locks to preserve their integrity during concurrent updates. The area of data locked away during an update operation is called a granule. Locking granularity refers to the size and hence the number of granules within a database system. A system may choose to have fine granules such as pages or records, or it may choose coarse granules such as files.

A simulation study at UC Berkeley [Ries and Stonebraker 79] concluded that coarse granularity is the most efficient, provided that the granules are "well placed". That is to say, the data records are organized in a sensible manner so that each query sentence accesses only a small number of granules. Figure 3.14 is taken from Ries and Stonebraker's paper. It shows that the ideal number of locks is between 10 and 100.

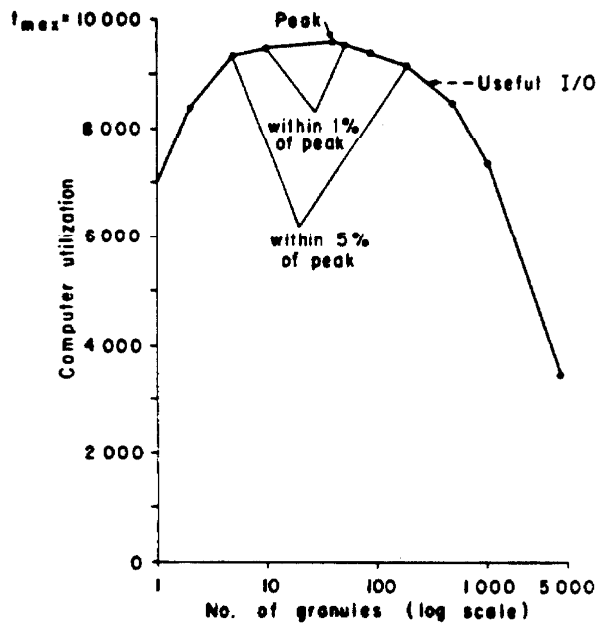

Figure 3.14 - Computer time versus number of granules.

The natural granules of a communicative database system are the individual databases. They are well placed by usage and design. Disregarding personal working databases, which need never be locked, an organizational database system comprises between ten and one hundred databases, giving it ideal granularity.

CHAPTER 4

IMPLEMENTATION

This chapter presents an informal description of an implementation strategy for the linkage operators - basing and channeling. It is intended to be a discussion of some of the more interesting problems encountered during the implementation of CDMS as an extension of the REL/POL System. It is not a comprehensive programming documentation. The principal tool for describing specific algorithms will be graphical depictions of their data structures. These will be supplemented by verbal descriptions of programming logic only so far as the latter are needed.

Since CDMS was designed as an extension of the REL/POL System, its implementation conforms to the conventions and data structures of REL/POL, and depends on many of the latter's capabilities. The first part of this chapter consists of a description of the relevant aspects of the REL/POL System.

Due to its power and generality, many of the data structures in the REL/POL System are complicated. Graphical depictions of these data structures will be greatly simplified so that we can focus on the few features that have direct bearing on the implementation of the linkage operators. These depictions of the data structures will be accurate only to the extent that they correctly describe the associated programming logic.

4.1 The REL/POL System

REL (Rapidly Extensible Language) is an advanced user-oriented database management system developed at Caltech under the leadership of Drs. Frederick and Bozena Thompson [B. Thompson 77]. Its features include general, optimized relational database handling, definitional capabilities, and natural language query facility. It was successfully demonstrated in 1978 and 1979. CDMS has been partially implemented in the REL System.

POL (Problem Oriented Language) is the logical successor to REL. It is being implemented in an extension of the PASCAL programming language on the Hewlett-Packard 9845 desktop computer. It encompasses the communicative database system concepts developed in this thesis. Other POL features that are not part of the REL System include an advanced metalanguage developed by my former colleague, Dr. Gideon Hess [Hess 80], and habitability features such as diagnostic query responses and spelling correction.

REL and POL are sufficiently alike in their kernel program and data structures that we can treat them as a combined REL/POL System for the purpose of explaining the implementation of CDMS.

4.2 REL/POL Paging

A database's programs and data records are stored on "pages". A page is a fixed-sized, addressable segment of disk space. A pointer to a data record consists of three fields:(owner; page; displacement). "Owner" identifies the database which owns the page. Only the owner can alter its contents. "Page" represents the device address of the page. "Displacement" is the displacement between the data record and the top of the page.

In the diagrams in this chapter, pointers are represented either by arrows or by small letters such as a, b, c. To identify the owner of a page explicitly, the small letters are prefixed by the name of the database; for example, A.a, Inventory.b.

To reference a data record the page on which it resides must be copied into main memory. If the main memory copy is altered in any way it must be written out to disk again. This process is called "paging". Because paging is a time-consuming process, its optimization is an important factor in the design of algorithms.

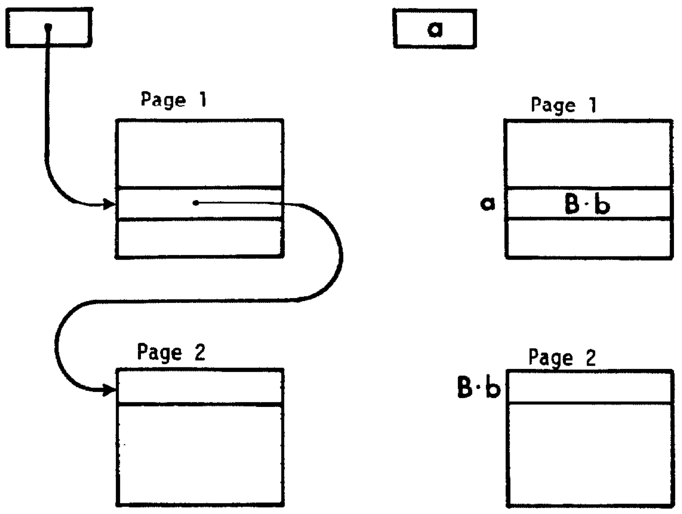

Figure 4.1 - Two ways to represent page pointers.

4.3 REL/POL Data Records

There are four types of data records in an REL/POL database: individuals (e.g. John) are single entities; classes (e.g. student) are sets of individuals; attributes (e.g. parent) relate individuals to one another; number attributes (e.g. age) relate individuals to numbers. Figure 4.2 illustrates the basic structures of these record types.

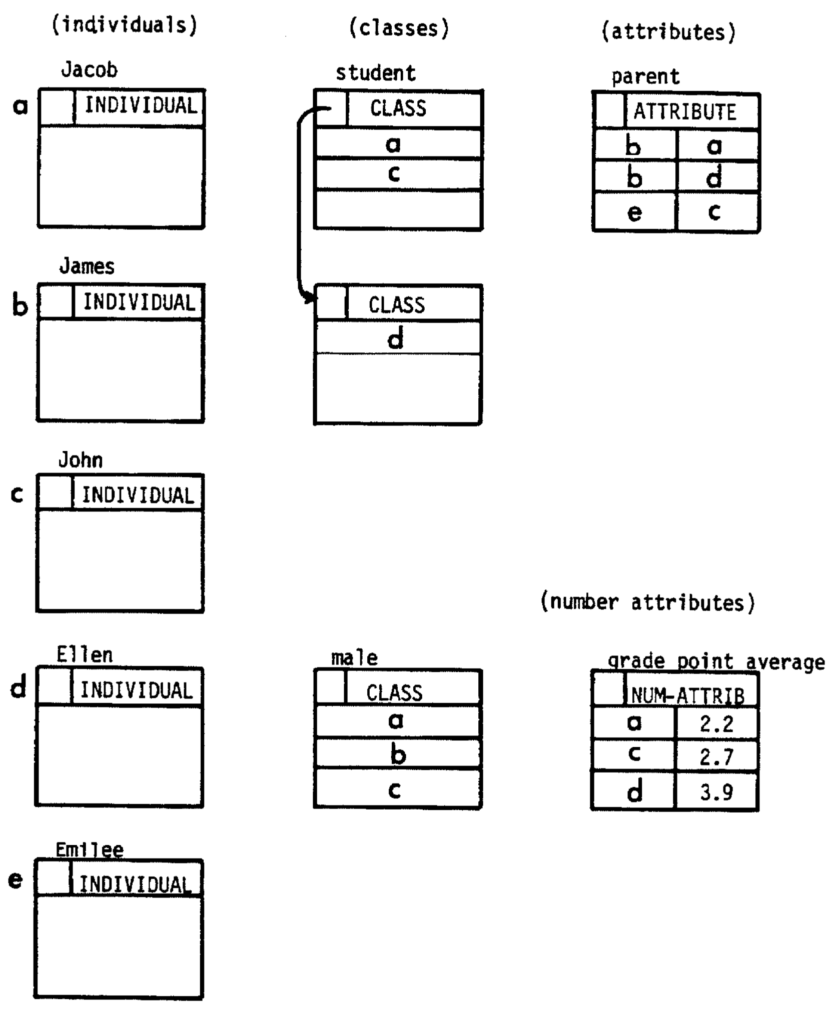

Figure 4.2 - Data record structures. Jacob, John, and Ellen are students. Jacob, James, and John are males. The parents of James are Jacob and Ellen. The parent of Emilee is John. The grade point averages of Jacob, John, and Ellen are 2.2, 2.7, and 3.9. Notice that the class "student" is stored in two linked pages.

Classes and attributes may have dynamic subclasses and subattributes. For example, the attribute "parent" may have a subattribute "father."

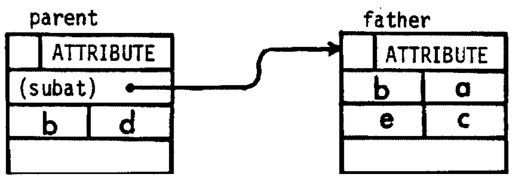

Figure 4.3 - The data structure that represents an attribute "parent" that has a subattribute "father". The "subat" key indicates pointer to a subattribute. Data records for individuals are omitted. Notice that "parent" contains the same relationships shown in Figure 4.2.

4.4 REL/POL Dictionary

The data in an REL/POL database are accessed via a lexicon and a dictionary. The lexicon contains references to data records. The dictionary contains definitions and the grammar of the query language. The implementation decision to have a separate lexicon and dictionary is based on efficiency considerations. The dictionary can be easily adapted to handle lexical items as well as definitions and grammar rules. For the purpose of explaining the implementation of the linkage operators at an abstract level, then, let us assume that only the dictionary exists. Figure 4.4 represents the basic structure of a database under this simplifying assumption.

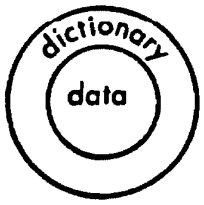

Figure 4.4 - The dictionary and the data are the two indivisible parts of a database.

The REL/POL dictionary is a complex data structure that incorporates such linguistic notions as deep and surface ambiguities, features, case semantics, and type 0(General Rewrite Rule) grammars. We shall limit our discussion to a stripped down version of the dictionary, divested of these sophisticated structures.

A dictionary consists of a number of dictionary entries. Each dictionary entry has a syntactic component and a semantic component. The syntactic component is a rule of the form LHS=>RHS(right-hand-side parses into left-hand-side). The semantic component is contained in a data structure called the "PI-stack". For the time being let us assume that the PI-stack contains a page pointer to either a data record or a semantic routine.

Associated with the dictionary is a program called the "Parser". The Parser parses a query sentence, generating one or more parsing graphs of the sentence according to the contents of the dictionary. Each complete parsing graph represents an interpretation of the sentence.

To understand how a query sentence is parsed, let us look at a simple example: The dictionary of a database includes the following three entries, among others:

```
(1) rule:     <noun>=>"male"
    PI-stack: a=pointer to the class "male"

(2) rule:     <noun>=>"student"
    PI-stack: b=pointer to the class "student"

(3) rule:     <noun phrase>=><noun><noun>
    PI-stack: c=pointer to the semantic routine "class intersection"
```

If the database receives a query sentence that includes the string "male student", the parser would create a parsing graph. Figure 4.5 is an abbreviated representation of the parsing graph, showing only the parts that were produced by the three rules listed above.

Given the above parsing graph as input, another program called the "Semantic Processor" would then call the routine "class intersection" to generate the class in which each member is both a male and a student.

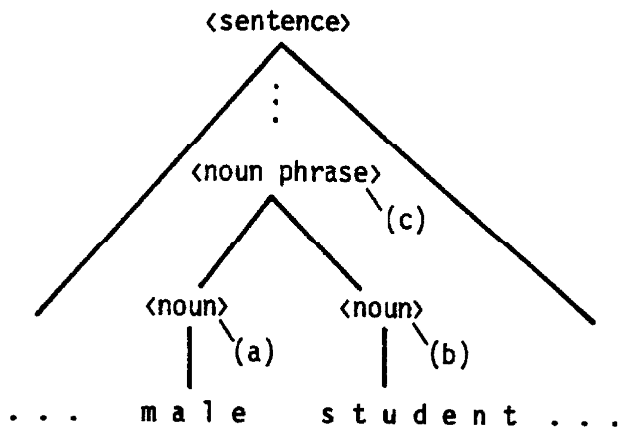

Figure 4.5 - An abbreviated parsing graph.

To implement CDMS we shall have to modify the structure of the PI-stack. Therefore let us look at it in greater detail. Contrary to what its name suggests, a PI-stack is a list. Each list element represents a different semantic meaning. When the list has more than one element, we say that the associated rule is ambiguous. A PI-stack element has three fields: a link to the next PI-stack element, a "type" field, and a "payload" field. The payload field may contain an integer, a real number, a boolean constant, a character constant, or a page pointer to a data record, a semantic routine, or a definition. The type field identifies the data type of the payload.

Figure 4.6 shows a specific example of PI-stack structure. The associated rule is <noun>=>"teacher". This rule is ambiguously defined. "Teacher" is the name of a class as well as of an attribute.

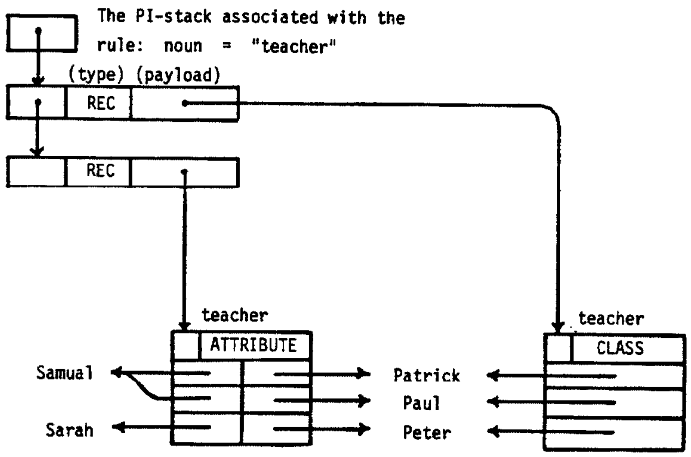

Figure 4.6 - An example showing the structure of the PI-stack.

In the majority of cases, ambiguities are resolved by context. The following sequence of queries to the database in Figure 4.6 show how the ambiguous term "teacher" can be used in query sentences to produce unambiguous answers.

```
Who are teachers?
  Patrick
  Paul
  Peter

Who are samuel's teachers?
  Patrick
  Paul
```

4.5 The Implementation of Basing

The basing operator provides the superior database A with knowledge of the information structure of its subordinate B. This knowledge allows A to access B′s data correctly. There are two ways in which we can implement the basing operator: (1) we can simply establish A′s right to use B′s dictionary, or (2) we can merge a copy of B′s dictionary into A′s.

The first method requires no duplication of any part of the subordinate′s dictionary, resulting in a trivial basing operation and minimum disk space requirements. However, we pay a heavy price in increased processing time whenever we query a superior database. Each query sentence must be parsed using not only the dictionary of the superior database but also the dictionary of each of its direct and indirect subordinates. Since each dictionary is a sorted tree, parsing with one large dictionary is many times simpler and faster than parsing with several small dictionaries.

The second method has the advantage of an efficient parsing algorithm using a single sorted dictionary. Since rapid response time is a primary concern in real-time database management system design, this method was chosen for the implementation of CDMS.

The basic basing operator algorithm is as follows: One at a time, modified copies of the dictionary entries of the subordinate database B are added non-redundantly into the dictionary of the superior database A. The rules are added without change, but the new PI-stack elements in A contain pointers to the corresponding PI-stack elements in B, rather than to B′s programs or data. A pointer to the dictionary PI-stack element of another database is called an "external dictionary reference".

Postponing explanations about why things are done this way to later sections, let us use an example to see what exactly is the structure of the new PI-stack elements that basing has added to the superior database: A database B contains the lexical item "teacher", ambiguously defined exactly as was shown in Figure 4.6. Another database A now bases itself on B. The basing operation adds the rule <noun>=>"teacher" to A′s dictionary. Figure 4.7 shows the structure of the corresponding PI-stack.

Notice that for each PI-stack element in B, the basing operator has created a corresponding new PI-stack element in A. The type fields of A′s PI-stack elements contain the code "xref" indicating that their payloads contain external dictionary references. The type fields of B′s PI-stack elements contain the code "rec" indicating that their payloads contain pointers to local data records.

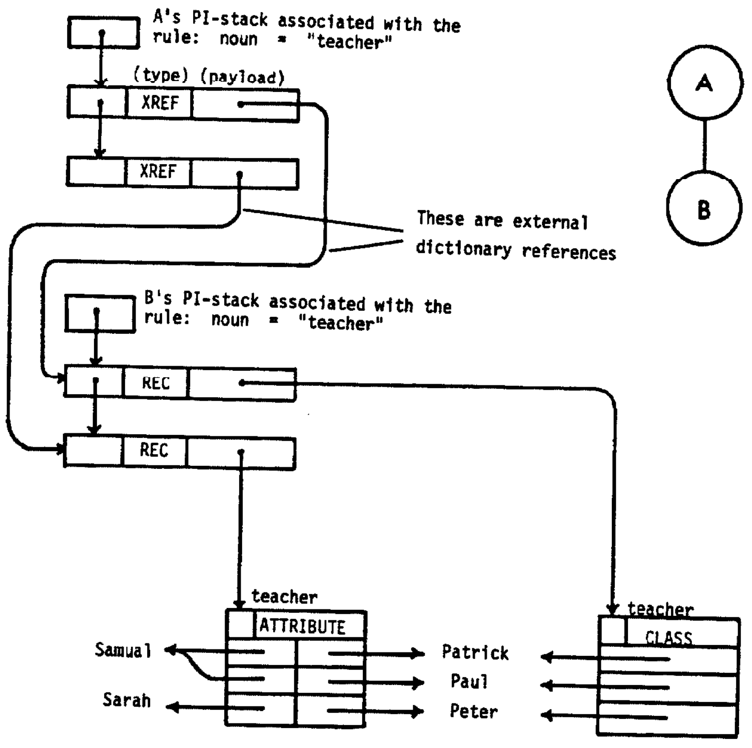
A's PI-stack associated with the rule: noun = "teacher"
(type) (payload)
XREF
XREF
These are external dictionary references
B's PI-stack associated with the rule: noun = "teacher"
REC
REC
A
B
teacher
ATTRIBUTE
Samual
Sarah
Patrick
Paul
Peter
teacher
CLASS

Figure 4.7 - The structure of a PI-stack created by basing.

4.6 The External Dictionary Reference Rule

The External Dictionary Reference Rule states that the dictionary of a database may contain pointers to its own data records or external dictionary references, but not direct pointers to the data records of another database. In the previous section we have seen how this rule is applied to the implementation of the basing operator. Thus basing links the dictionary superior database to the dictionary of the subordinate database.

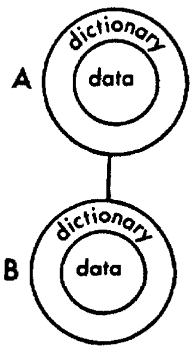

Figure 4.8 - Basing links the dictionaries of two databases.

The External Dictionary Reference Rule keeps the basing relationship well structured. A well structured basing relationship satisfies two conditions: (1) all changes in the subordinate's logical contents are reflected in the superior, and (2) changes in the physical representation of the subordinate's data records do not affect the superior, subject to condition (1). Figures 4.9.1, 4.9.2, and 4.9.3 demonstrate in three steps the necessity of the External Dictionary Reference Rule.

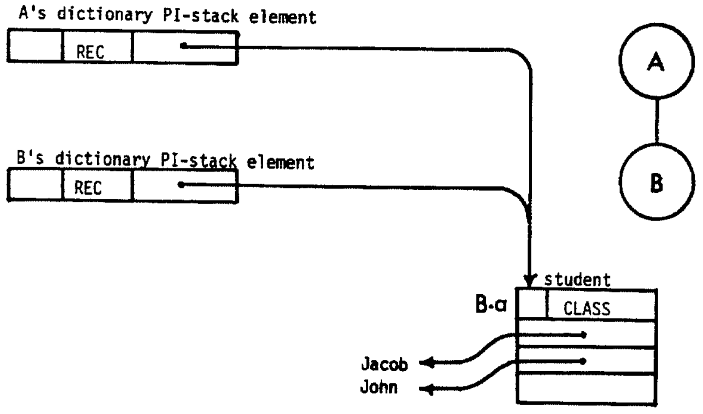
A's dictionary PI-stack element
REC
B's dictionary PI-stack element
REC
A
B
student
B·a
CLASS
Jacob
John

Figure 4.9.1 - This example violates the External Dictionary Reference Rule. The dictionary PI-stack element of the superior database A contains a direct pointer to the data record "student" belonging to the subordinate database B.

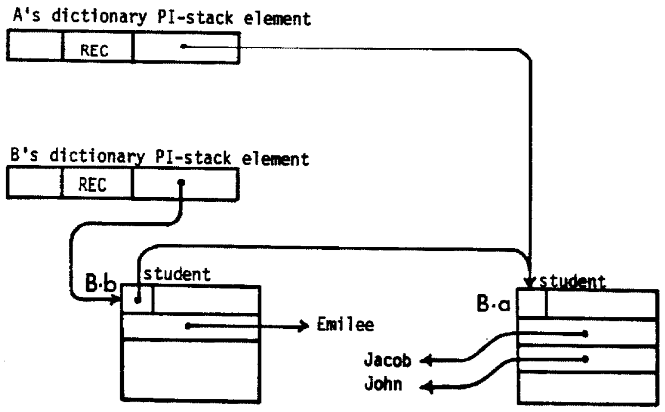
A's dictionary PI-stack element
REC
B's dictionary PI-stack element
REC
B·b
student
Emilee
student
B·a
Jacob
John

Figure 4.9.2 - A new individual is added to the class "student", but the addition is not reflected in the superior database A. Condition (1) of a well structured basing relationship is violated.

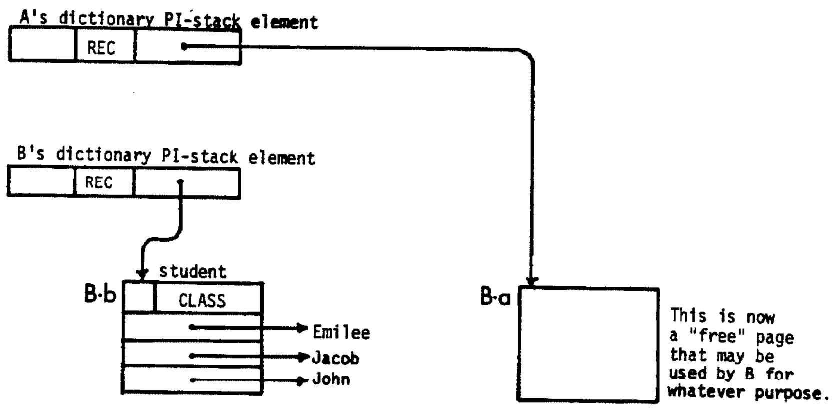
A's dictionary PI-stack element
REC
B's dictionary PI-stack element
REC
student
B-b
CLASS
Emilee
Jacob
John
B-a
This is now a "free" page that may be used by B for whatever purpose.

Figure 4.9.3 - The subordinate database B reorganizes the physical representation of the class "student" without altering its logical contents. This action jeopardizes the integrity of the superior database A, thus violating condition (2) of a well structured basing relationship.

Instead of exercising the External Dictionary Reference Rule, the problems indicated in the above example can be circumvented by imposing certain restrictions on the ways a subordinate database can manipulate its own data records. However, such restrictions would violate our general principal that a database should not be affected by the existence of a superior, with the single exception that structural deletions are ruled out.

4.7 Multiple Level Basing - Basing Path vs. Direct Line Algorithm

The previous sections described an algorithm to establish a basing relationship between two databases. Our next step is to extend that algorithm to handle multiple level basing. Figure 4.10 shows the simplest possible multiple level basing structure.

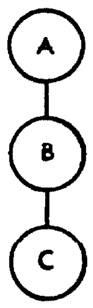

Figure 4.10 - A multiple level basing structure.

Let us introduce some terminology: When a database A obtains information from another database C which is a direct or an indirect subordinate of A, C is the "source database" of that information. A "basing path" is a sequence of basing linkages that lead from a superior database to a source database. We can uniquely identify a basing path by listing in order the databases along it. In figure 4.10, for example, A has a basing path (A,B,C) to C.

How can A access C's data? The first method that comes to mind is a straightforward extension of the single level basing algorithm: A's dictionary would contain pointers into B's dictionary, which would contain pointers into C's dictionary, which would contain pointers to its data. In other words, we follow the basing path that leads from A to C. This is called the "basing path algorithm."

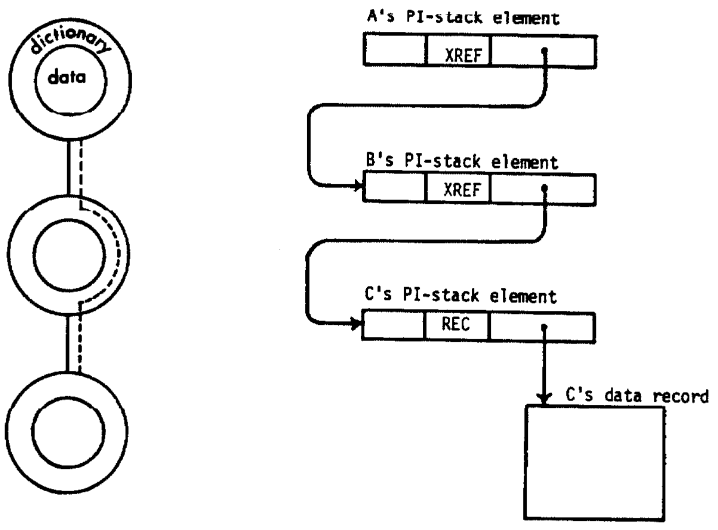

Figure 4.11 - The basing path algorithm.

The basing path algorithm is easy to implement and corresponds closely to the conceptual basing structure. Unfortunately, it becomes very slow when there are many basing paths to the same source database. Since the querying database only knows about its immediate subordinates, every path must be followed. To follow a single path, the page on which each

PI-stack element along that path is stored must be loaded into core. Figure 4.12 gives an idea of the number of page loadings required to access a data record in a source database several levels below the querying database. Since paging is a time consuming process, the basing path algorithm is unacceptably inefficient.

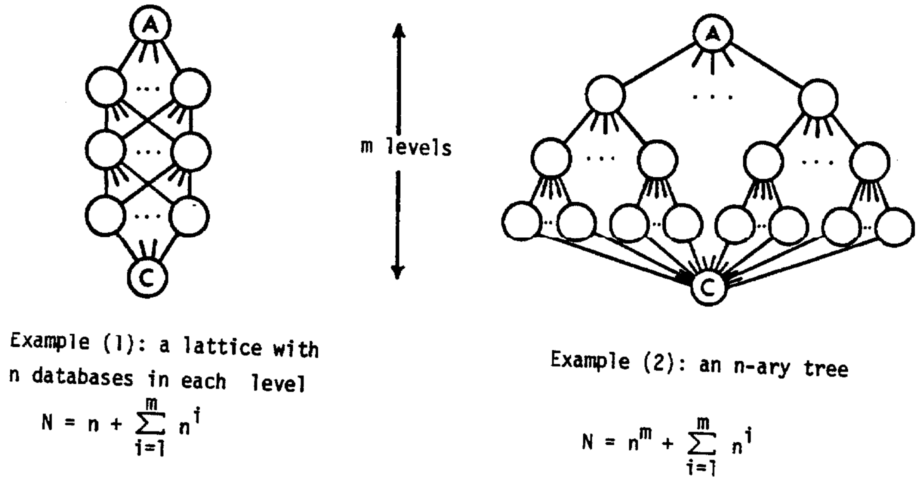
A
C
m levels
A
C
Example (1): a lattice with n databases in each level
$N = n + \sum_{i=1}^{m} n^i$
Example (2): an n-ary tree
$N = n^m + \sum_{i=1}^{m} n^i$

Figure 4.12 - Two types of basing structures. A is the querying database. C is the source database. We can think of C as the database that contains the grammar of the query language. N is the number of page loadings required each time A accesses C's data. In example (1), N is the worst case figure under an unoptimized algorithm.

An alternative way to implement multiple level basing is to have external dictionary references that point directly to the the dictionary of the source database, bypassing the intermediate databases along the basing path. In essence, we create direct lines of access to every indirect subordinate database. This is called the "direct line" algorithm.

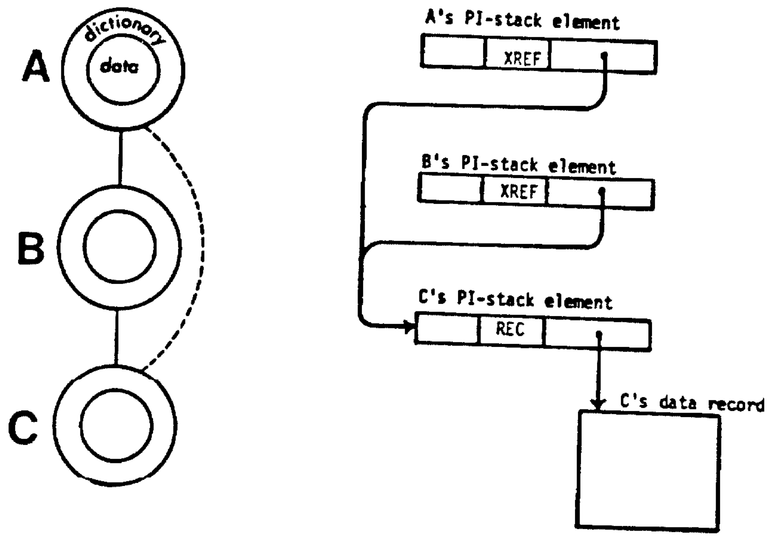

Figure 4.13 - The direct line algorithm.

The direct line algorithm optimizes paging. Accessing any external data record requires the same number of page loadings, regardless of whether it belongs to an immediate subordinate or an indirect subordinate. However, it introduces a technical problem that must be solved.

Let us use the basing structure shown in Figure 4.14 to describe the problem. A has one basing path to C, namely (A,B,C), and two distinct basing paths to E, namely (A,B,E) and (A,D,E). When we unbase A from B, (A,B,C) and (A,B,E) are destroyed. A can no longer access C, but can still access E through the basing path (A,D,E).

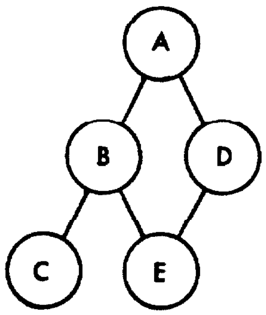

Figure 4.14 - Unbasing A from B prevents A from accessing C but does not prevent A from accessing E.

The basing path algorithm satisfies the situation we have just described trivially. A contains only external dictionary references to its immediate subordinates. To unbase A from B we simply go through A's dictionary removing all PI-stack elements whose payload contains an external dictionary reference to B. If we choose the direct access method, then the unbasing operator must be smart enough to remove all external dictionary references to C as well as to B, but to preserve external dictionary references to E. The next section describes how that can be done.

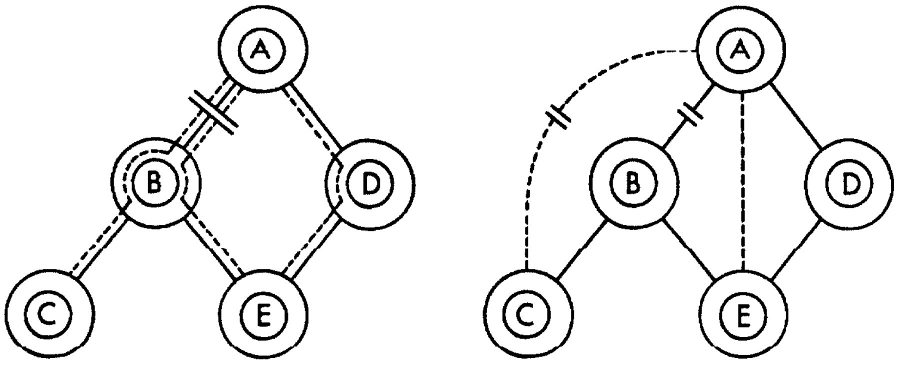

Figure 4.15 - What unbasing A from B involves under two different implementation methods. The left hand side diagram represents the basing path algorithm; the right side diagram represents the direct line algorithm.

4.8 Agents

To perform unbasing correctly the direct line algorithm must be modified. Since the superior database knows only about its immediate subordinates, that knowledge is what we have to work with. Let us introduce the notion of an "agent". If a basing path leads from a querying database A through one of its immediate subordinates B to a source database C, then B is an "agent" of the direct line of access from A to C. Notice that B and C may be the same database, and that a direct line of access to a source database may have more than one agent.

Figure 4.16 shows at an abstract level how the agent concept is used to modify the direct line algorithm. A direct line of access to a source database contains one or more alternate paths, each path is identified by the name of an agent. When A queries E, the program that handles external dictionary references, recognizing that the two alternate paths marked B and D are part of the same direct line of access to E, combines them and accesses E only once. To unbase A from B, alternate paths that are marked by B are eliminated from each direct line that leads away from A.

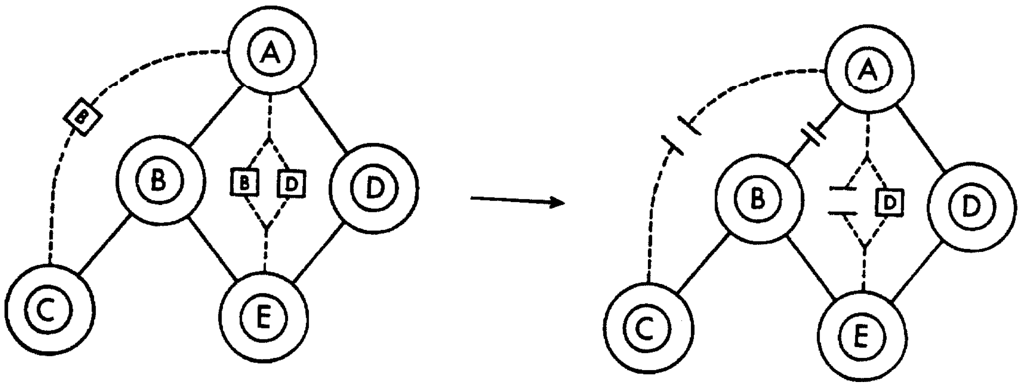

Figure 4.16 - Unbasing A from B under the modified direct line algorithm.

To incorporate the notion of agents into the direct line algorithm, the PI-stack element must be modified to include two new fields. The first is the "flag" field. It contains a boolean scalar. Turning the flag field off is equivalent to removing the PI-stack element from the PI-stack. The second is the "agent" field, it identifies the agent database whenever the payload field contains an external dictionary reference. Figure 4.17 shows the structure of the modified PI-stack element.

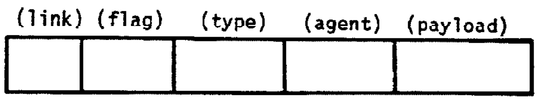

Figure 4.17 - The fields in a modified PI-stack element.

Figure 4.18 shows an example of the actual data structures used by the modified direct line algorithm. A's PI-stack corresponds to a direct line of access to E. Each PI-stack element corresponds to an alternate path, identified by the agent field. To unbase A from B, the flag field in A's PI-stack element whose agent is B is turned off, as shown in the diagram. This corresponds to the elimination of the alternate path marked by B. When A bases on B again, the flag field is switched on.

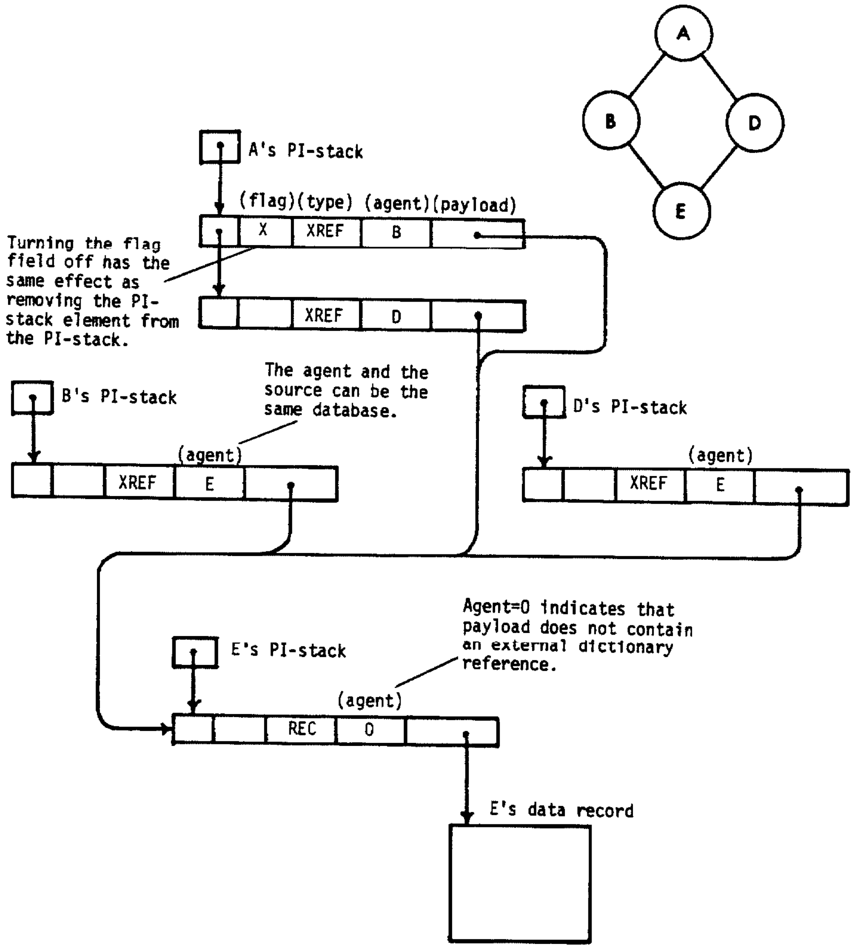
A
B
D
E
A's PI-stack
(flag)(type) (agent)(payload)
X
XREF
B
Turning the flag field off has the same effect as removing the PI-stack element from the PI-stack.
XREF
D
B's PI-stack
The agent and the source can be the same database.
(agent)
XREF
E
D's PI-stack
(agent)
XREF
E
Agent=0 indicates that payload does not contain an external dictionary reference.
E's PI-stack
(agent)
REC
0
E's data record

Figure 4.18 - The PI-stack structure under the modified direct line algorithm. The upper right hand corner figure represents the basing structure.

4.9 Extending the Data Records of a Subordinate Database

When a superior database extends the data records of a subordinate database, new data records that intertwine with the subordinate's data records are created in the superior. It becomes difficult then to sever the basing relationship cleanly without destroying some parts of the superior. Hence the extension of a subordinate database must be implemented with the unbasing operation in mind.

First let us list the desired properties of the unbasing operation. The superior database unbases from the subordinate database so that the latter can make structural deletions. A subsequent basing operation once again establishes linkage. Whatever extensions to the data records of the subordinate that the superior had made before the unbasing operation should be preserved so that the superior does not have to repeat any of its labor. Also, there should be no loose ends in the superior at any time. These requirements are stated more formally as follows:

(1) Provided that the subordinate is unchanged, the net effect of an unbasing operation followed by a basing operation should be zero.

(2) The unbasing operation preserves the integrity of the superior.

(3) Whatever changes the subordinate makes after the unbasing operation, the subsequent basing operation preserves the integrity of the superior.

With that in mind, let us outline an implementation. Consider first the following case: A database B contains a class called "ship". Another database A bases itself on B. A creates a new class called "vessel", adds a new member to "vessel", and makes "ship" a subclass of "vessel". A′s class "vessel" is now an extension of B′s class "ship". The following sequence of commands by A and B show the apparent effects of these actions, of an unbasing operation, and of a subsequent basing operation.

```
         (A)                        (B)

                              ENTER B
                              What are ships?
                                Enterprise
                                Kittyhawk
                              AUTHORIZE BASING BY A
BASE A ON B
ENTER A
vessel:=CLASS
Hornet:=NAME
Hornet is a vessel.
Ships are vessels.
What are vessels?
  Enterprise
  Hornet
  Kittyhawk
EXIT
UNBASE A FROM B
ENTER A
What are vessels?
  Hornet
EXIT
BASE A ON B
ENTER A
What are vessels?
  Enterprise
  Hornet
  Kittyhawk
```

Notice that even though "vessel" is an extension of "ship", "vessel" belongs to A. Unbasing from B takes away all of A's knowledge about "ship", but does not affect the rest of A's class "vessel". A second basing operation restores "ship" as a subclass of "vessel", requiring no further action by A.

To implement the above, we introduce the notion of an external subrecord. An external subrecord may be either an external subclass or an external subattribute. An external subclass is a subclass that belongs to another database. For example, B's class "ship" is an external subclass of A's class "vessel". External subattribute has a similar definition.

Figure 4.19 shows the data structure that represents an external subclass. Notice that the external subclass pointer in A's class "vessel" actually points to a PI-stack element in A. The unbasing operation turns off the flag field in A's PI-stack element that represents "ship". An external subclass pointer that points to a PI-stack element whose flag field is off is equivalent to a subclass pointer to an empty class. The next basing operation simply turns the flag field on again.

Figure 4.19 also shows why we turn off the flag field instead of simply eliminating the whole PI-stack element. Eliminating A's PI-stack element for "ship" when A is unbased from B would cause the external subclass pointer in "vessel" to point to an unknown quantity.

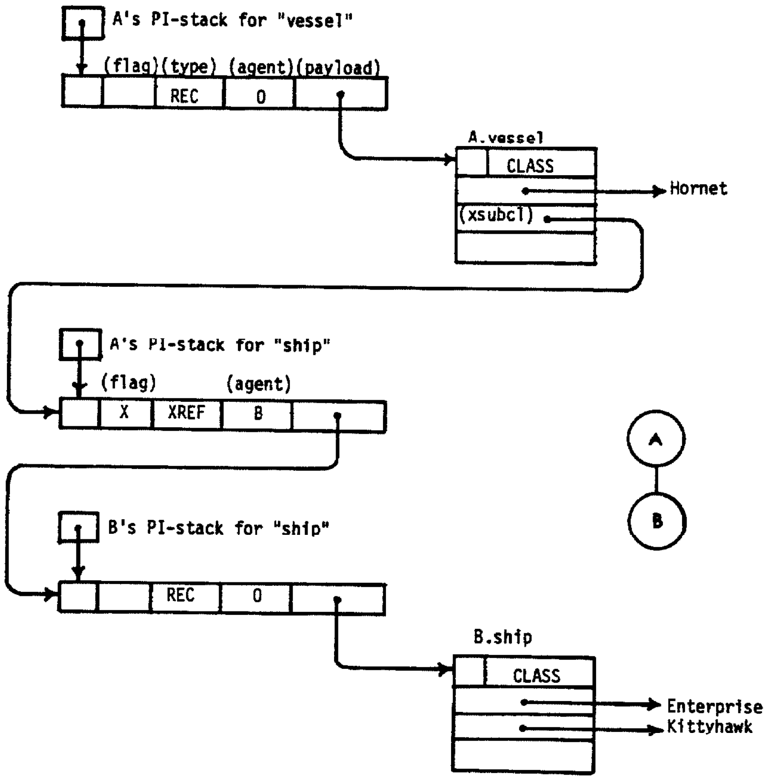

Figure 4.19 - The data structure that represents an external subclass.

The following example illustrates a second way in which a database may extend its subordinate's data records: Again A is based on B which contains the class "ship". This time A expands "ship"'s membership directly without creating a new class. In a sense A has adopted the term "ship" for its own use, giving it a broader meaning. A has created a new class A."ship" that includes B."ship" as a subclass. Following this line of argument, an unbasing operation strips A of all knowledge of B."ship", but preserves the rest of A."ship".

(A)

```
BASE A ON B
ENTER A
What are ships?
  Enterprise
  Kittyhawk
Nimitz:=NAME
Nimitz is a ship.
What are ships?
  Enterprise
  Kittyhawk
  Nimitz
EXIT
UNBASE A FROM B
ENTER A
What are ships?
  Nimitz
EXIT
BASE A ON B
ENTER A
What are ships?
  Enterprise
  Kittyhawk
  Nimitz
```

Figure 4.20 shows how the above example is implemented. Notice that the type field of the second element in A's PI-stack for "ship" contains the code "XSUB". It stands for external subrecord. A PI-stack element whose type field contains "XSUB" is not treated as a separate semantic interpretation of the rule associated with the PI-stack.

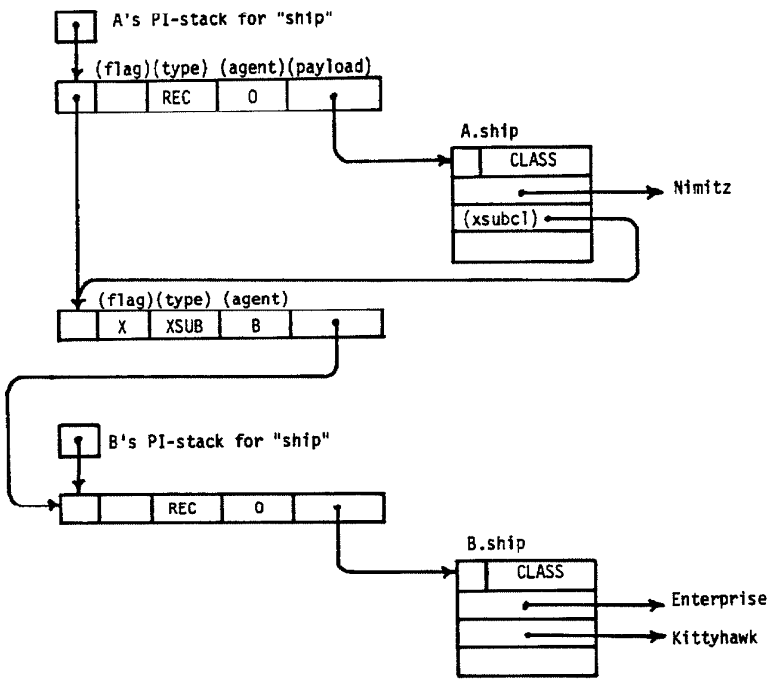

Figure 4.20 - **Direct extension of a subordinate's class** record.

4.10 The Implementation of Channeling

The implementation of the channeling operator is made easy by the REL/POL definitional capabilities. A database builds a regular definition for its own consumption, it builds a channel definition for the consumption of a specified recipient database.

Regular definitions are stored in the dictionary. Channel definitions are stored in the interpreter of the supplier database. The interpreter is simply a dictionary that (1) is separate from the regular dictionary, and (2) contains only channel definitions for other databases.

To establish a channel between a supplier database S and a recipient database R, two dictionaries are involved: the interpreter of S and the regular dictionary of R. It is a two step operation. First S makes the appropriate channel definitions for R, storing them in its interpreter. Next R merges a copy of those channel definitions in S's interpreter that are marked for R into its own regular dictionary. This merging operation is identical to the basing operation, with the single exception that the former selects only those interpreter entries that are marked for the recipient database, while the latter copies the entire dictionary of the subordinate database. In actual implementation they share the same programs.

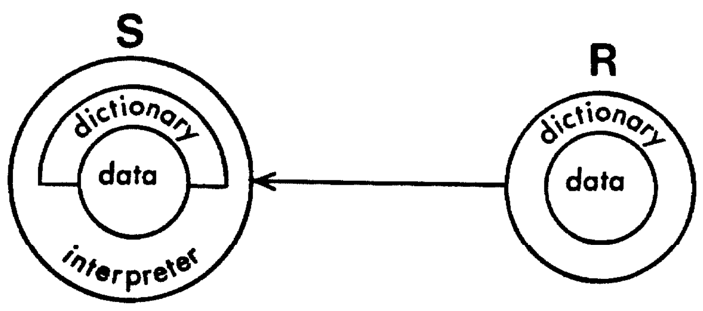

Figure 4.21 - Channeling links the recipient's regular dictionary to the supplier's interpreter.

To show the data structures used in the implementation, let us look at an example: A supplier database S defines the term "Norwegian ship" variously for two recipient databases Q and R. Q and R each then calls the channeling operator to establish channels to S.

```
      (S)

ENTER S
DEF FOR Q:Norwegian ship:ship whose flag is Norway
DEF FOR R:Norwegian ship:ship whose home port is/
some city in Norway
EXIT

      (Q)

ENTER Q
CHANNEL TO S
EXIT

      (R)

ENTER R
CHANNEL TO Q
EXIT
```

Figure 4.22 illustrates the data structures that result from the above sequence of commands.

The "detach" operator severs a channel. It recurses through the dictionary of the recipient database, turning off the flag fields of any PI-stack elements whose agent is the supplier. In Figure 4.22, the flag field in R′s PI-stack element is turned off, indicating that R has been detached from S. The "detach" and "unbase" operators are one and the same.

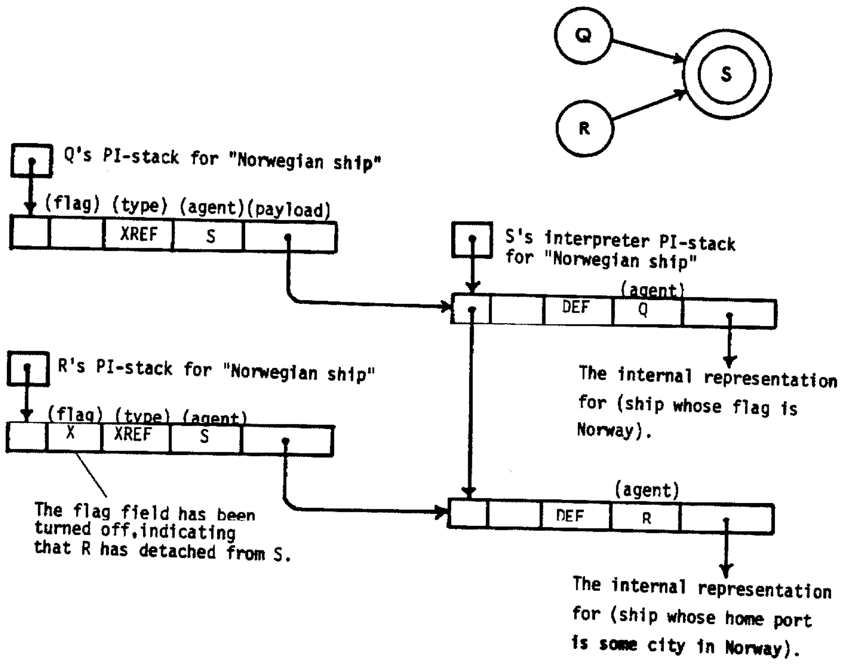
Q
S
R
Q's PI-stack for "Norwegian ship"
(flag) (type) (agent)(payload)
XREF
S
S's interpreter PI-stack for "Norwegian ship"
(agent)
DEF
Q
The internal representation for (ship whose flag is Norway).
R's PI-stack for "Norwegian ship"
(flag) (type) (agent)
X
XREF
S
The flag field has been turned off, indicating that R has detached from S.
(agent)
DEF
R
The internal representation for (ship whose home port is some city in Norway).

Figure 4.22 - Example of the data structures used to implement channeling. Notice that the "agent" field in the supplier's interpreter marks the intended recipient of the definition.

The similarities in the implementation of basing and channeling emphasize the affinity between these two linkage operators.

CHAPTER 5

DISTRIBUTED DATABASE SYSTEM MANAGEMENT

One important piece remains missing from the communicative database management system: What happens when a hierarchical organization operates from several distant locations.

Up to this point we have focused on single location organizations. A single location organization may be part of a larger organization, just as CPR is part of the Upjohn Company. The assumptions we have made about such organizations include the following:

(1) Its information system consists of a single contextual hierarchy like a monolithic mountain. Changes in the lower strata of the mountain send tremors all the way to the top.

(2) Everybody works in the same building. People can get together easily enough to set up new basing relationships or channels of lateral communication. In other words, minor adjustments in the database system are made continually to meet the changing demands of the organization.

(3) All databases are stored in the same computer system. A user can access data from another database about as efficiently as from his own.

These assumptions no longer hold true when the organization operates from several distant locations. This chapter discusses the additional managerial and technical problems faced by multi-location hierarchical organizations, and outlines a distributed database management system for them.

5.1 Database Systems Implemented on Local Computer Networks

Before going into multi-location organizations, let us take a quick detour to look at single location organization database management systems that are implemented on local computer networks. They are generally considered to be distributed database systems, but have very different properties from the distributed database systems for multi-location organizations.

A local computer network is a network connecting processors and peripherals that are physically close together. Let us say within one kilometer of one another. Examples of local computer networks are Ethernet [Metcalfe & Boggs 76], Fibernet [Rawson & Metcalfe 78], and IBM's System Network Architecture (SNA) [Gray & Blair 75].

The local computer network is an alternative architecture to the single processor for the implementation of database management systems for single location organizations. There is no data to indicate that either a larger processor or a local network of smaller computers has a clear cut advantage in cost/performance ratio. Much depends on the specific application. A network of small computers does have some potential advantages: it is more easily expanded, so a system can start with a relatively modest configuration, and security is easier to maintain when databases reside in separate hardware modules.

Operationally, it should make no difference whether a system is implemented on a single computer or a local computer network. The internal configuration of a well designed database management system is opaque to the user. Hence our only concern is whether we can implement a communicative database management system on a local computer network.

Technically there are two considerations. First, whether the network has sufficient bandwidth. Second, what changes need to be made in the dictionary structures to access data stored in a remote computer through the network.

The Ethernet has a bandwidth of about 3 Mb/s(million bits per second). There is a new implementation with a bandwidth of 10 Mb/s. The Fibernet, utilizing fiber-optic technology, has a bandwidth of about 50 Mb/s. Therefore, in the foreseeable future, a reasonable figure to use for an estimate of network bus bandwidth would be 100 Mb/s. This bandwidth must be divided by the number of slots or nodes on the network, let us say 10. Hence a ballpark figure for the apparent bandwidth of the network to the user would be about 10 Mb/s, sufficient for most applications.

The dictionary structure must be able to identify external nodes in the network as well as the databases concerned. This simply involves appending a device identification before each pointer to a data item.

5.2 Geographically Distant Distributed Database Systems

Here we are concerned with organizations that operate in several locations that are far apart. "Geographically distant" means that the locations are separated by more than a few kilometers - far enough so that people from different locations do not meet often, and far enough so that it would not be practical to stretch a local computer network over the whole organization.

The following terminology will be used for the rest of this chapter. The sub-organization at each location is a "division". A division may be a manufacturing plant, for example, or a sales office, or a military base. The computer system used at a division is a "node". A node may be a single computer, or it may be a local computer network.

Internally, a division operates like a single location organization. It has a dynamic database sub-system tailored to its own needs. At the same time, as part of a larger organization, the division must communicate with other divisions, and with organizational headquarters. Its database hierarchy is linked in some way to the database hierarchies of the other divisions, forming a larger distributed organizational database system.

The relationship between divisional database sub-systems differ greatly from the relationship between databases within a single division. There are operational as well as technical reasons for the difference. Let us look at the operational reasons first.

As before, let us begin from the broader perspective of the organizational information system.

5.3 Volume of Communication Within a Division and Between Divisions

The more closely a group of people work together, the more information they pass on to one another. Within a division, the "density"(roughly, frequency divided by the number of people) of instances of information exchange is greatest within small, closely-knit working groups. It is less between subgroups of larger groups, and still less between horizontal groups. Taken as a whole, the division is a community of people who work closely together and pass a tremendous amount of information to one another. They have to communicate efficiently.

A division attains this information efficiency by the compounding of contexts into a tightly structured, highly optimized context hierarchy which is adjusted continually to fit the changing needs of the division. Within the contextual hierarchy, people who communicate most frequently with one another share the largest amount of common context. Hence they also communicate most efficiently. By its very nature the contextual hierarchy is unique to the division.

The volume of communication between divisions is many orders of magnitude less than that within divisions. It has to be. To increase the amount of communication between divisions they have to share a larger common context. That would rob the divisional context hierarchies of much of the very individuality and flexibility that allow them to satisfy their internal information requirements.

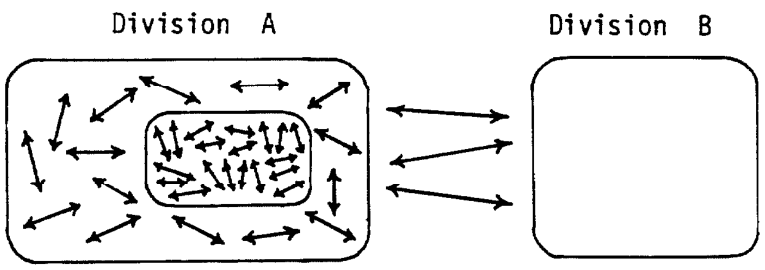

Figure 5.1 - Volume of communication within a division and between divisions.

The context shared by divisions is static and general in nature. Any changes in that context affects a larger number of people at different locations, presenting a difficult logistic problem. Frederick Thompson's essay titled "Military Information Systems" [Thompson 61] contains an excellent discourse on the contextual structure of military organizations. Much of this section is a rephrase of his ideas.

Figure 5.2 is a topographical representation of the contextual structure of an organization with several divisions.

The monolithic mountains represent the context hierarchies of the divisions. The blurred lines of their peaks show that they are constantly changing, especially at the higher levels. The flatland on which the mountains are based represent the general, rigid context underlying all divisions. It forms the basis for inter-division communication.

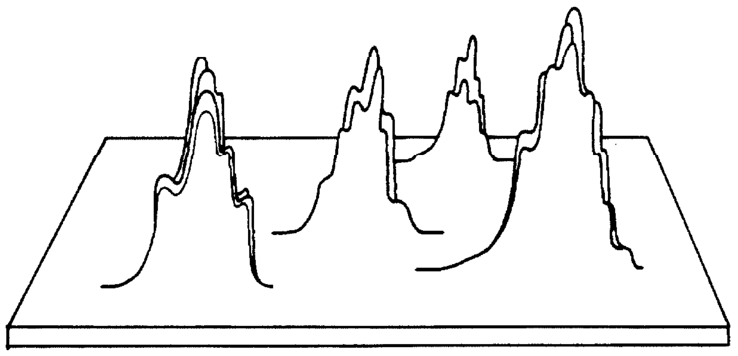

Figure 5.2 - The contextual structure of an organization with four divisions.

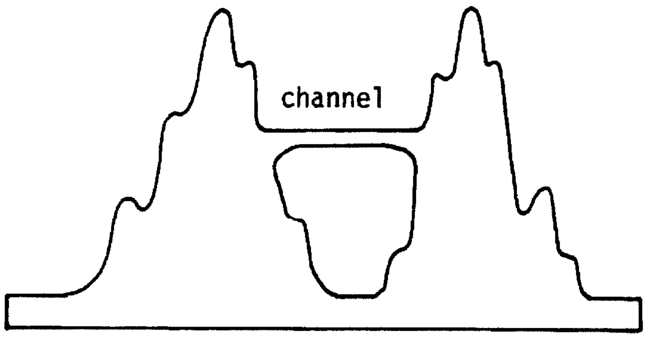

Figure 5.3 - A channel for lateral communication between divisional contextual hierarchies

Just as lateral communication occurs between horizontal groups within a division through rigid channels outside their shared contexts, channels of lateral communication can be set up between groups from different divisions. These inter-division channels are few, however, as it is difficult for people from distant locations to get together to establish them.

5.4 Technical Limitations to the Volume of Communication Between Divisions

Given today's technology, it is expensive to transmit large amounts of data across geographically significant distances in real time. Hence there is a technical limitation to inter-division communication independent of operational considerations.

Basically, there are two established ways to transmit data across long distances quickly: via satellite and through telephone lines. Satellite telecommunication has a transmission delay of about four seconds (One second to beam up to the satellite. Another second to beam down to earth. The same on the way back). It has a bandwidth wide enough for television - about 4 Mb/s. The trouble with satellite service is that it is expensive, and due to government discounts to small users, cost goes up more than linearly with the volume of data transmitted.

The medium of communication used by most existing computer networks such as ARPAnet, Cybernet, and Timenet is the leased telephone line. It has a bandwidth of about 50 Kb/s.

Let us do some quick and dirty calculations. A standard query to a medium sized database loads the order of 10,000 records, or about 2.5 million bits. If raw data were to be transmitted across the telephone line, the transmission delay per query given a dedicated line would be in the order of a minute. With a shared line which is part of a communication network, the delay would run into many minutes, which would be unacceptable for real time database management systems.

Until someone comes up with a revolutionary technical breakthrough, satellites and telephone lines will be the media for data transmission between distant nodes. In either case the volume of data transmitted through the communication network per query must be restricted. In the case of satellite telecommunication it is restricted by cost. In the case of telephone lines it is restricted by their bandwidth.

5.5 Some A Priori Decisions about the Properties of a Distributed Communicative Database Management System

I would like to propose a structure for a complete database management system for geographically dispersed organizations. Such a system is necessarily distributed. Before doing that, some decisions have to be made on its properties.

First, within the general context shared by divisions, communication should occur through homogeneous distributed databases whose structures remain stable. These databases may draw their information from the dynamic, individualistic divisional database hierarchies, but the proper interfaces must be provided so that they maintain uniform, consistent schemata. From experience, homogeneous distributed databases are the only kind that can be handled well by the state of the art.

Second, any communication that occurs regularly between divisions but outside their shared context must pass through channels of lateral communication established between the working groups involved.

Third, the database management system should be able to use existing network technology without modification. This in turn implies that it is bound by the bandwidth of the telephone line or the cost of satellite transmission. Therefore all processing has to be done within the nodes that contain the raw data. Only queries and processed results are allowed to pass through the network.

Finally, database security and integrity should be monitored at the nodes involved. The same mechanisms that provide security and integrity within a divisional database sub-system should be used for the distributed organizational database system.

5.6 Resolving the Conflicting Database Requirements of the Division and of the Organization

The issue of centralized versus distributed management control is an ever present concern in database management system design. We have seen that the database sub-system of a division must be tailored to meet its own requirements. On the other hand, unchecked freedom often results in incompatible sub-systems that make inter-divisional information exchange impossible. Hence the divisional database sub-system must maintain its internal efficiencies while conforming to certain external rules and requirements.

How can a database management system satisfy these seemingly incongruous conditions? The answer lies in using a translation process similar to the action of the interpreter in a channel of lateral communication. Databases called "agents" are set up to give the divisional database sub-system the desired appearance to the organization outside of the division. Why they are called agents will soon become apparent. An agent has various channels to local databases that are the ultimate sources of the information provided to the organization without. These

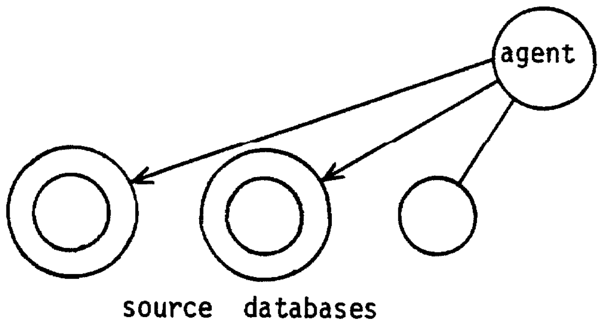

Figure 5.4 - The schema of the agent satisfies the requirements of the organization.

source databases, structured to meet the internal needs of the division, do not necessarily conform to the general requirements of the organization. The only constraint imposed upon them is that collectively they contain the necessary data that can provide the required information to the rest of the organization through proper translation by the channel interpreters. When the structure of an internal database coincides with the requirements of the organization, the agent is based directly on it.

In general, there are as many agents in a divisional database sub-system as there are distributed databases used by the organization that contain information from this particular division. The source databases may change in structure, but as long as the channel interpreters are adjusted accordingly, the schema of the agent can be maintained.

5.7 How to Query a Distributed Database – Procedure and Implementation

To set up the necessary querying mechanism to a distributed database d, the following procedure is followed:

(1) A "window" database is created in the node from which the queries are to be made. The window is "remotely based" on each of the agents of the distributed database, d. This remote basing process creates PI-stack elements in the dictionary of the window whose "agent" fields contain the identities of the external agent databases. The contents of the "payload" fields are of no consequence. The actual remote basing can be done by taking a copy of the dictionary of each agent to the window node, and basing the window on the local dummy copies of each agent in turn. The exact procedure is of no importance.

(2) Each agent explicitly authorizes the remote basing by the window. After the authorization any messages from the window would be processed by the agent.

(3) User databases are based on the window.

(4) The user can then type a query. For example, "What is the year-to-date sales of rubber sandals?" The query is parsed, and by examining each complete parsing graph, a list of all the relevant nodes, that is, nodes at which the query would also parse, can be produced.

The user can also specify the nodes to which his query is to be directed. For example, "What is the year-to-date sales of rubber sandals in London and in Kuala Lumpur?"

(5) A transaction containing the query is sent to each relevant node via the communication network. The transaction explicitly identifies the user database, the window database, and the agent database.

(6) At each destination node, the local database management system translates the transaction into the following sequence of operations:

```
ENTER agent
process query
EXIT
send result to user-database
```

If window had not been granted the authority to base on agent, the "ENTER agent" operation would be aborted. In that case "result" would be an error message.

(7) The user database, upon receiving the returns from all the transactions, combines and displays the result. For Example:

```
YEAR-TO-DATE SALES OF RUBBER SANDALS:
London : $500.00
Kuala Lumpur : $236,200.00
```

5.8 A Distributed Database Example

Figure 5.5 shows the structure of a distributed database, d. Irregular boxes represent nodes. d is the sales and marketing database of a world-wide shoe distributor. Node B, located in India, contains a divisional database sub-system that stores sales volumes in terms of Rupees. Node C in Malaysia uses Ringgits. Node D in England uses Pounds and Shillings. Through channel interpreters, each of the agents represent sales volume information in terms of U.S. Dollars. From Node A, the headquarters of the company in New York, marketing planners can query d as a homogeneous distributed database system that gives all its answers in dollars. The interpreters at each division are adjusted daily to reflect the going currency exchange rates.

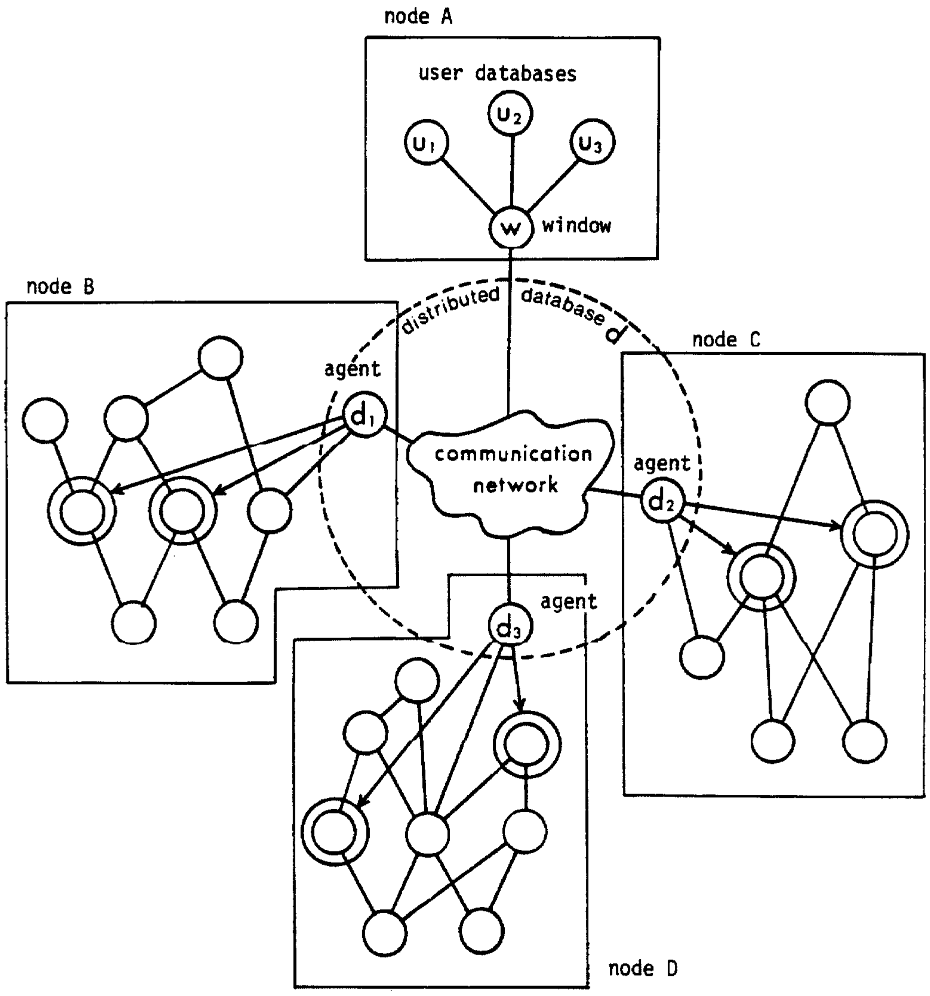

Figure 5.5 - Querying a distributed database stored in several geographically distant locations.

5.9 Distributed Database System Security and Integrity

The security of the distributed database is maintained within the window and agent nodes. The communication network plays no part in the maintenance of security. The window database and the agent database function as key and lock. Users whose databases are based on the window have access to the only key within the node. Hence the basing mechanism restricts access to the distributed database to authorized users.

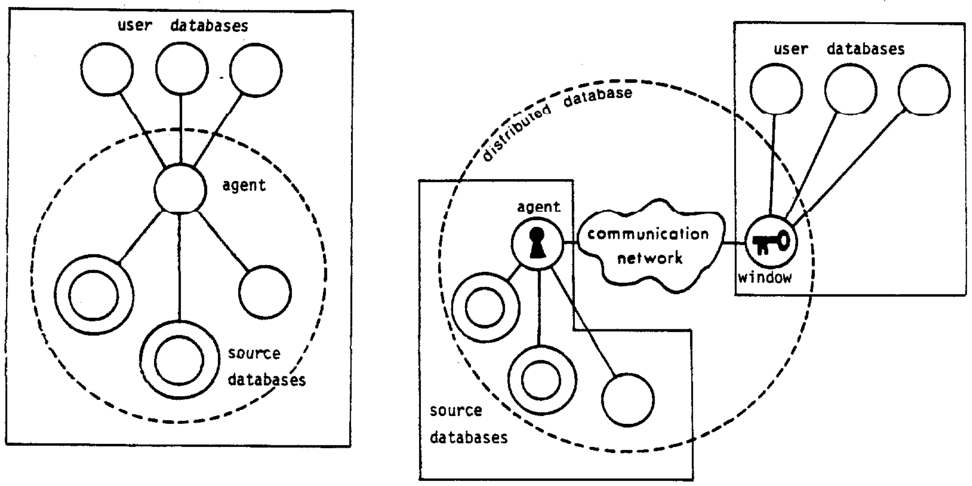

Figure 5.6 - How database security is maintained within a division and between divisions.

Within the agent node, the agent preserves the integrity of the source databases from which it draws its information. Since the queries are processed within the agent and not within the source databases, they have no effect on the contents of the latter.

5.10 Channeling Across Divisions

Any two working groups from different divisions constitute a pair of horizontal groups. Their shared context is even smaller and more rigid than the shared context of horizontal groups within a division. Any regular communication outside this shared context must pass through a channel of lateral communication. The same window-agent mechanism can be used to set up a channel between two nodes.

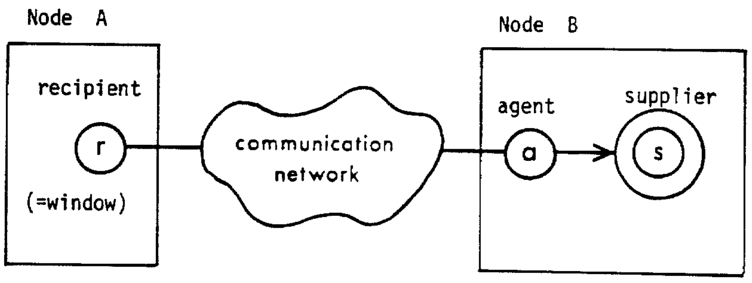

Figure 5.7 - Establishing a channel of lateral communication into a geographically distant node.

5.11 Updating a Distributed Database from Multiple Nodes

Sometimes a database is updated as well as queried from many different locations. An example is the airline reservation system. The flight and passenger reservation information is stored in one or more nodes. From any of a number of airport terminals or ticket offices airline personnel can reserve a seat on a given flight, thereby altering the contents of the passenger reservation database stored in a remote computer.

We can construct such a database by placing the information that can be updated from remote locations directly into the agent databases. Figure 5.8 illustrates the airline reservation example. Notice that flight information cannot be altered from the remote nodes.

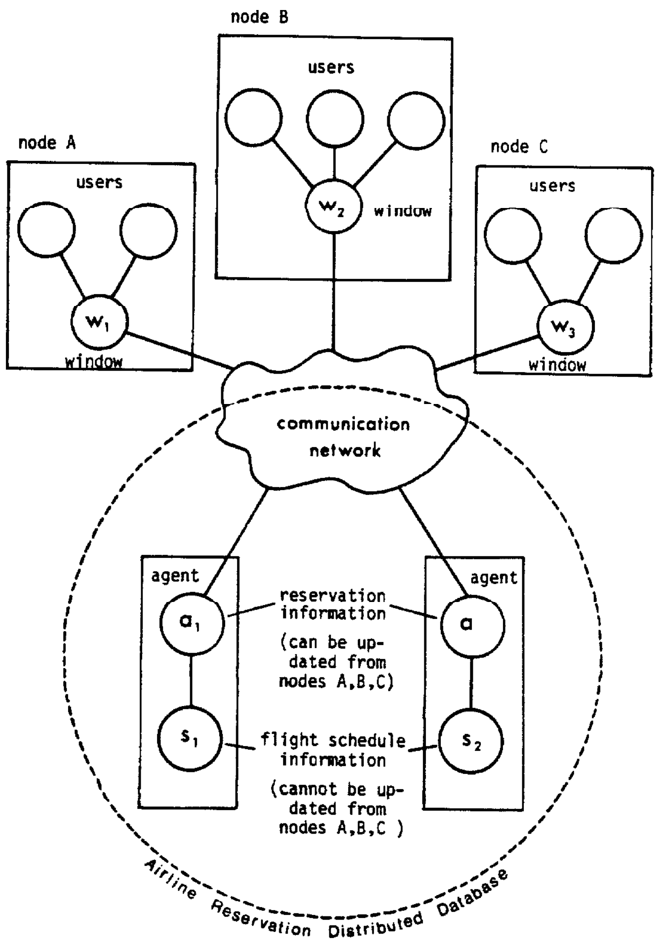
node B
users
w_2
window
node A
users
w_1
window
node C
users
w_3
window
communication network
agent
a_1
s_1
agent
a
s_2
reservation information
(can be updated from nodes A,B,C)
flight schedule information
(cannot be updated from nodes A,B,C)
Airline Reservation Distributed Database

Figure 5.8 - Updating a shared database d from several geographically distant nodes.

A distributed database may be queried from a node which contains part of its information. For symmetry the same mechanisms must be used to access the part of the database stored locally, as shown in Figure 5.9.

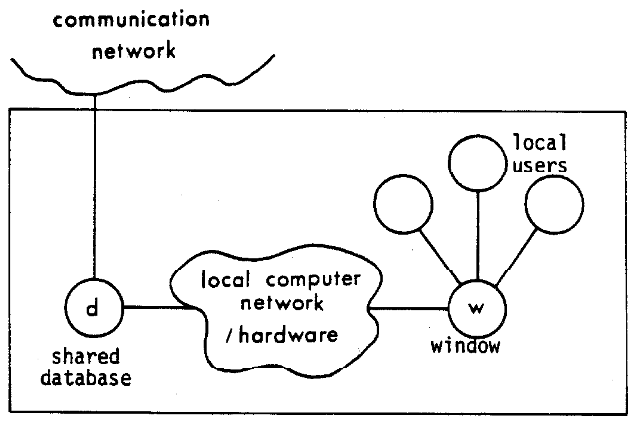

Figure 5.9 - How a shared database that can be updated from several distant nodes is updated from within its own node.

Figure 5.10 shows a complete example of a homogeneous distributed database that is stored in and updated from multiple geographically distant nodes. A specific instance of such a database is the hotel chain reservation system: Nodes A, C, and E are located in three different hotels; nodes B and D are located in airports.

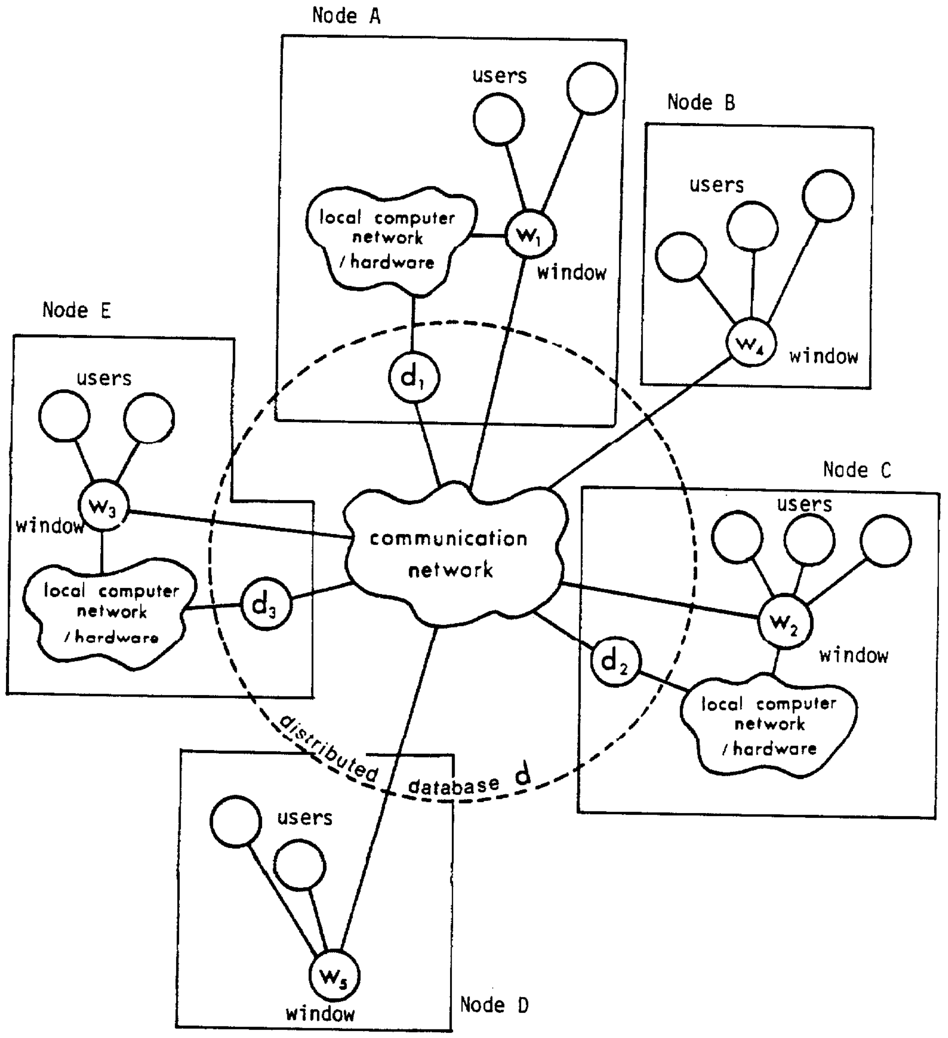

Figure 5.10 - A homogeneous distributed database stored in and updated from multiple nodes.

5.12 Updating Redundant Information

In some applications, such as those involving national defense, it is necessary to maintain copies of the same database in multiple, distributed nodes. This information redundancy allows the system to respond correctly and rapidly even when some of its nodes are disabled. To ensure that the contents of redundant distributed copies of a database remain identical, they must be updated simultaneously by a single command.

The updating command is processed at the issuing node, producing the address of each redundant database, where the text of the command is sent. At each destination, the command is parsed unambiguously. The processed results are then sent back to the issuing node, which eliminates redundant answers by comparison.

5.13 Communication Across Multiple Networks

Section 3.5 described the transitive nature of the basing relationship. When a superior database A is based on a subordinate database B, A gains access to all of the information in the database subsystem subordinate to B. As far as A is concerned, it is based on a single database. This fact has been highlighted in the diagrams by drawing a dotted line circle around the subordinate database subsystem.

The transitivity of the linkage operators can be generalized to allow communication across arbitrarily many computer networks, as Figure 5.11 illustrates. Each intervening node contains an agent-window database pair, serving as the "gate" between two computer networks.

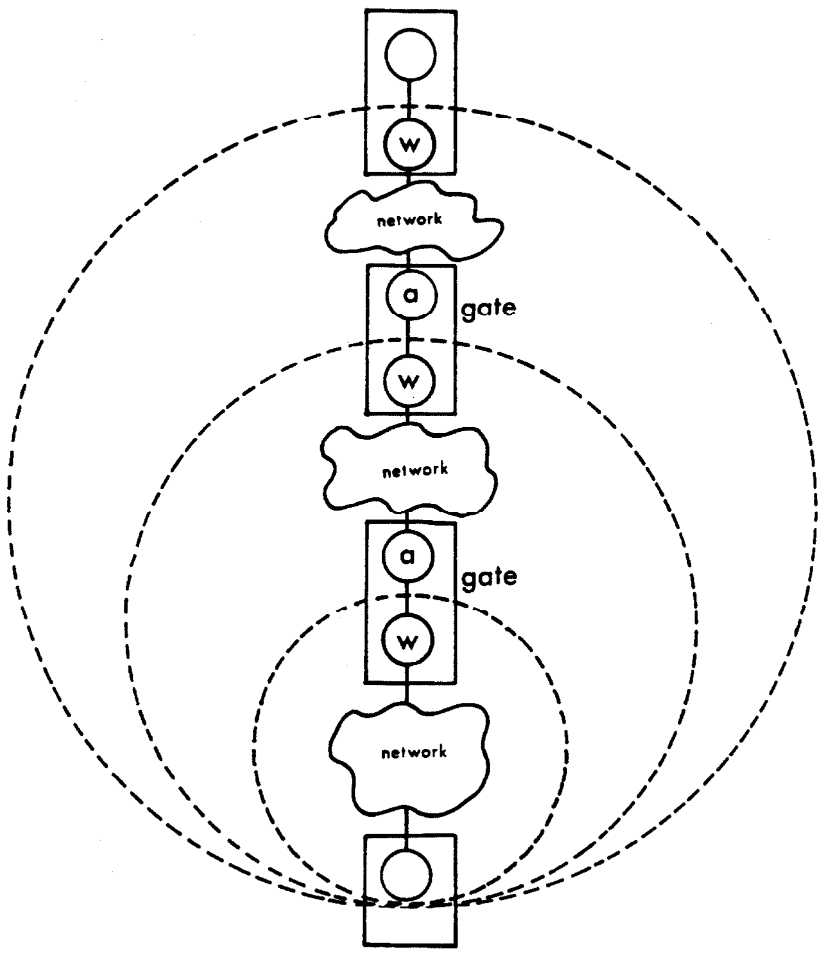

Figure 5.11 - Basing across several networks.

5.14 Summary

Let us stand back now and take a look at the abstract structure of the organizational database system.

The single location organization, of which a division of a multi-location organization is a special case, has a heterogeneous database system forming a single basing hierarchy. Whether it is a distributed database in the sense that it is stored in several computers in a local network is technical. Such a heterogeneous database hierarchy may be represented by a pyramid.

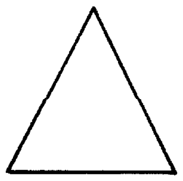

Figure 5.12 - A heterogeneous database hierarchy - the configuration of a single location organization database system.

Homogeneous distributed databases span the divisions of a multi-location organization. Within each database, the data stored at different nodes may differ in content only or in structure as well. What is important is that the homogeneous distributed database has a consistent and stable schema.

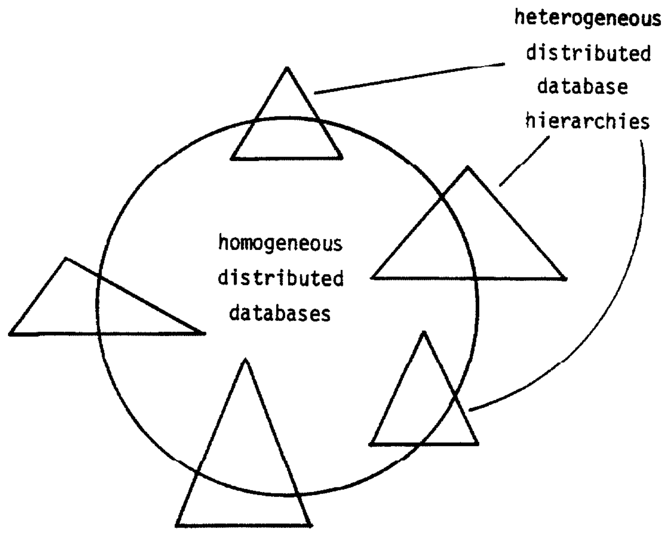
heterogeneous distributed database hierarchies
homogeneous distributed databases

Figure 5.13 - **The configuration of a multi-location** organization **database system.**

Superimposed on the entire database structure are channels of lateral communication set up between horizontal groups.

CHAPTER 6
CONCLUDING REMARKS

The orders-of-magnitude increase in computing power in the past decade has raised the public's expectations in all classes of computer services. In many instances, these expectations have not been justified.

Database Management Systems belong to that class of services that have fallen short of expectations. The need for DBMS is apparent. More are installed each day despite their obvious shortcomings. Yet very few users are truly satisfied with their own systems.

Responding to this need, database management has become a popular research topic. Due to the large number of unsolved problems, most researchers narrow their attention to a particular facet of the problem area such as database security, or distributed database systems.

This thesis described a different direction of research. It identified a particular class of intended users - the members of a hierarchical organization - and treated their database management problem from a global perspective.

Two aspects of our approach to the problem are worth mentioning again:

First, the communicative database management system was designed to reflect the natural information structure of the organization, not to replace it. Only after that condition is satisfied do we attempt to improve upon its performance.

Second, to gain an understanding of the information structure of the organization, a model was created. The importance of the model lies not in its truth, but rather in that it provided us with a conceptual framework within which the problem area can be understood.

BIBLIOGRAPHY

Ackoff R.L., "Management Misinformation Systems", Management Science, Vol 14, Num 4, December 1967

Aron J.D., "Information Systems in Perspective", Computing Surveys, Vol 1, Num 4, December 1969

Canning R.C., "The Cautious Path to a Data Base", EDP Analyzer, Vol 2, Num 6, June 1973

Chervany N.L., and Dickson G.W.,"An Experimental Evaluation of Information Overload in a Production Environment", Management Sciences, Vol 20, Num 10, June 1974

Denning D.E., Denning P.J., and Schwartz M.D., "The Tracker: A Threat to Statistical Data Base Security", ACM Transactions on Database Systems, Vol 4, Num 1, March 1979

Gray J.P. and Blair C.R., "IBM's Systems Network Architecture", Datamation, April 1975

Hess G.D., "A Software Design System", Ph.D. Thesis, Caltech, March 1980

Hirsch R.E., "The Value of Information", The Journal of Accountancy, Vol 125, Num 6, June 1968

Housel B.C., "Pipelining: A Technique for Implementing Data Restructures", ACM Transactions on Database Systems, Vol 4, Num 4, December 1979

Lorin H., "Distributed Processing: An Assessment", IBM Systems Journal, Vol 18, Num 4, 1979

Luke J.W., "Data Base Systems: Putting Management Back into the Picture", CSC Report, Vol 9, Num 1, May 1975

Martin J., "Principles of Database Management", Prentice-Hall, Inc., 1976

Metcalfe R.M. and Boggs D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks", CACM, Vol 19, Num 7, July 1976

Montgomery D.B. and Urban G.L., "Marketing Decision Information Systems: An Emerging View", Journal of Marketing Research, Vol 7, Num 2, May 1970

Raver N. and Hubbard G.H., "Automated Logical Database Design: Concepts and Applications", IBM Systems Journal, Vol 16, Num 3, 1977

Rawson E. and Metcalfe R.M., "Fibernet: Multinode Optical Fibers for Local Computer Networks", IEEE Transactions on Comm., Vol 26, Num 7, July 1978

Ries P.R. and Stonebraker M.R., "Locking Granularity Revisited", ACM Transactions on Database Systems, Vol 4, Num 2, June 1979

Thompson B.H., "REL User's Reference Guide", REL Project Report, Caltech, 1977

Thompson B.H. and Thompson F.B., "Practical Natural Language Processing: The REL System as Prototype", REL Project Report, Caltech, April 1975

Thompson F.B., "Design Fundamentals of Military Information Systems"

Yu K. and Chess R., "Report to CPR", Caltech, 1978

Ziegler K., "A Distributed Information System Study", IBM Systems Journal, Vol 18, Num 3, 1979