1-1-1982

Toward the design of a distributed data base system.

Jan Gary Kroc

TOWARD THE DESIGN OF A DISTRIBUTED DATA BASE SYSTEM

by
Jan Gary Kroc

A Thesis
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Master of Science
in
Industrial Engineering

Lehigh University
1982

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

MAY 10, 1982
(date)

Professor in Charge

Chairman of Department

TABLE OF CONTENTS

List of Figures

ABSTRACT

Toward the Design of a
Distributed Data Base System

Jan Gary Kroc
Supervisor: Dr. John Krobock

The investigation of alternative distributed data base system (DDBS) structures is reported. Based on this investigation, preliminary considerations for information system and user management to design a DDBS are defined. The considerations defined are general in nature, however, no claim is made to their applicability to meet all industrial requirements.

Two classes of Distributed Data Bases; homogeneous DDBS's and heterogeneous DDBS's are defined. The homogeneous DDBS was selected as the class which could have a design methodology specified without becoming involved in specific vendors and industrial hardware, software, and business structure.

Three structural types of Distributed Data Bases are described. These include: Partitioned Data

Bases; Replicated Data Bases; and Hybrid Data Bases. Because of the wide variety of usage patterns that can be associated with the three structures defined, two methods of partitioning and replication are identified; i.e. Horizontal partitioning and/or replication and Vertical partitioning and/or replication. Possible implementation circumstances are described.

The design methods used for centralized Data Base Management Systems were determined to be partially applicable to the Distributed Data Base System case. Realignment and addition to the ANSI/SPARC three schema approach are reported.

A qualitative and quantitative approach to begin addressing the questions of how to design distributed Data Base applications is reported. High level examples of both the qualitative and quantitative approaches are provided.

1.0 Introduction

In the past several years,there has been a significantly strong movement in commercial computing toward the distribution and/or decentralization of data processing.[Cypser 78; Enslow 78; Scherr 78]. At this time, there is virtually no large corporation in which the data processing capacity is focused in a single computer system. [Scherr 78]. Additionally,there are few corporations that operate in only one geographic location.

There are several underlying developments that have aided the move to distributed processing. This move is one of several possible today in large part because of the reduction of the costs of computer hardware. [Cypser 78; Martin 76]. In the past, economies of scale have been sufficient to justify one large data processing center to perform most services.

Technical developments have also had a major influence on the practicality of distributed data processing. Major technological changes in the areas of networking and communications have made it possible to have many new capabilities.[Cypser 78; Martin 76].

Given the fact that distributed data processing is an area of burgeoning interest and development, it is interesting to note that there is another development in the area of the distributed concept. This development is the marriage of the concepts of distributed data processing with the concepts of data base management systems.[Edelberg 78, Rothnie 80, Severino 77].

Distributed data base management systems are the converging points of apparently contrasting areas; networking and data bases which are representative of distribution and integration respectively. The concept of data base management systems evolved to meet the needs of integration in large centralized systems. Distributed processing has evolved from the need to have processing power and data available to remote locations. Benefits associated with each are being sought in today's systems and will be required in the systems being planned for the future.

Interest in the development of distributed data base systems is being stimulated by two major trends in commercial computing. These are:

1) An ever increasing dependence on readily accessible and reliable information . [Buchanan 80]

2) An increasing use of data communications that make computer applications more convenient and feasible geographically. Therefore, they are more effective and efficient to the organizations they serve. [Cypser 78]

The major reasons for these trends are undoubtedly economic in nature. Contention for hardware are software development are also key reasons for this trend. Availability of accurate and up-to-date information for all levels of an organization has in many cases assumed critical importance. Moreover,the information resource represents a tremendous financial investment in terms of personnel, hardware, software, etc. Because of this situation, considerable interest has developed in the discipline of data base management systems, particularly in an on-line environment. Data base management systems have become a way of enhancing both the availability and consistency of data within an organization.

Along with the trend toward utilizing the benefits provided by a DBMS, the economies of scale associated with having a centrally located computer connected to remote users via communications lines have slowly disappeared due to decreases in hardware costs. Simultaneously, communications costs have decreased, but, the drop in communications costs are occurring at a far slower rate as compared to the decline in hardware costs. This relationship is illustrated in Figure 1.1. [Cypser 78]

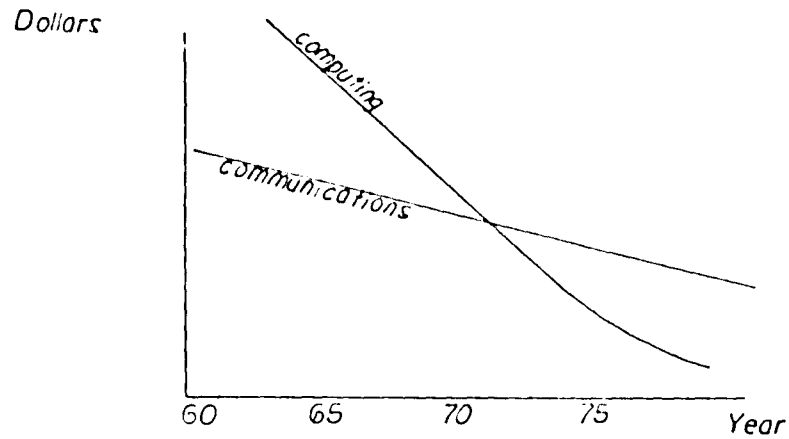


Figure 1.1

Relative Cost Decline

[Cypser 78, pg. 37]

The dispersion of computing power has been the logical consequence of this non-symmetric relationship between computing costs and communications costs.

1.1 Relevance

Against the above described background, the potential benefits of a system in which parts of the data base are distributed to the sites which most frequently use them have become both more apparent and more attainable. These possible benefits and some of the possible problems associated with distributed data base systems will be described in the following sections.

1.1.1 Why distribute the data base?

Some of the benefits associated with a distributed data base might include combinations of or all of the following : [Edelberg 78, Rothnie 80, Severino 77]

1) Reduction of total data processing costs due to a reduction in communications needed.

2) Faster acccess to locally held data via the elimination of the need to transmit access requests to a central site.
3) Faster response to local needs or priorities in accessing the data base(s).
4) Improved data security due to local control.
5) Improved reliability in the overall system due to distribution. This is true in the sense that no failure of a single processor can stop the entire system. Note: This is not true if there is a master/slave relationship.
6) Improved control of the data processing function.
7) Improved ability to extend the data base as required without radical restructuring of an existing system because of the increased modularity of the distributed systems.

1.1.2 Distributed processing vs. Distributed data bases

The term distributed data processing has become one of the most frequently used terms in the field of information systems. It is also one of the most misunderstood concepts as well. At least four physical components of a system can be distributed:[Enslow 78]

1) Hardware or processing logic

2) Data

3) The processing itself

4) Control; such as the operating system

It is not merely the physical distribution of some of the components of a system that makes a distributed system, but also the fact that the components of the system interact. Graphically this general situation can be depicted as follows in Figure 1.2.

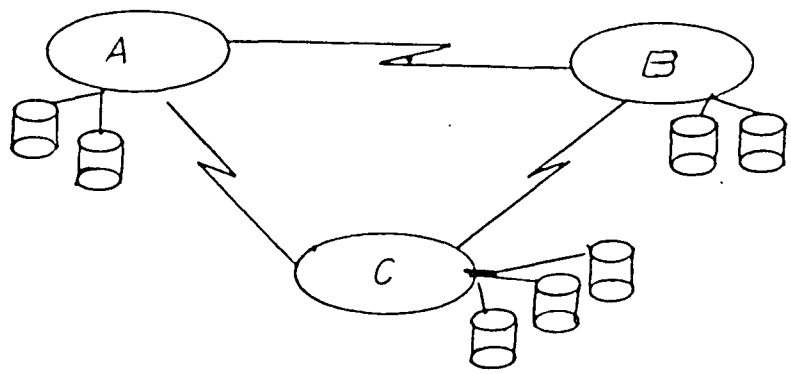


Figure 1.2

Distributed System Interaction

In general, the situation is one in which a user can access data in a remote computer system. Figure 1.3 is an example.

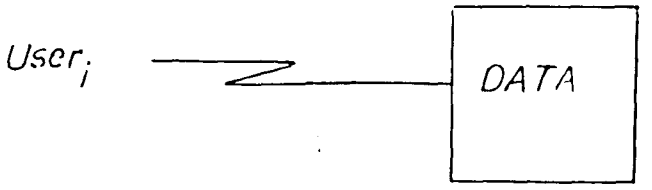


Figure 1.3

User Data Access

The term distributed data base is sometimes used to refer to the distributed data processing situation, however, its use generally implies something more. A distributed data base can have the same topological setup as depicted in Figure 1.2. However, the distinction lies in the fact that in a distributed data base environment there are data relationships between the dispersed elements which are important to the users. Moreover, the desire is to structure the system as if the data resided on a single machine; ie. hide the distribution from the users and the application programmers. Graphically, the situation of the dispersed data relationships can be

summarized as shown in Figure 1.4.

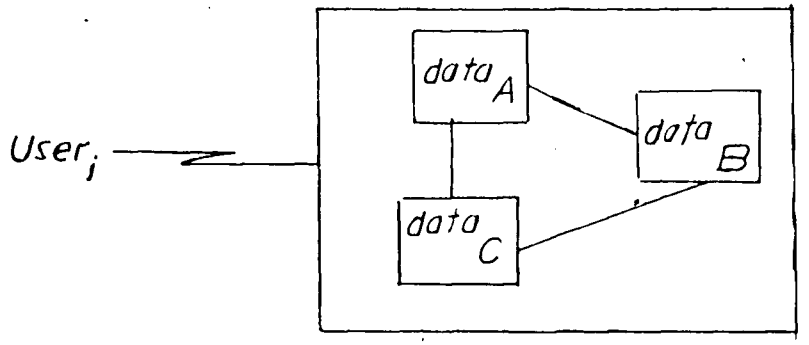


Figure 1.4

Dispersed Data Relationships

1.1.3 The distributed data base environment

The relationship that exists between the dispersed data elements is the key distinction that separates a simple distributed system (DDPS) from that of the more complex distributed data base system (DDBS). The former (DDPS) can and should interact with other nodes in the system, however, each portion of the system is separated into specialized functional units. In a distributed data base system, there is a partitioning of the data base which can vary according to the desired degree of distribution . In order of increasing distribution, this may be: [Enslow 78]

1) A partitioned data base that consists of the data most frequently utilized by a particular node is kept at that node with copies as required at other nodes. In addition, a complete copy of all files is kept at a master node. Under this arrangement, any transactions causing an execption condition at a local node is transferred and processed by the master node. This arrangement is frequently refered to as a Vertical distribution model.[Scherr 78] This is illustrated in Figure 1.5.

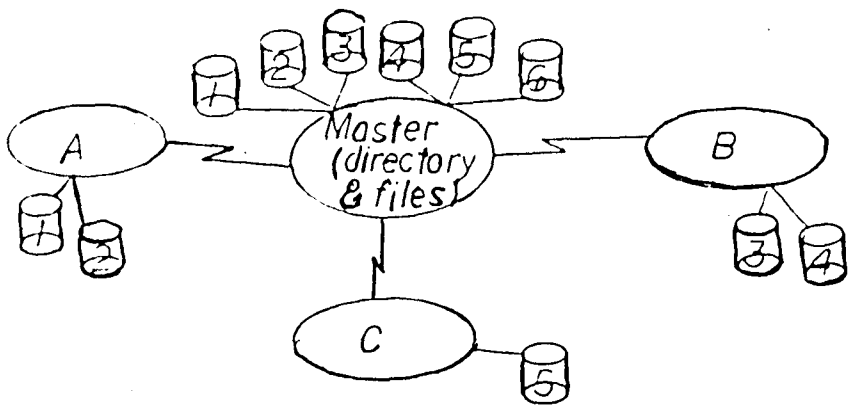


Figure 1.5

Vertical Distribution Model

2) A partitioned data base consisting of data most frequently utilized by a particular node is

retained at that processing node, however, no duplicated files are kept at a master node. The master node would still exist but only have a complete directory for routing of transactions that are the cause of local exceptions. This arrangement is frequently referred to as a horizontal distribution model.[Scherr 78]. This is illustrated in Figure 1.6 .

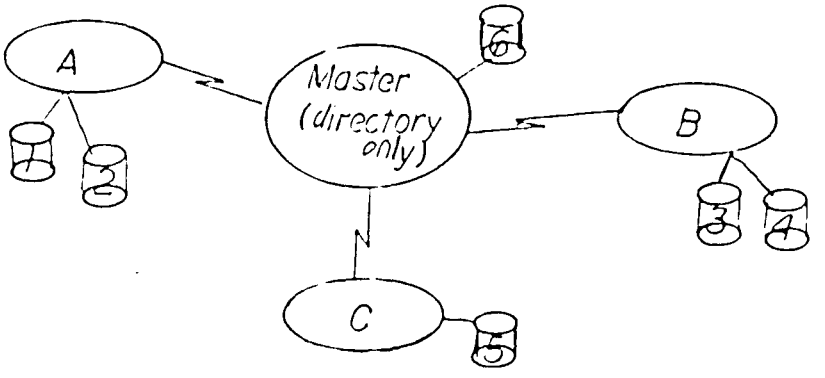


Figure 1.6

Horizontal Distribution Model

3) A partitioned data base with no master node, files or directory. There is no single source of information concerning the location of a specific collection of data. This is what would be considered a heterogeneous data base

model. In the other cases, the data base model would be considered homogeneous. This is illustrated in Figure 1.7 .

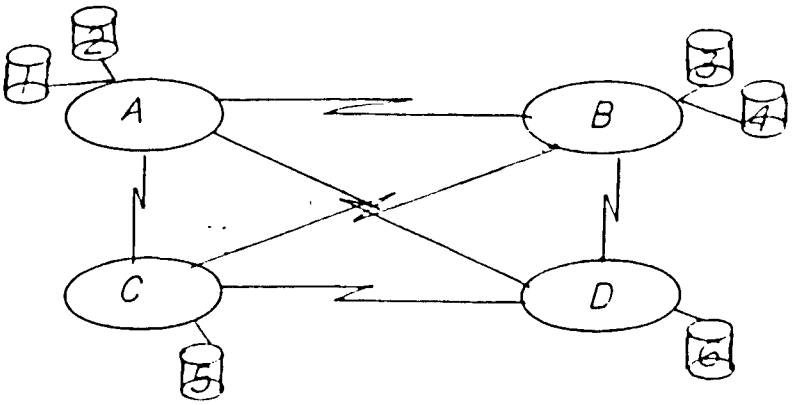


Figure 1.7

Heterogeneous Distribution Model

1.2 Contribution

Numerous problems are inherent in the creation of any form of a distributed data base system. The potential benefits of distributed data bases cannot be obtained without considerable efforts. Because of the size, and more importantly, the potential for interaction among the nodes within the system, the design of a distributed data base system poses many problems.

1.2.1 The scope and content of the data base

The first issue encountered in the design of any data base management system is how to define the scope and content of the data base. The ultimate objective of any data base is to make applications development easier,cheaper,faster and more flexible. [Martin (b) 76; Ullman 80]. This objective also applies to the distributed case.

In order to attain these goals, the data base must be defined in ways so that it represents the inherent structure of the data. This is accomplished by deriving logical data structures which represent the associations inherent in the data; ie. the true properties of the data. When the data base is structured in this fashion, an organization's information system becomes more than simply a labor saving device that supports the activities of people in one or more departments. In fact, it becomes a device that can help control, plan and coordinate. It should be designed to fit an organization's formal structure and facilitate the achievement of its business goals.[Buchanan 80].

The case of defining the scope and content is a further extension of the general central data base system. The definition of the scope and content of the distributed data base model is a problem in that it is an attempt to represent in its structure the data structure of the organization that it is to serve. This involves , in many cases, overlaps and a reasonably high chance of change because organizations are dynamic . It is this interface with the future that is the most difficult and important aspect of the distributed data base model. The desire is to have both physical and logical data independence. [Martin 77].

a) Physical data independence: when changes are made to the physical organization of the data or to the hardware, the changes should be reflected in the DBMS software, but, should leave the applications programs untouched.

b) Logical data independence: must be possible to add new fields to a record without having to rewrite any application programs that use that record.

1.2.2 Data Distribution

The location of the files and the question of what data are stored where will be determined largely by the balance of the costs between data storage and data transmission. Application programs frequently require data from two or more sources within the data base. Conversely, a single source or portion of a data base frequently supplies data to two or more application programs.[Baker 80]. Due to these interdependencies, the probability is high that programs in one group will require data located in another group. Clearly, this is not a simple question and is integral to the effective and efficient design of a distributed data base model.

1.2.3 Duplication in the Data Base

The inclusion of redundant data or copies can improve the overall reliability of a system because a damaged or unavailable piece of data can be retrieved from an alternate source. [Eckhouse 78]. Through this increased availability of data, a failure can be avoided. Retreival performance can also be improved because a local (close) copy can be retrieved rather than a remote copy.

Along with these potential advantages there is, however, the problem of update performance. Update performance can be degraded when several copies must be consistantly updated. Consistency control is far more difficult with redundant data.

1.2.4 The Concurrent Access Problem

The main operational problems in a distributed data base system result from the fact that data is a shared resource. Concurrency control is a major problem that results from this fact. In any large on-line system, it is necessary to be able to access the same data base partition from multiple concurrently executing procedures. Along with this, there is also the need to prevent these concurrent users from interfering with one another or at least the interference must be controlled to an acceptable level. [Ries 79].

Concurrent access control can be accomplished by means of the locking of data base elements such as records, pages, areas, sets, fields, etc.. This locking can delay other procedures that require access to the same element but does prevent the damage caused by conflicting

use of the data base. These delays can be quite long if an update procedure must accomplish modifications at several locations. In general, locks are used to ensure other procedures will not access certain elements until the updating is completed. [Potier 80].

Whenever concurrent access to multiple resources is allowed, the possibility of a deadlock situation exists. The deadlock or deadly embrace is a complex case of delay caused by the situation in which a procedure has one element locked and wishes to lock a second, while another procedure has the second element locked and needs the element locked by the first procedure. As can be seen, each procedure can wait for the other to release its lock indefinitely unless resolved by the system software.

The major point with respect to the problem of concurrent access is that it is an issue that must be dealt with at the design level. This will ensure that available software can be utilized to support the integrity and control needed in a distributed data base environment.

1.2.5 Privacy and Security

Access to data held in one location by a user from another location can require duplicated security checking. In addition, transmission of data between sites incurs the risk of interception.

In the traditional centralized approach with only local batch input and output, simple methods can be used to provide security . An example is locking the door to the computer area. In the distributed environment, the situation of access control is far more complicated because of the physical data dispersion and the difficulty in deciding where to locate the control mechanisms. If access control is distributed to every node, implementation can become quite expensive. Yet, if a centralized access control mechanism is used, performance can be heavily penalized [Date 79].

In a data base environment, security and privacy are also problems. The data base can be used by many different users with different security requirements. They can have access to a common pool of data with varing degrees of sensitivity.[Woods 80].

The combination of both the distributed environment and the data base environment creates a complicated and delicate situation. This must be considered from many angles by the designers of a distributed data base system.

1.3 Overview of the thesis

The potential benefits of a distributed data base system are very desirable. Listed below are some of the claims made for the case of distributed processing. In general, they can also be applied to the distributed data base case .[Enslow 78]

1) High systems performance
2) Fast response times
3) High throughput
4) High availability of the system
5) High reliability of the system
6) Reduced communications costs
7) Graceful degradation (ie. if one portion of the system is down, the entire system is not crippled).
8) Ease of modular, incremental growth and configuration flexibility

9) Resource sharing

10) Automatic load sharing (balancing)

11) High adaptability to changes in work loads

12) Easy expansion in both capacity and function

13) Easy adaptation to new functions

14) Good response to temporary overloads

Contrasted with these potential benefits are many problems. These were briefly discussed in section 1.2 . The problems illustrated are intimately intertwined. This is true because a solution generated for one can have great implication for the others. For example; the method used to provide security will be effected by how the data are distributed among the sites. This in turn will have an affect on the question of how to control concurrent accesses and how to guarantee integrity.

1.3.1 Types of problems to be solved

There are many technical problems in a distributed data base system. These relate to the lack of common networking and data standards.[Pouzin 78, Green 79, Zimmerman 80]. Without such standards, transfer of data is a major problem. Considerable work is being done by

standards groups such as; CCITT, ISO; to rectify this and other networking and communications problems.

In addition to these technical problems, there are operational issues that need to be solved.[Germano 80]. These include the development of software to deal completely with problems such as resolution of concurrent access, syncronization, and security/privacy problems. These areas are being researched by many large commerical organizations and research groups. Although no major vendor to date has been able to provide a comprehensive and acceptable software package to deal with the distributed data base environment, some smaller software houses are beginning to develop systems to deal with small distributed data base situations. Examples include, Appplied Data Research ,Cincom Systems, Britton Lee and Computer Corporation of America .

The last major area of concern is that of the design of the distributed data bases. [Lorin 81]. The design of distributed data bases as yet lacks a coherent and effective methodology. Two possible cases must be examined when dealing with the topic of a methodology to design a distributed data base. First is the case of the heterogeneous systems of several linked pre-existing data

bases. For this particular case, it is unlikely that a methodology can be prescribed due to the variations possible among such systems. As an example; the case of cross interfacing between a Codasyl data base and an IMS data base could prove so difficult as to negate any possible benefits attainable by any information and/or processing exchange. The second case is that of the homogeneous distributed data base in which a single data base is distributed among several sites. [Zigler 79]. For this second case, a coherent methodology could be developed.

1.4 Plan of the thesis

The combination of the concepts of distributed data processing and data base management systems can offer great benefits to the field of information systems, however, this marriage is not easily accomplished. Problems exist in terms of technology, operations, management and design. Conversely, benefits such as reduced communication costs , greater overall system reliability, and more control of locally held data

resources, are attainable. It will be an evolutionary process in which the problems of distributed data base management system will be resolved in order that benefits of such systems may be enjoyed.

The objective of this thesis is to develop a means by which information systems and user management can begin to investigate the potentials of distributed data base systems. In order to accomplish this , a number of goals must be attained. First, it will be necessary to define what is meant by distributed data bases. Second, once DDBS's are defined , various approaches will be illustrated. Third, the impact that design methodologies for a centralized DBMS's can have on the design of a DDBS will be examined. In total, the methodology developed should allow information systems management to have:

1) A functional idea of what a DDBS is and is not.
2) A functional knowledge of various options and available alternatives.
3) A means by which DBMS design methods can be applied to the case of a DDBS.

4) A general method by which a design of a DDBS could be started.

The explict definitions and models to be used in the development of this methodology will be defined in Chapter Two. In Chapters Three and Four, system possibilities and potential approaches will be discussed. The purposes of the development of the methodology will be to delineate what must be considered and to define what possible alternative paths may be followed. The fifth and final chapter will be a summarization of the research effort and recommendations.

2.0 Introduction

In many newly developing technological areas, understanding is hampered, to some degree, by imprecise terminology. The entire concept of distributed processing has been plagued by a wide assortment of nebulous definitions, many of which are at odds with each other [Enslow 78]. In the area of distributed data bases, the same situation exists. A wide variety of systems, real and conceptual,have been classified under the title of distributed data base systems (DDBS). Researchers and scholars addressing particular aspects of DDBS have often proceeded on premises that are quite different than those adopted by others working in the same field [White 79].

In Chapter One, an overview of the reasons for developing a DDBS's were given. The overview included the descriptions of the motivating factors that are leading to the design and eventual usage of DDBS's. In addition, important technical and operational problems associated with a DDBS were described. This overview was intended to give the reader an understanding of the framework within which further investigation into the design of distributed data base systems can take place.

The goal of this, the second chapter, will be to describe in detail what a DDBS is by definition. Further, the possible structures of a DDBS will be illustrated. The explicit definitions and models described in this chapter will be the basis upon which a design methodology will be developed. The actual design issues will be presented in subsequent chapters.

2.1 Definitions

At the highest level, three basic categories of systems can be identified [Champine 77].

1) Centralized - all processing and storage located in a single geographic location. This type of system can also have terminals attached at a remote location to give the central system input or accept output. No processing is done at a remote site.

2) Decentralized - a system composed of independent computers located in different geographic areas. These separate computers do not have the ability to

communicate.

3) Distributed -a computer system that is composed of two or more computers located in differing geographic areas which communicate and process application software cooperatively.

Schematically these basic categories of systems are illustrated in Figures 2.1, 2.2, and 2.3 .

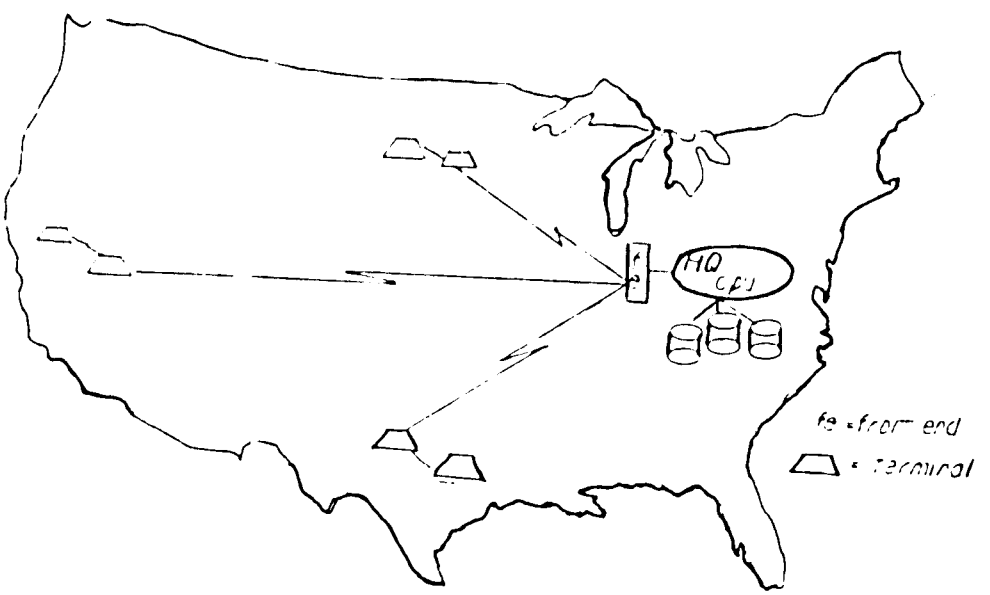


Figure 2.1

Centralized System

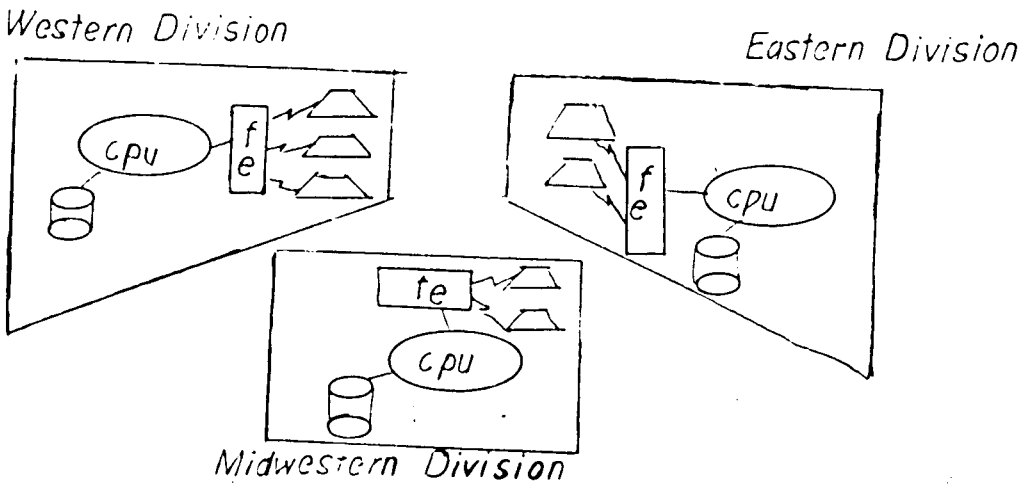


Figure 2.2

Decentralized System

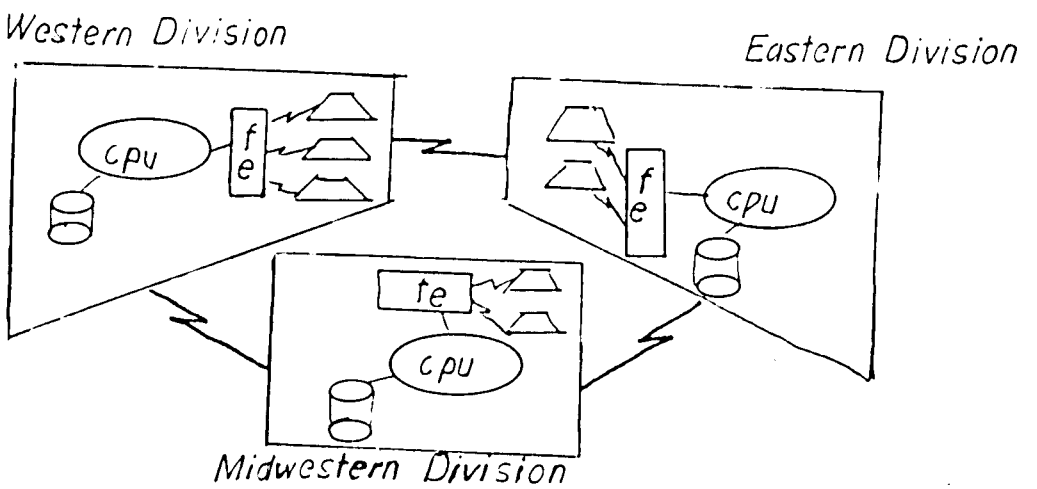


Figure 2.3

Distributed System

[Booth 81, pg.155]

A distributed system, as shown in Figure 2.3, is composed of a number of elements. Each node in the system has a complete computer system at a single location. Further, there must be two or more nodes. The composition of each node includes a host computer; processor and storage; communications hardware and software, operating system and application oriented software.

2.1.1 Variations of Distributed Systems

Within the realm of the distributed systems environment, a further classification is possible. A distributed system can operate using 'flat files' or a data base management system. The distinction between these two types of systems and between multiple independent data bases has been indentified succinctly by Booth:

"A distributed database exists when related data elements are stored at two or more processors within a distributed system. The term ' related ', when applied to the elements of a distributed data base, is quite flexible; the relationship may be very close and require a great deal of coordination among the processors, or it may be very loose requiring only minimal coordination. In any

case, the existence of this relationship distinguishes a distributed data base from multiple independent databases." [Booth 81]

Within every organization, there exists what is known as a conceptual data base. The conceptual data base is the collection of all the data necessary and of interest to an organization. In no reasonable case is the entire conceptual data base implemented in the form of computerized data storage/retrieval system. Only the elements deemed to be most important are captured and retained in the computer based system. The portion of the conceptual data base that is captured is composed of data elements. These data elements can be divided into groups of relationships according to their respective entities and attributes. When related data elements are stored in two or more sites and the relationships among the elements cause interlocation access and/or coordination between locations, a distributed data base is formed. [Booth 81].

2.1.2 Classes of Distributed Data Bases

The definition of a DDBS given by Booth and others [Champine 77, Davenport 79, Rothnie and Goodman 77] assumes that there is only one logically integrated data base in the distributed system. This particular assumption is not shared by others in the field [Whitby-Stevens 79, Weber, et al 79]. Others extend the definition to include the possiblity of multiple communicating, differently configured data base sytems. These differences identify the two basic classes of distributed data bases.

The first class is known as the homogeneous distributed data base system. In these systems, a federation of data bases are coupled to form one logically integrated data base.The characteristics of this class of DDBS include use of similar or the same hardware configurations. In the homogeneous class of DDBS, an integrator is needed to provide the data and procedures so that the collection of data bases act as a single overall entity. [Davenport 79]. This integrator would generally be in the form of a master directory and/or master schema.

The second class of DDBS is known as heterogeneous DDBS. This class of DDBS is characterized by the federation of distinctly different data bases. These different data bases may use different supporting hardware and utilize different DBMS structures. In the heterogeneous class, the same need for an integrator exists as in the homogeneous class. In addition, there is a further complication in that some type of translator is required to coordinate the data base systems and allow communication among them. An example of the need for a translator is the case of communications between a relational model DBMS and a network DBMS.

In general, the problems associated with the design and implementation of a DDBS are increased exponentially with the multiplicity of data bases in a system and their heterogeneity.[Davenport 79, Booth 81]. Most of these design problems are due to a lack of standardization of data base management systems, data communications systems, and teleprocessing architectures.

Figures 2.4 and 2.5 illustrate schematically the two classes of distributed data base systems. Both figures reflect the organizational framework in which such a structure might be found. In Figure 2.5, the

heterogeneous case, different organizations are represented. It is possible that the heterogeneous case could exist even within a single organization if different DBMS packages and/or different hardware configurations are utilized. Such a case is easily conceivable if an organization has no standard policy for acquiring hardware and software.

The most important fact noted in Figures 2.4 and 2.5 is the need for an integrator to ensure the continuity of the data base structure. Further, the heterogeneous case requires the addition of a translator so that data elements can be exchanged.

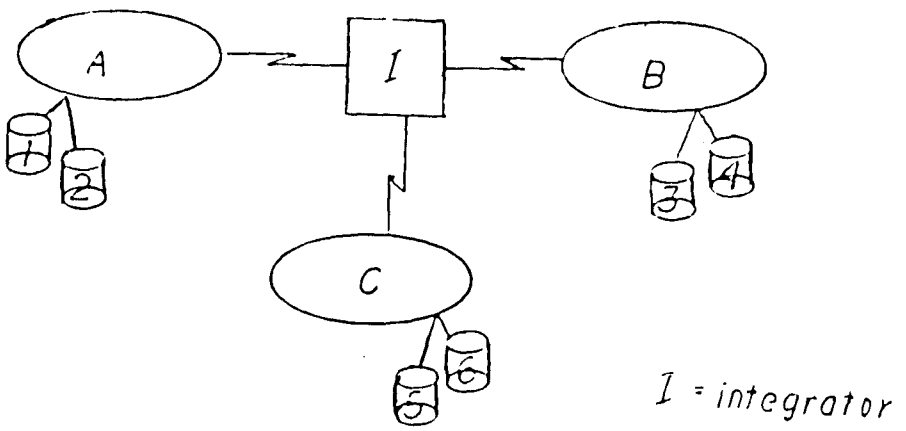


Figure 2.4

Homogeneous DDBS

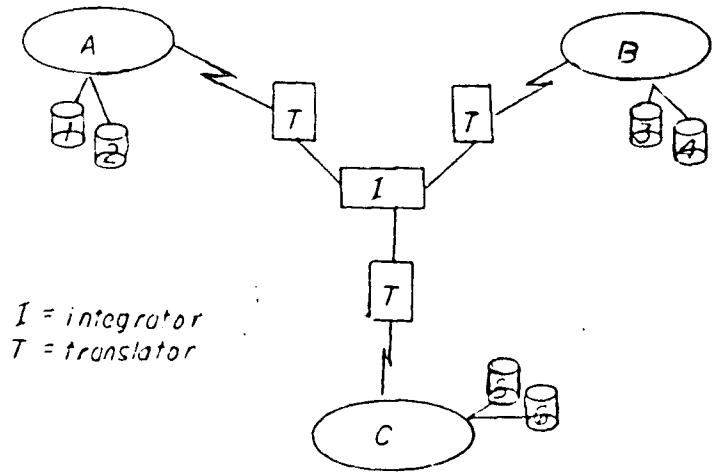


Figure 2.5

Heterogeneous DDBS

2.2 The Structure of Distributed Data Bases

To progress further into an understanding of the taxonomy of DDBS's, two basic structural types of distributed data bases can be identified. [Davenport and Palmer 77, Booth 76]. The first structure is known as a partitioned data base. This structure is characterized by the division of an organization's conceptual data base into non- redundant segments and distributing the

non-redundant elements to two or more information processors. The second type of stucture is known as a replicated data base. The replicated data base structure is characterized by placing copies of all or parts of the data base files at two or more locations.

Although the two basic structures can be clearly identified, it would not be uncommon in a highly complex system to have a hybrid combination of the two structures.[Booth 81, Mohan and Yeh 79] The combination of the two structures could be adopted for practical reasons such as matching data base structures to the overall structure which it serves.

2.2.1 Partitioned Data Bases.

A partitioned data base is a data base which has been decomposed into physically separate units then distributed to multiple nodes within the system. The partitions that are physically distributed must logically form a single overall data base. Schematically, the concept of a partitioned data base is illustrated in Figure 2.6.

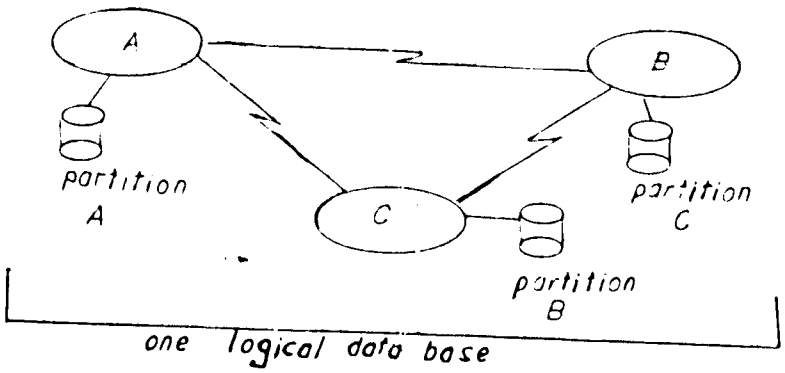


Figure 2.6

Partitioned Data Base Structure

[Booth 81, pg.50]

The partitions of the data base are composed of non-redundant data elements. These specific pieces of data are stored at only one location.

There are three reasons why a data base might be partitioned: [Booth 76]

1) The most common reason would be to locate each data element as close as possible to where the demand for that data originates.
2) A data base may be so large and accessed so frequently that the load cannot be handled by any single processor.
3) A data base may be partitioned in order to

minimize the cost of transmitting data to and from remote locations.

With respect to reason number one, there is an assumption that partitioning is possible in identifiable and justified natural groupings of data elements. An example of this type is an airline reservation data base. In practice, it is known that most flight reservation requests come from the same general area where the requested flight originates [Booth 76]. In this situation, the data base could be partitioned into three segments. The first segment would include data on all flights originating on the east coast, the second to contain data on all mid west flights and the third, data on all flights originating in the west. Thus, one overall logical data base is partitioned in three distinct units, with overall interconnection.

Reason number two for partitioning a data base is becoming more important with the ever increasing number of very large and complex on-line systems found in organizations throughout the world. Modularization or partitioning has become one of the most important concepts in use today to reduce the complexity in the design and

operation of large sytems.

Data communications costs have been a significant factor in large and small on-line sytems for some time. The probability is also very high that these costs will remain a significant factor in all large on-line systems. Notable savings can be attained by distributing processing and data base facilities into regional centers or other geographically benefical centers.

Due to the variety of usage patterns associated with a data base, a distributed data base system can be partitioned in several ways. The two most basic methods are the horizontal processing structure and vertical processing structure. [Joyce 77] [Note: vertical structure is also referred to as hierarchical processing structure]

2.2.1.1 Horizontally Partitioned DDBS

The Horizontal processing configuration has been called the "ring system" because of the basic network topology on which it depends. Additionally, it has been called the "peer group" system because of the equality, in

terms of control, that exists between the nodes. [White 79]

In the horizontally distributed system there is no central computer. Each node is a center in its own realm, yet is a partition of the single overall logical data base. All processors in the system cooperate logically at an equal level.[Palmer and Davenport 77] An example of this type of system can be seen in the airline system noted earlier where data elements can be naturally grouped into geographic partitons. Schematicaly this situation is illustrated in Figure 2.7.

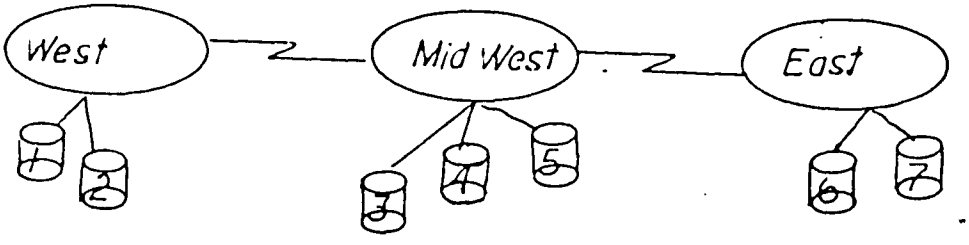


Figure 2.7

Horizontally Partitioned Data Base

2.2.1.2 Vertically or Hierarchically Partitioned DDBS

Another method of forming a partitioned data base is to distribute the data elements found in a particular logical data base across a systems hierarchy.[Booth 81] This systems hierarchy is the processing hierarchy which corresponds to a particular organizational hierarchy. The hierarchically partitioned data base is also refered to as a vertically partitioned data base.[Schneer 78]

In the hierarchical system, it is possible to identify a central computer by virtue of its placement in the architecture and function in the system. Generally, the highest level in the hierarchy is the corporate level processor and attached are any number, within the limits of the system architecture, of satellite processors. Each satellite maintains its own local data base partition where there is no duplication of data elements. All partitions of the hierarchy can therefore be summed to equal one overall organizational data base.

An example of the hierarchical or vertical partitioned data base is shown in Figure 2.8.

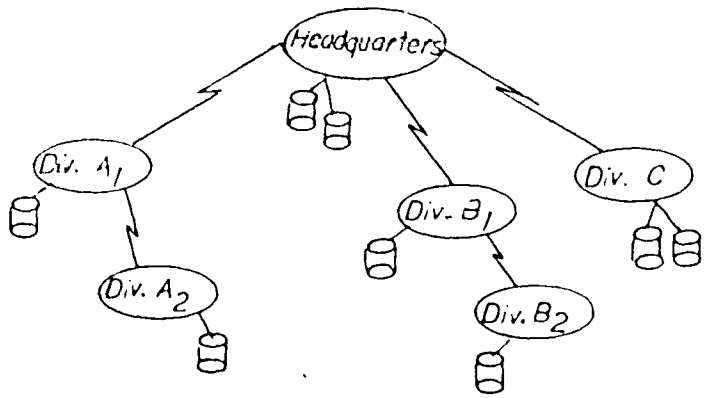


Figure 2.8

Hierarchically Partitioned Data Base

Information exchange can occur both up and down the hierarchy. Hence it's being refered to synonymously as vertical.

The hierarchically partitioned data base is likely to become the most common configuration.[Champine 79] This is felt to be true because of the general philosophy that the data processing organization should be congruent to the people in the organization it will effect. Since nearly all organizations are hierarchical in nature, a hierarchically distributed data base system fits well.

2.2.2 Replicated Data Bases

The second of the two basic structures of distributed data bases is known as a replicated data base. It is simlar to the partitioned data base in that replication serves to move data elements close to the points where requests originate. The basic difference is that replication is used to create the distributed data base, so duplicated data are moved rather than original data.

There are several reasons why a data base may be replicated.[Booth 76, Booth 81]

1) As in the partitioned case, one reason is to move data to where the processing activity can most optimally occur.

2) Replication can serve the same purpose as data base journals by providing a backup copy of the data. This advantage is subject to synchronous updating procedures.

3) Selective replication can increase data security and privacy by having publicly available data replicated and sensitive

data retained at a single location with strict privacy protection procedures.

4) Selective replication can also serve as a means of spreading out the processing load on multiple processors.

As in the case of the partitioned data base, the replicated data base can be formed in two ways. First, by copying data within a hierarchily distributed system. Secondly, by copying data within a horizontally distributed system. [Salter, et.al 77, Booth 81].

2.2.2.1 Hierarchically Replicated DDBS

In the case of a hierarchical replication, the database is structured to be congruent to the organization using it. In this type of system, the host processor maintains a master copy of the data base files and directory. Satellites of the host processor contains copies of the part of the master data base that fulfills local needs. Therefore, each local data copy replicates only data required for a particular satelitte.

A schematic diagram of a hierarchically replicated data base is shown in Figure 2.9.

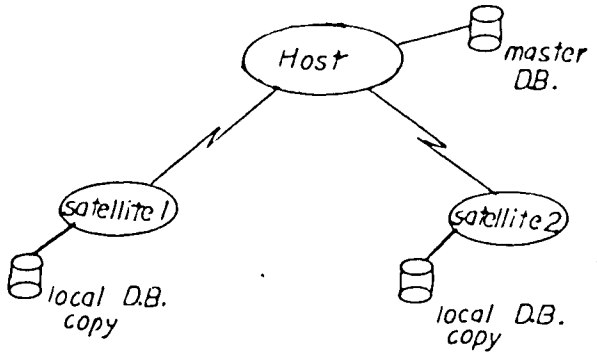


Figure 2.9

Hierarchically Replicated Data Base

The case of a hierarchically replicated data base has a number of possible uses. One example is in a production environment where a host would contain master information concerning all aspects of the production operation. From this master, satellites would be connected. The master would contain replications of data, such as scheduling data needed for that satellites particular production operation.

2.2.2.2 Horizontally Replicated DDBS

In general, a data base cannot be only horizontally replicated. In its pure form, this situation would be exemplified by a data base containing exactly equal data elements across two or more communicating data base systems. Such a case is hard to imagine, let alone place a value on its practicality. For the sake of clarity, an example might be seen in the case of a number of warehouses that stock and sell identical items. In this hypothetical data base, each would be maintained as to stock quantity changes, etc for all locations so that each data base location always had equal information.

2.2.3 Hybrid DDBS

The techniques of replication and partitioning can be used separately, however, can be combined to optimize a distributed data base structure. [Salter, et al 77] The formation of hybrid DDBS structures can occur by replicating data elements that are static in nature and partitioning those elements that are dynamic in nature.

An example of a hybrid data base structure can be developed from the example given in section 2.2.2.2. The same warehouse situation could be both partitioned and replicated in order to optimize the system. This can be done in the following way: warehouse A will maintain complete information on stock quantity, etc. it holds locally; warehouse B and C would do the same. This represents the partitioning. In addition to the detailed information kept on all locally held quantities, rudimentary data would be kept on stock held at other locations such as who has what and in what quantity. This represents the replication. Thus , such a hybrid arrangement allows each location to check locally stocked items and also allows each location to check location and other framework information on items not held locally. Figure 2.10, depicts this situation graphically.

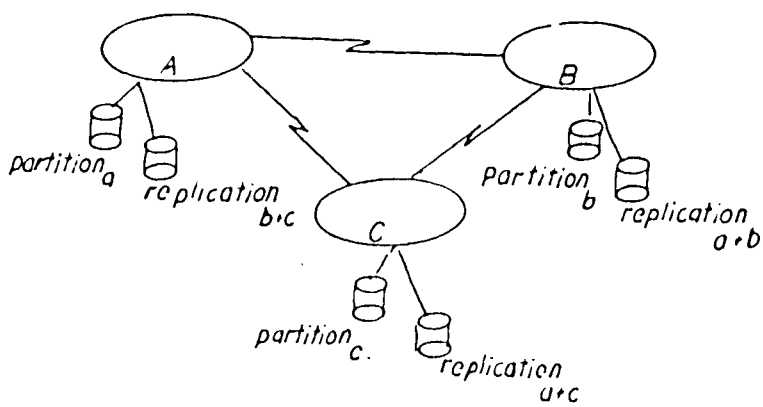


Figure 2.10

Hybrid DDBS

Hybrid DDBS

It is most probable that the hybrid case of both replicated and partitioned distributed data bases will be the most useful in the future. The reasons for this are its inherent flexibility and its ability to allow optimization of the distributed data base structure.

Even though the hybrid situation is a desireable option, the decision to partition and/or replicate data elements is an extremely difficult design problem at this time. [Booth 81, White 79] In general, the actual configuration of a DDBS is largely determined by the needs of the applications. [White 79]

2.3 Summary

The preceeding sections in this chapter have defined what a distributed data base can be. Further, the various alternative classes and structures of distributed data bases have been described. A summary of the taxonomy of DDBS's is presented in Figure 2.11.

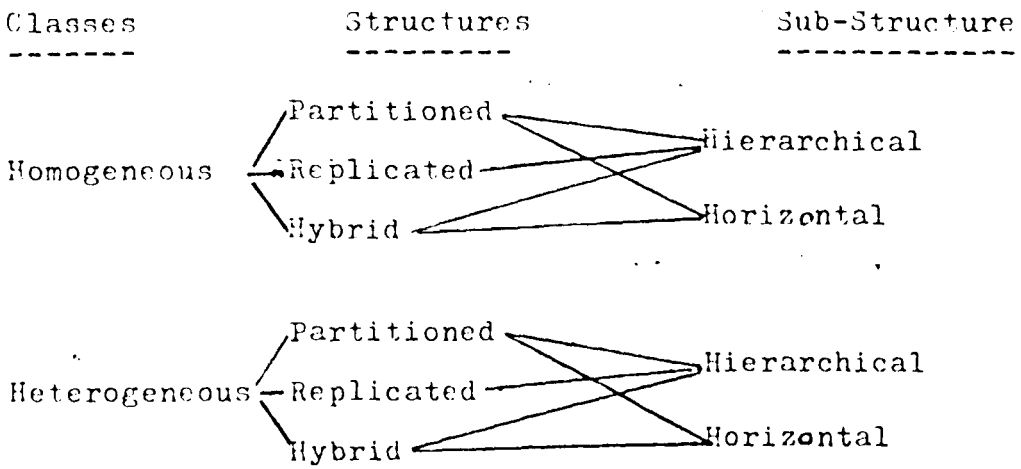


Figure 2.11

DDBS Taxonomy

The essential features of a DDBS can be summarized as follows: [Davenport 79, Booth 81]

1) The system must posses two or more geographically separate computing facilities. The composition of these nodes will include a CPU with main and secondary storage as well as communications capability. Generally each node will service a separate organizational subunit.
2) All nodes within the system must be linked. Normally this link is accomplished via tele-

communications.

3) All nodes in the system must have stand alone capability.

4) The data base used in the system must be one in which the data elements within the system are partitioned and/or replicated. Further, there must be a logical relationship among the data elements used in the DDBS.

The information presented in this chapter will serve as a beginning upon which the difficult questions of how to design a distributed data base system can be addressed. Without a doubt, the various alternative structures are applicable to vastly differing situations. It is this flexibility that is one of the most challenging aspects of the concept of distibuted data bases.

3.0 Introduction

In the design process of any system, it has long been recognized that a methodology must be developed and utilized. The methodology developed should address all the design issues rationally to achieve a successful system.

In the area of systems design,there has been a great deal of literature published during the past few years. Several design strategies have been proposed. These include the structured methodology of Yourdon, Constantine and DeMarco. In the area of classical methodologies, there are those of Semprevivo and Ross and Murdock. All these strategies are aimed at addressing the complex general problems of a computer system design. The general issues addressed by the currently available systems design methodologies are good starting points, however, all general approaches must be modified depending on the type of system being designed.

3.1 System Possibilities

Because the characteristics of any design are a function of the type of system being designed, the starting point of this chapter will be a definition of the nature of the system to be considered. The definition can be derived by comparing a DDBS to other possible types of systems. Figure 3.1, illustrates the possible classes that can be designed.

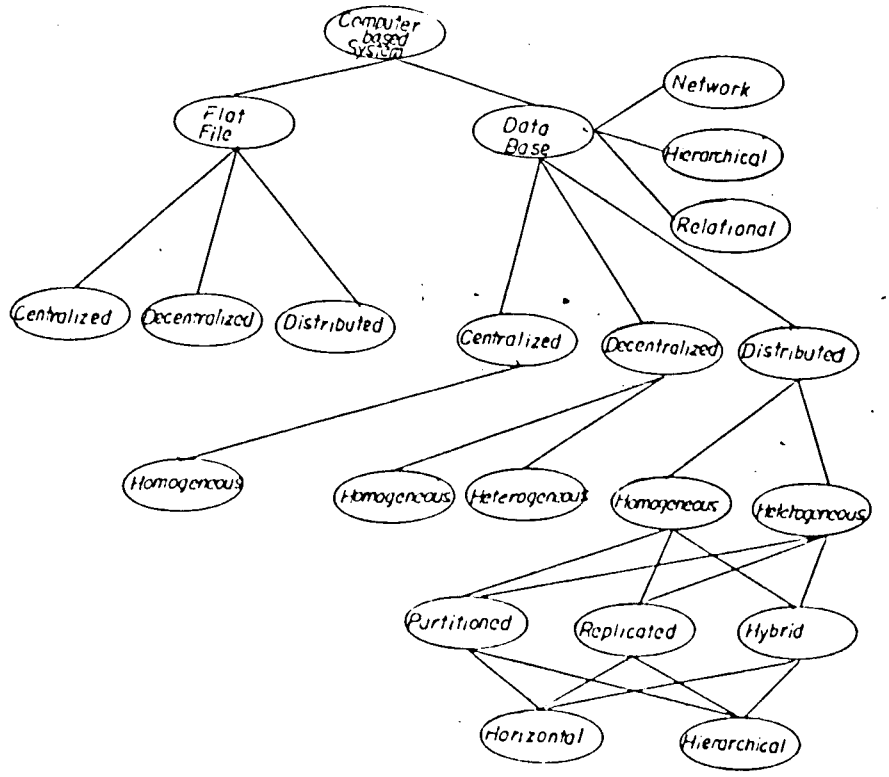


Figure 3.1

Possible classes of system

3.1.1 General Design Alternatives

The taxonomy depicted in Figure 3.1 illustrates many of the options available to a designer. For example;

1) At the top level, assuming the designer has chosen a computer approach, a basic choice can be made between a flat file (indexed, sequential, random, etc.) system or a data base system.
2) If a data base system approach is selected, a choice can be made as to which data model [Borkin 80] will be used. The possibilities include:
    a) Network model (eg. CODASYL)
    b) Relational model (eg. Codd 70,74)
    c) Hierarchical model (eg. IBM's IMS)
3) Assuming a data base approach has been selected, a choice of processing structures can be made. A data base system can be centralized, decentralized or distributed in nature.

The remainder of the figure's options were discussed in detail in Chapter two. The point that should be identified from Figure 3.1, is that distributed data bases are forms of general data bases. The concept of DDBS is to benefit from the standard data bases control of integrated data while not imposing a centralized organization. [Adiba 78]

3.2 Approaches to DDBS Design

Two basic approaches to design a DDBS, or any complex system, can be identified.

1) The Top-down approach
2) The Bottom-up approach

These two approaches have received a great deal of attention in the literature. [Freeman 78, Denning 76, Yourdon 80, DeMarco 78]. Each will be examined with respect to their applicability in the design of a DDBS.

3.2.1 bottom-Up Design

The bottom-up approach is a method in which the lowest level decisions are made first. Gradually the system is built from the lowest level to the highest level

concepts. This approach is also termed the method of successive compositions. [Denning 76] The basic characteristics of this method is a sequence of steps beginning with a set of base functions through which a series of iterative steps are developed to a single overall complex function.

3.2.2 Top-Down Design

The most popular of the two methods is the top-down design. The top-down method is characterized by a sequence of steps beginning with global design issues which gradually progress to levels of greater and greater detail. The level of any decision is often a function of its distance away from actual implementation concerns. [Freeman,78] Thus, a decision on the general contents of a report would be considered top-level whereas the actual format of the report would be considered low level.

3.2.3 Applicability to DDBS

Both of the two main system design approaches; top-down and bottom-up; are applicable to the distributed data base environment. Diagrammatically, the two approaches with respect to a DDBS are shown in Figures 3.2

and 3.3.

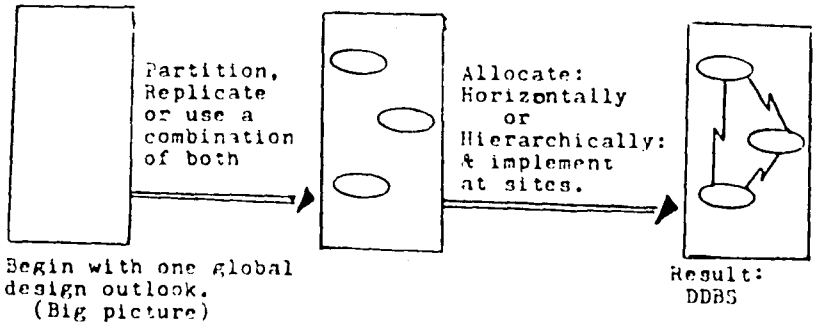


Figure 3.2

Top-Down Design for a DDBS

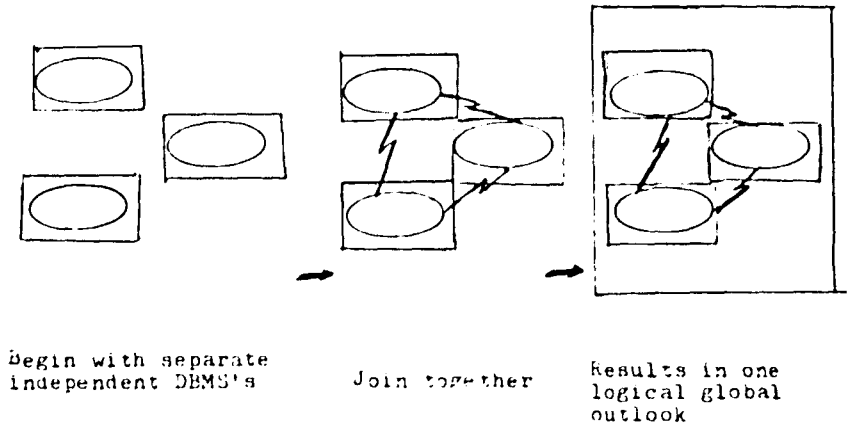


Figure 3.3

Bottom-Up Design for a DDBS

The top-down approach, Figure 3.2, is generally associated with systems being designed from scratch [Adiba 78, White 79] The bottom-up approach, Figure 3.3, is associated with systems of independently existing functions or possibly entirely separate DBMS's that are going to be integrated to form a DDBS. [Adiba 78, White 79]

The integration of two formally distinct systems leads to the complex design task known as the post-cooperation problem. [Mohan and Yeh 79] The post-cooperation problem leads inherently to a bottom-up design. Conversly, a planned cooperation situation points to a top-down design.

There is growing agreement that distributed data base systems should be designed, whenever possible, using a top-down methodology. [White 79, Joyce 79, Davenport 79, Patrick 80] The consensus is due primarily to the problems that develop when a designer becomes involved immediately with detailed considerations that exist at the low ,operational, level. This approach in a complex system can lead to a cloudy view of what the designer's main objectives are in developing a DDBS.

Naturally, there are situations which preclude the use of a top-down methodology. Regardless of which methodological approach is used, the final product must be a single logical design of a data base system which is distributed to maximize the benefits associated with a DDBS.

A comparison of the broad elements shown in Figures 3.2 and 3.3 , would seem to point to the conclusion that the end results achieved by both processes should be the same. In reality the end results are rarely equal. Figure 3.4 illustrates this difference between theory and practice must be considered when selecting a methodology.

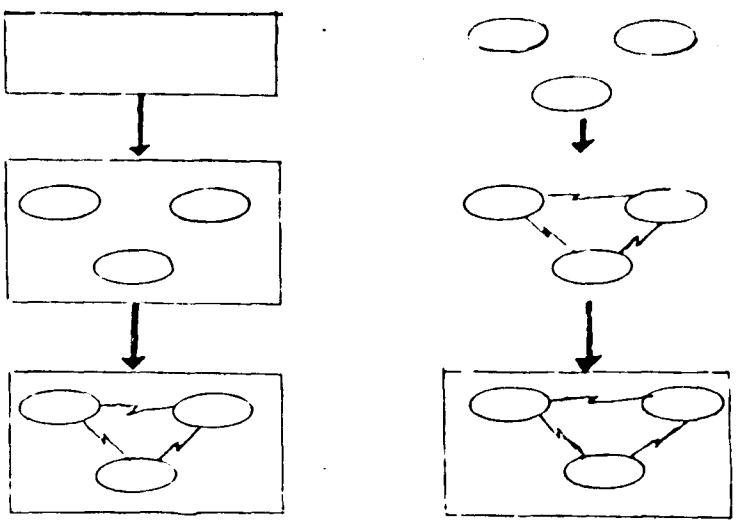


Figure 3.4

Comparative Summary of End Products

Top-Down and Bottom-Up

3.3 Stages in the Design Process

Before proceeding into a detailed design strategy, it is appropriate to take a step back and look at the overall process of system design. A data base systems life cycle will provide an overall view of design from a management perspective.

The main phases of a data base systems life cycle have been identified by Atre. [Atre 80].

1) design the data base
2) physical creation of the data base
3) conversion of the existing data sets and applications to match the newly created data base
4) integration of the converted applications into the new data base
5) operations phase
6) growth, change and maintence phase

[**note: steps 3 and 4 can be eliminated if existing data sets are not to be converted to the newly created data base] [Atre 80]

3.3.1 Application Life Cycle

Contrasted with Atre's view, the major stages in the life cycle of a data base application have been identified by Teory and Fry. [Teory and Fry 78]. According to Teory and Fry, the major classes of inputs to and the results from the data base design process are as follows:

Major classes of inputs

1) User view of the organization, or organizational subunit, for which data are to be collected.
2) Data base objectives; user requirements
3) Users view of data to be collected and stored in the data base
4) Processing requirements:
   a) specific data elements required for each application
   b) the data volumes
   c) processing frequencies in terms of the number of the times each application will have to be executed in a specific time frame
   d) DBMS specifications and/or constraints
   e) operating system / hardware configurations

Major classes of outputs

1) The data structure (logical data base structure)
2) The storage structure
3) The specifications for applications programs based upon data base structure and processing requirements

The results of all the above will be the specifications for the data base implementation. The distinction between the specifications and the implementation are illustrated by the basic steps in a data base systems life cycle. Teory's and Fry's life cycle differs from that given by Atre's in that the former stresses the design phases. These basic steps are presented below:

1) Requirements formulation and analysis
2) Logical design
3) Physical design and evaluation
4) Implementation
5) Operation and monitoring
6) Modification and adaptation

[** note: 1,2 and 3 are considered the design phase]

[Teory and Fry 78]

The actual data base design occurs in steps one through three. The importance of this phase has been the subject of a virtual plethera of works; ie. [Yourdon 80, DeMarco 79, Patrick 80, Booth 81, Martin 76, 81, Langefors and Sundgren 75] The common conclusion reached by all is summarized in graphic form in Figure 3.5.

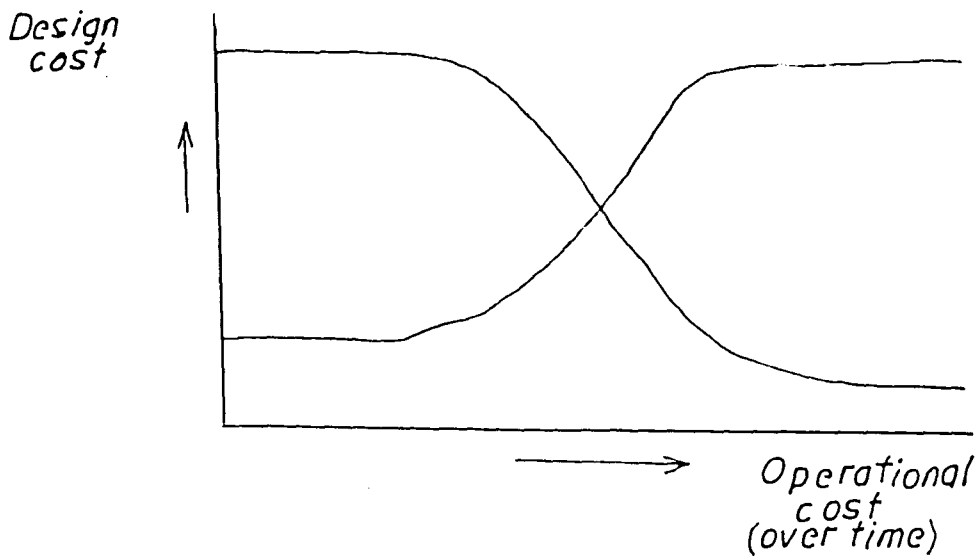


Figure 3.5

Effect of Efforts spent on System Design

[Semprevivo 76, pg.145]

Figure 3.5 illustrates two critical facts:

1) The systems development phase is,as a rule, far shorter than the operational phase
2) Cost analyses show that costs of operation and maintenance is several times the cost of developing the system. [Patrick 80]

Because of these facts, it becomes apparent that to spend more time, and as a consequence, more money on the initial systems design, the operational cost of maintence can be significantly reduced. In addition, it is important to note that an incorrect analysis during the design phase will ultimately result in an improper implementation and lead to a non-responsive application system.

3.3.2 Design Methodologies

Up to this point, emphasis has been placed on general approaches to data base design. Further, basic considerations having a significant impact on the development of a methodology have been discussed. The methodology to be developed in this report is concerned with the general processes shown in Figure 3.6.

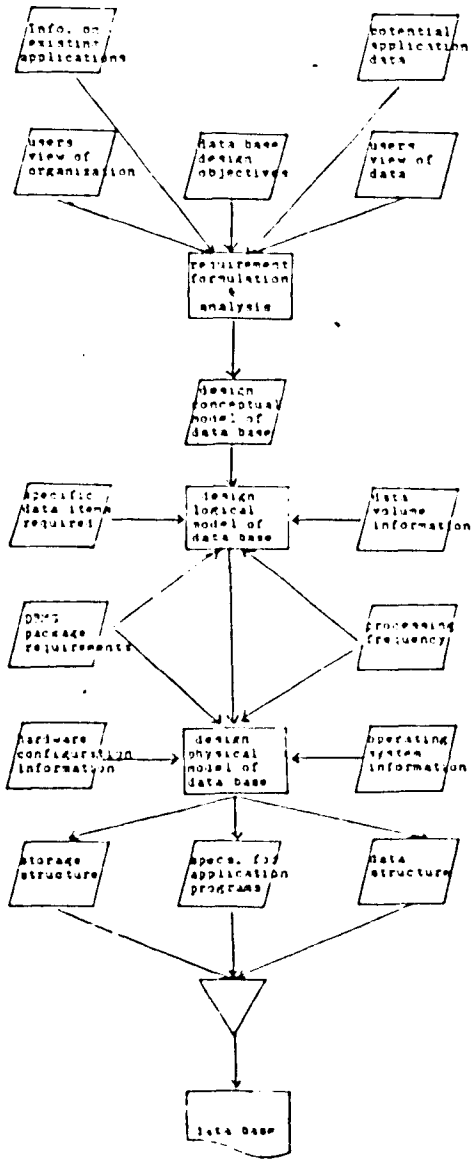


Figure 3.6

Data Base Design Processes

Although there has been a great deal of literature published on the topic of data base design, no widely accepted methodology or technique exists. There is, however, one common thought that spans the majority of data base design methodologies. This common thought is that a data base should be designed using a two stage design approach. The two stage design proposes that the logical and physical structures should be designed separately, then connected by means of a mapping function. [Yeh,et al 78, Atre 80, Martin 76]

The applicability of this commoness in DBMS design methods to a methodology to design a distributed data base system is simply that a DDBS is a form of general DBMS's. The goal is to establish the level of data integration needed by a system. Once attained, the decision of what to distribute can be considered.

The two stage approach is fundamental to the design decisions concerning the data base and its associated processes. The allocation of data and processes to hardware is a consequence of this design - not its starting point.

Any methodology formulated for the design of a distributed data base system must inherently be based on methods used in the design of a centralized DBMS. Naturally, a methodology for a distributed data base system will include considerations related directly to a distributed environment.

3.1 Process I - Data Analysis

3.1.1 System User Requirements

The need for the participation and assistance of the users in the design of a data base system has long been recognized. The collection and analysis of the systems users requirements is the first step in the data base development cycle. This step is probably the most difficult and time consumming of the entire process. The importance of this step is simply that the majority of subsequent design decisions are based on this step.

The major tasks associated with this process are: [Teory and Fry 78]

1) Collect information from all present and potential users of the data base on their information requirements and desires
2) Collect information from all present and potential users regarding their processing requirements.
3) Analyze the requirements to insure consistency of the users objectives.
4) Analyze the information gathered to minimize redundancy of information and data

The evaluation of the users information and processing requirements is called data analysis. [White 79, DeMarco 78] The objective of data analysis is to determine the fundamental nature of an organization's data resources and how these resources are used. The results of this analysis will be organized and documented to be used as the basis upon which the data base will be designed.

3.4.1.1 Structured Analysis Techniques

The process of data analysis can be completed using a variety of techniques. [Orr 77, DeMarco 78, Semprevivo 76] Of the techniques available, the structured analysis techniques offer unmistakable benefits. The structured methodologies provide a means of understanding and documenting complex environments. The techniques use tools such as graphic methods by which all the attributes and relationships in a system can be expressed clearly; a data dictionary to allow the analyst to keep track of the enormous flood of data involved in a system design, decision trees, decision tables and more. The result of using such a methodology is a highly structured specification which can be easily verified with the systems users and maintained over the systems life cycle.

An excellent treatment of the topic can be found in Thomas DeMarcos text " Structured Analysis and Systems Specifications " [DeMarco 78]

3.4.1.2 Entity and Functional Analysis

Another means by which data analysis can be accomplished is by using entity and functional analysis. [Davenport 79] This is a two phase method. Entity analysis is a means by which a system can be understood and documented in terms of its entities and attributes. The emphasis is on the elements which make up the system rather than on details of how they are used. Functional analysis is concerned with understanding and documenting the basic business activities that utilize the entities found in phase one of this approach.

The most important aspect of entity and functional analysis is that it is a means by which the fundamental nature of an organization's data resources can be understood, documented and eventually used as a sound basis for data base design.

The normalization process of E.G. Codd; [Codd 70]; is used in both structured analysis and entity/functional analysis techniques. In both cases, the process of data analysis is iterative because it is unlikely that a completely satisfactory model will be produced on the first iteration.

Both techniques mentioned can and should be employed regardless whether a centralized, decentralized or distributed approach is selected for use.

3.1.2 Design Objectives

In addition to the definition of user requirements at the outset of the DDBS and DBMS design process, there is a need to define the system objectives . These objectives are those which the final design must achieve to satisfy the users. Some of these objectives are quantifiable, such as performance figures, while others will be qualitative and therefore require a subjective assessment by the designers.

The following is a list of design objectives that would be most important in a majority of circumstances. [Martin 76]

1) The ability to represent the inherent structure of the data. The DBMS structure must be able to represent the nature of the data.
2) Performance. The data base system must be able to process transactions at an appropriate throughput rate within the framework of a

specified response time. A DDBS has an advantage in this objective because the processing power can be located in close proximity of the users location.

3) Economy. The DBMS must be able to produce acceptable performance at an acceptable cost.

4) Elimination or minimization of redundancy. The DBMS organization should try to eliminate data redundancy so inconsistencies can be avoided. Additionally, there are economic motivations to minimize redundant data.

5) Search capability. The DBMS design should allow a variety of search techniques for queries. This search capability should also be reasonably fast.

6) Integrity. The DBMS should be able to serve many different users without different users destroying shared data elements. In addition, the data base should be insulated from systems failures and ridiculously large data values.

7) Security and Privacy. Data security is concerned with insuring against accidental or intentional data disclosure to unauthorized persons, unauthorized modification or destruction of data. Privacy is concerned with when, how and to what

extent data can be accessed and/or transmitted to individuals and organizational subunits.

8) Ease of implementation. The DBMS must be able to be implemented. This is the ability to produce the systems processing requirements for performance with acceptable cost using acceptable resources. Further, a time frame for implementation should be met.

9) Flexibility. The system must be able to deal with changes in flow of transactions, change in applications , additions and deletions, and be able to incorporate new technology. DDBS's are particularly resilient to change because of the localized effect that a change can have.

10) Simplicity. Data structures should be made as simple as possible so that an overall logical view of the data base can be made in a simply understandable and neat way.

3.5 Process II - A Model of an Organizational Information Resource

The objective of the data analysis process is to produce the information needed to build a model of an organization's information resources. This model is described by a schema. The schema will serve as the basis for the data base design and as the central control element of the data base system.In order to be the best possible description of the data model, the ANSI/SPARC study group proposed a three schema model to support data base management systems [ANSI 76]

3.5.1 ANSI/SPARC Three Schema Approach

In an effort to avoid some of the current confusion about data base organization, the ANSI/SPARC study group has developed a method by which a distinction can be made between data as seen by the system and data as seen by the application programmers. These two levels are called the internal and external levels respectively. An additional level called the conceptual level is also identified. This level represents the organizations description of the information modeled in the data base. The description provided in a conceptual schema is that which can be used to resolve any disagreements between users and programmers over what is meant by specifications. [Steel 78] Diagrammatically, the three schema model is presented in

Figure 3.7.

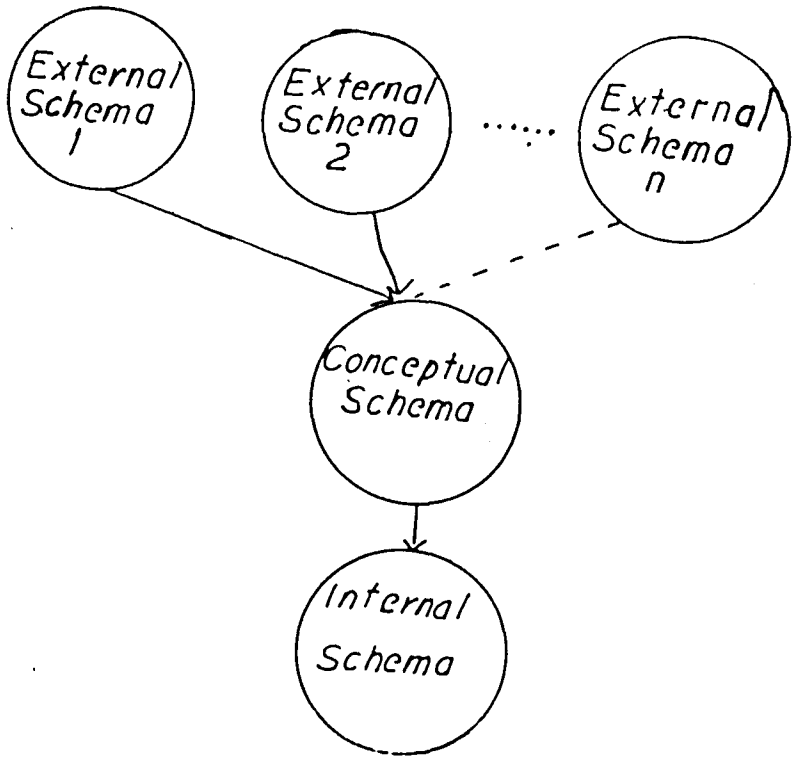


Figure 3.7

ANSI/SPARC Three Schema Model

The ANSI/SPARC model's impact on the development of the model that can be presented to the DBMS is illustrated in Figure 3.8. [Atre 80] As an example, the conceptual model that can be presented to the DBMS is called the logical model. The users are presented with a subset of this model which are called the external models. The logical model is then mapped to physical storage and the

internal model is then completed.

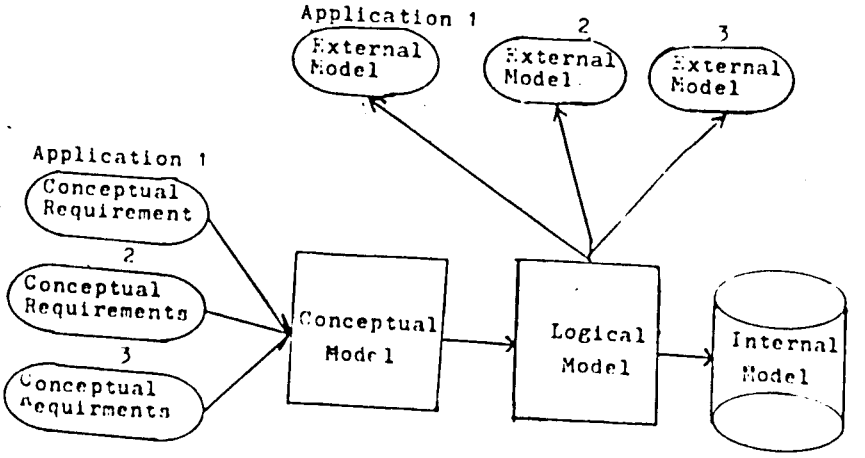


Figure 3.8

DBMS Presentable Model Transformation

[Atre 80, pg.19]

CODASYL has also adopted the three schema approach which corresponds to the ANSI/SPARC schemata. The relationship between these two groups follows: [White 79]

ANSI/SPARC	CODASYL
External schema	Subschema
Conceptual schema	Schema
Internal schema	Storage schema

3.5.2 Relevance of the Three Schema Approach to DDBS

In the distributed environment, the ANSI/SPARC concept of a single internal schema , Figure 3.7, can be extended to several sets of local internal schemata. Figure 3.9 diagrams this structure. [Adiba 79]

The DDBS is composed of local data bases described by their respective local external schemata (LES). A local internal schema (LIS) describes each physical data base at a particular host. A local conceptual schema (LCS) describes the local organization's description of the information modeled in the data base.

The union of all the local external schemata will result in a global conceptual schema that describes all the data modeled in the entire logical data base system in a unified manner at a DDBS level. The global external

schema will represent the view offered to the various applications. [Adiba 78]

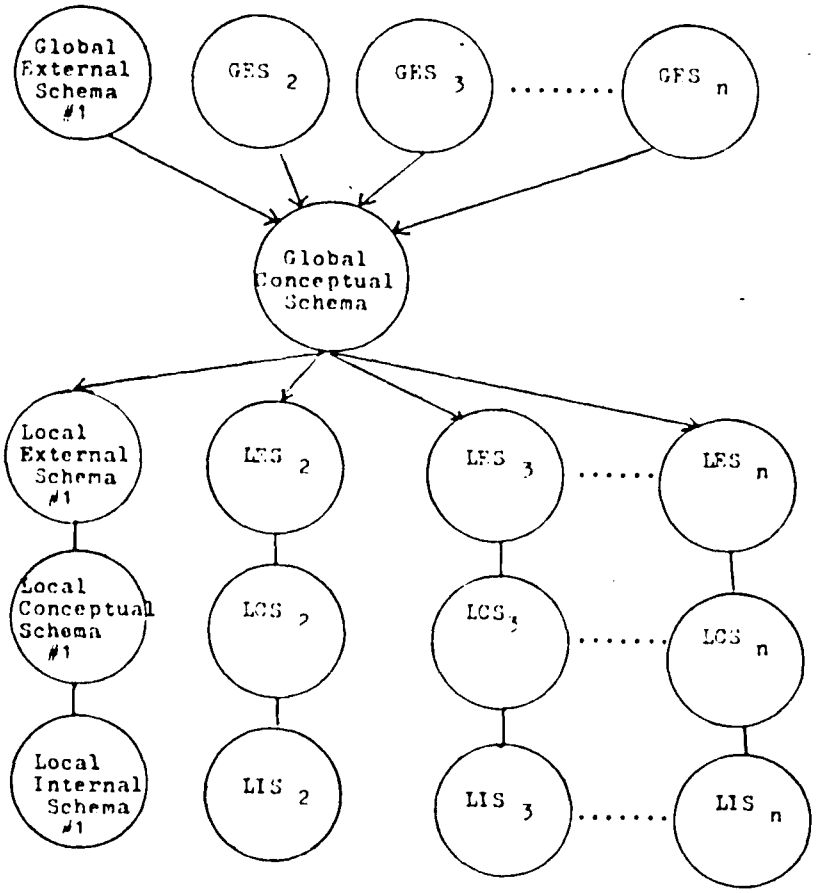


Figure 3.9

ANSI/SPARC Three Schema Approach

Applied to the DDBS

The global conceptual schema is the overall data base description. The many views of the data base held by the users can be mapped into diverse internal data models. [Wijssen 78] The development model with respect to this concept is presented in Figure 3.10.

As illustrated in Figure 3.10, the global conceptual model is the conceptual schema for the total information system. The development of the global conceptual model is the starting point of the design process. In addition to a starting point, it is an invaluable aid throughout the design process to keep the design in line with the DDBS's objectives and purpose.

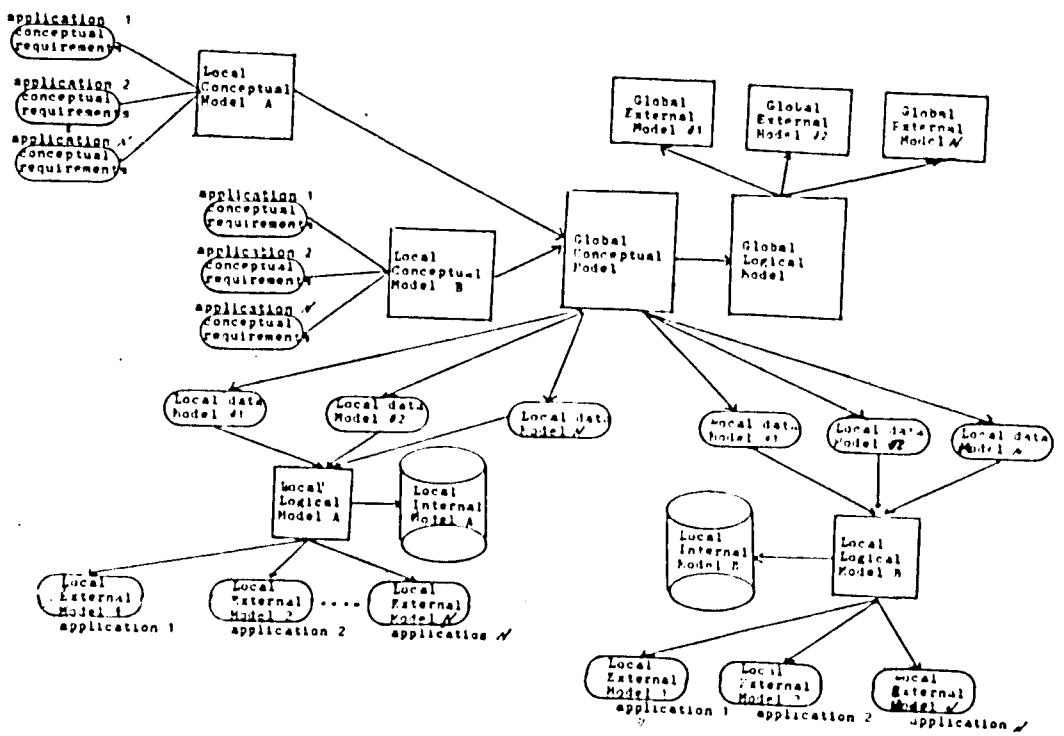


Figure 3.10

DDBS Model

3.6 Summary

In any data base design process, it is necessary to define what interactions are to occur in an information system. These interactions take place according to rules described in the conceptual schema. [Olissen 78] The process of deriving the description of an organizations information structure has two phases:

1) Gain an understanding of the information structure of an organization. This can be accomplished using analysis techniques such as structured analysis.
2) Provide a description of the understanding gained in the form of a specification.

Once the above is accomplished, one has a model of the conceptual schema. The design process can now proceed in a top-down fashion with the conceptual schema serving as the top level of the design. Chapter four will continue by offering a methodology to proceed in the process to design a DDBS. As noted in this, the third chapter, the process of designing a DDBS has inherent similarities to a DBMS. Naturally the design of a DDBS

must include additional considerations. These additional issues will be emphasized in chapter four.

4.0 Introduction

In chapter three, part one of a design consideration method for a DDBS focused on the steps needed to develop a framework in which a DDBS can be developed. The framework emphasized that the design of a DDBS must include many of the same considerations used when designing a centralized DBMS. Inherently, the two systems must be similar, however, the case of the DDBS includes extensions and additional steps to ensure a neatly meshing and logically sound system to satisfy users.

Once the top-level design steps of developing a global conceptual model have been accomplished, the task of designing lower level local models for the DDBS is possible. These lower level local models are the actual nodes that will interact in the system to create the distributed data base system. Figure 4.1 is an illustation of the DDBS model.

Two distinct levels are included in Figure 4.1. The upper portion of the diagram deals with the inputs and outouts needed in the development of a global conceptual model. The considerations involved in developing this upper portion were included in chapter three.

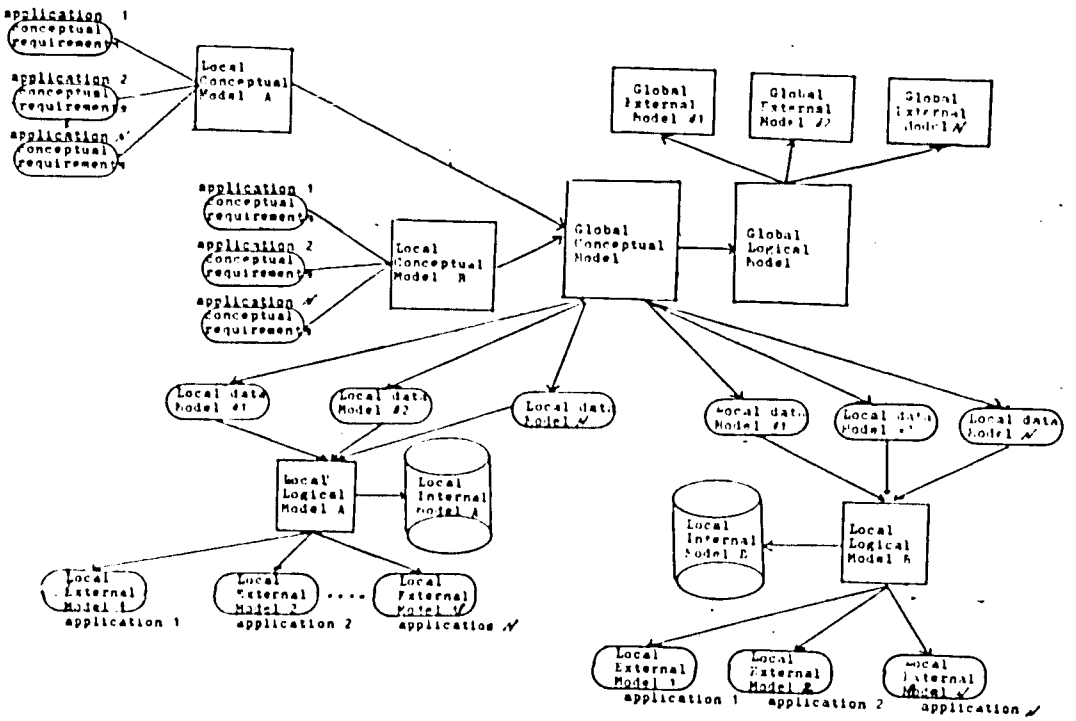


Figure 4.1

Design Model for a DDBS

Once attained, the global conceptual model can be refined to include local models. These local models are shown in the lower potion of Figure 4.1.

In essence, Figure 4.1 is the desired end product of the design process. This chapter will discuss the considerations needed to break apart the global conceptual model into a data base system which is not confined by the concept of centralization. The concept is similar to building and completing a puzzle. Once finished, all the thousands of pieces will be one picture. Even though the initial goal was to create one picture, the ultimate goal is to share the beauty of the picture with others. To do this, one must examine the picture very carefully in order that it may be divided in a way that each section of the puzzle is meaningful to the person with whom you wish to share it.

Even if a bit over simplified, the puzzle example is meaningful in understanding the goal of this chapter. The question is how does one break apart the global model into meaningful and yet interconnected portions? Two approaches will be discussed. First a qualitative approach and secondly a quantitative approach will be reviewed.

* The computer industry does not yet have the software which can do everything that one day might be desirable with distributed data bases. For the time being, the rule for systems analysts with such systems ought to be " Keep it simple ". [Martin 81]

4.1 A Qualitative Approach to Distributed Application Design

There are no hard and fast rules or formulas to decide what can and should be distributed. [Booth 81] There are, however, a few basic approches which can be applied to many situations in determining how to separate a global model.

The first step in logically breaking apart a single data model is to analyze application groups. In general, application groups will be: [Martin 81]

1) Reasonably self contained
2) Use small groups of record types
3) Have as little interaction with other application groups as possible, especially ones executed at a distance.

Examples of application groups are:

1) Local inventory control
2) Local labor analysis
3) Local accounts receivable

As a result of grouping applications into logical segments, a designer can also analyze the applications functions in terms of whether or not they should be considered for distribution. There are two possible reasons for considering the distribution of information processing and data base function: [Booth P1]

1) An application(s) cannot be implemented easily in a centralized structure. Size is the most common reason.
2) An application, if distributed, will provide advantages of cost, performance, survivability, response, flexibility, and/or control.

4.1.1 Case I - Required Distribution

It is reasonably easy to recognize when an application must be distributed because of its size. Any application which cannot be handled by a single processor while still having room to grow falls into this class. [Booth 81] Functional distribution is, in effect, forced in many large on-line systems. For example, no information processor built to date can directly handle several thousand terminals.Further, many large applications require high availability and reasonable response times which would be difficult, if not impossible, for a single processor to handle.

Most recently, very large applications have been implemented in decentralized configurations with data transfer via magnetic tape or disk pack. [Davenport 79, Booth 81] With the further advances in distributed processing, these applications are excellent candidates for implementation in a distributed system.

4.1.2 Case II - Optional Distribution

The majority of applications will fall into this category. Depending upon many factors, the decision whether to centralize or distribute an application group can be made. The design decision of whether or not to distribute, and further, what to distribute are extremly complex. A design technique developed at the Sloan School of Management, M.I.T [Rockart, Bullen, Levantor 78] for evaluating the pros and cons of centralization and decentralization can be expanded to include evaluation of distribution.

The first step in being able to use the evaluation technique is to identify logical end user nodes. This is accomplished by listing end user groups with their respective locations. For each user group, a list can be developed of all applications currently being used. The result is a set of logical application nodes. For each logical end user node, centralized or processing at the end user location is possible. [Martin 81] An example of this first step might be as follows:

End user groups and their locations

1) Accounting - city 1

   Accounting - city 2

Accounting - city n

Inventory Control - city 1

Inventory Control - city 2

End user groups and application groups utilized

Accounting - city 1

Payroll

Accounts Payable

Accounts Receivable

Inventory Control - city 1

Material Requirement Planning

Receiving

The second step is the development of a factor table for each application group. Figure 4.2 illustrates a hypothetical case for which a table was built. The columns are composed of the arguments for centralization, decentralization, and distribution. The rows are related to the factors to consider in the choice of centralization, decentralization, and distribution. [Martin 81]

	SYSTEM DEVELOPMENT				SYSTEM OPERATION						
	Application Selection	Application Development	Design of Data	Hardware/Software Selection	Input & Editing	Processing	Function Distribution	File	Data Base	Management Control	Strategic Planning
1)											
2)											
3)											
4)											
5)											
6)											
7)											
8)											
9)											

Key:
C= strong reason for centralization
c= weak reason for centralization
D= strong reason for decentralization
d= weak reason for decentralization
DS= strong reason for distribution
ds= weak reason for dist.

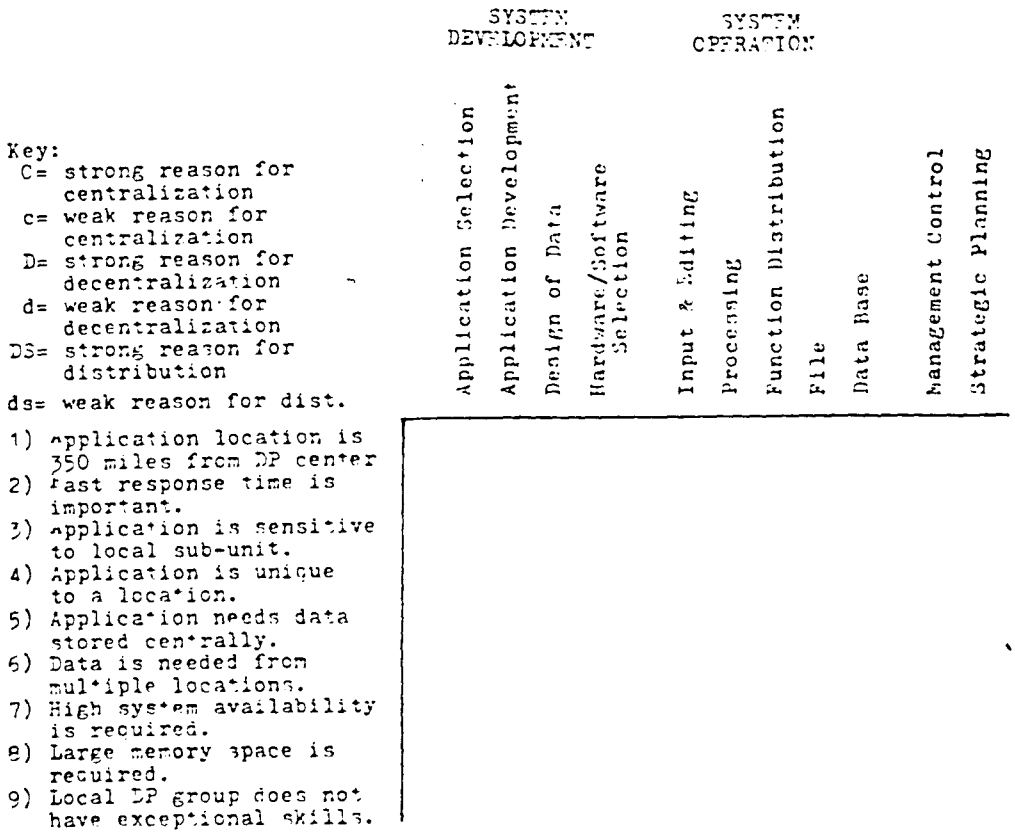
1) Application location is 350 miles from DP center
2) Fast response time is important.
3) Application is sensitive to local sub-unit.
4) Application is unique to a location.
5) Application needs data stored centrally.
6) Data is needed from multiple locations.
7) High system availability is required.
8) Large memory space is required.
9) Local DP group does not have exceptional skills.

***Note; A separate Factor table must be developed for each application group in an organization. The above example is intended to be nothing more than a hypothetical case, therefore the entries in the body are arbitrary. For further information regarding the development of these tables please refer to the reference sited.

Figure 4.2

Factor Table - Hypothetical Example

[Martin 81, pg.352]



***Note: A separate Factor table must be developed for each application group in an organization. The above example is intended to be nothing more than a hypothetical case, therefore the entries in the body are arbitrary. For further information regarding the development of these tables please refer to the reference sited.

Figure 4.2

Factor Table - Hypothetical Example

[Martin 81, pg.352]

The benefit of the development of a factor table is that the conflicting factors can be identified quite easily. The example given in Figure 4.2 must be reworked within the framework of a particular organization and application. Some factors may not be applicable to many organizations, others may not have been included. A complete study would identify and allow the systems analyst to analyze, what should be included.

4.2 Use of Factor Tables

Once factor tables are developed for each application group, further analysis can proceed. Each application group can be listed and then grouped into functional divisions. A hypothetical example of traditional functional divisions in an organization is illustrated in Figure 4.3 [Martin 81].

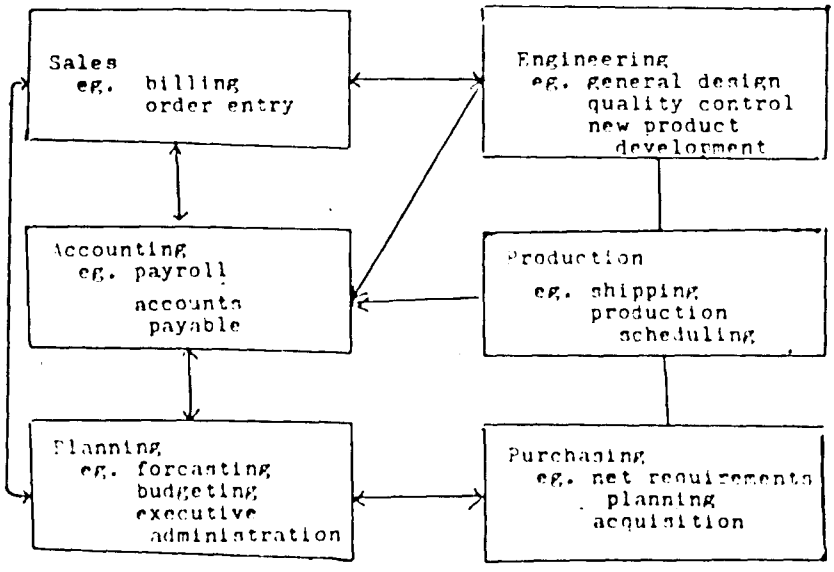


Figure 4.3

Applications Groups Functional Divisions

Figure 4.3 identifies the information flows among the functional areas. The functional areas include the application groups which may be associated with them. Naturally, each organization would have a unique Figure 4.3. The flow of information noted on Figure 4.3 is only an overall flow. Within each separate application group, there would also be data transfers among separate

applications. For example, closely associated application groups such as billing and acounts receivable, receiving, and accounts payable could share the same data; consequently, large numbers of internal data transfers take place.

The importance of Figure 4.3 is seen only if used in conjunction with the factor tables. The factor tables are essentially a list of potential reasons for selecting a distributed, decentralized, or centralized systen structure. The overall objective is to design a system which will most efficiently and effectively serve an organization. All the design decisions will be based on the potential benefits and costs associated with a particular design. For a single organization, there can be parts of an information system which should be centralized, others which can be decentralized, and further, those aplications which should be distributed. All these decisions are intimately tied to the type of organization in question, how many locations, what products it sells (if any) etc. This type of information can be reflected in a figure like Figure 4.3. The factor tables can be used with each of the application groups in Figure 4.3 to determine a preferred location for

1) its processing

2) the data it uses

3) the application development.

4.2.1 Summary of Usage of Factor Tables

The reasons for considering selection of a distributed, rather than centralized or decentralized system structure are numerous. By using factor tables, it can be seen that none of these reasons can be considered in isolation from the others. For example, a fast response time requirement may point toward distribution, however, management philosophy may favor centralization. Each aspect must therefore be considered and weighed in the making of the decisions of what, how, where, and when to distribute, decentralize, or centralize. Further, all decisions must be made on the basis of both management goals and philosophies as well as on the basis of technical and economic considerations.

4.3 A Quantitative Approach to Distributed Design

In sections 4.1 and 4.2, a qualitative approach to distributed application design was discussed. The qualitative approach was aimed at having the designer

assess the qualities of applications and associated data. Once completed, a general determination could be made as to what to distribute, decentralize, or leave centralized. In addition to the qualitative approach, James Martin [Martin 81] has developed a general quantitative method to be used with a qualitative method. The quantitative method is concerned with assessing quantities of interaction among nodes in a system. Both methods have weaknesses. Further, the two combined are not a panacea. The methods do however offer a starting framework which can be manipulated by organizations contemplating the design of a distributed data base system.

The quantitative design method is aimed at arranging the location of processing and data so that the traffic and interaction among the nodes in a system is low.

4.3.1 Logical Data Network

The first step will be identification of end user locations, their assorted application groups, and required data. The data used by applications are organized into data structures. These data structures can be either a file or a data base. A file is generally designed for one application or possibly a group of closely related

applications. A data base, in contrast, is a collection of interrelated data which are independent of application programs and which have been structured so that it can serve as many applications as is useful. [Martin 76, Martin 81, Artre 80, Ulman 78]

Once the above mentioned items have been determined, a logical data structure network can be constructed. Figure 4.4 shows a hypothetical example of a network diagram. [Martin 81]

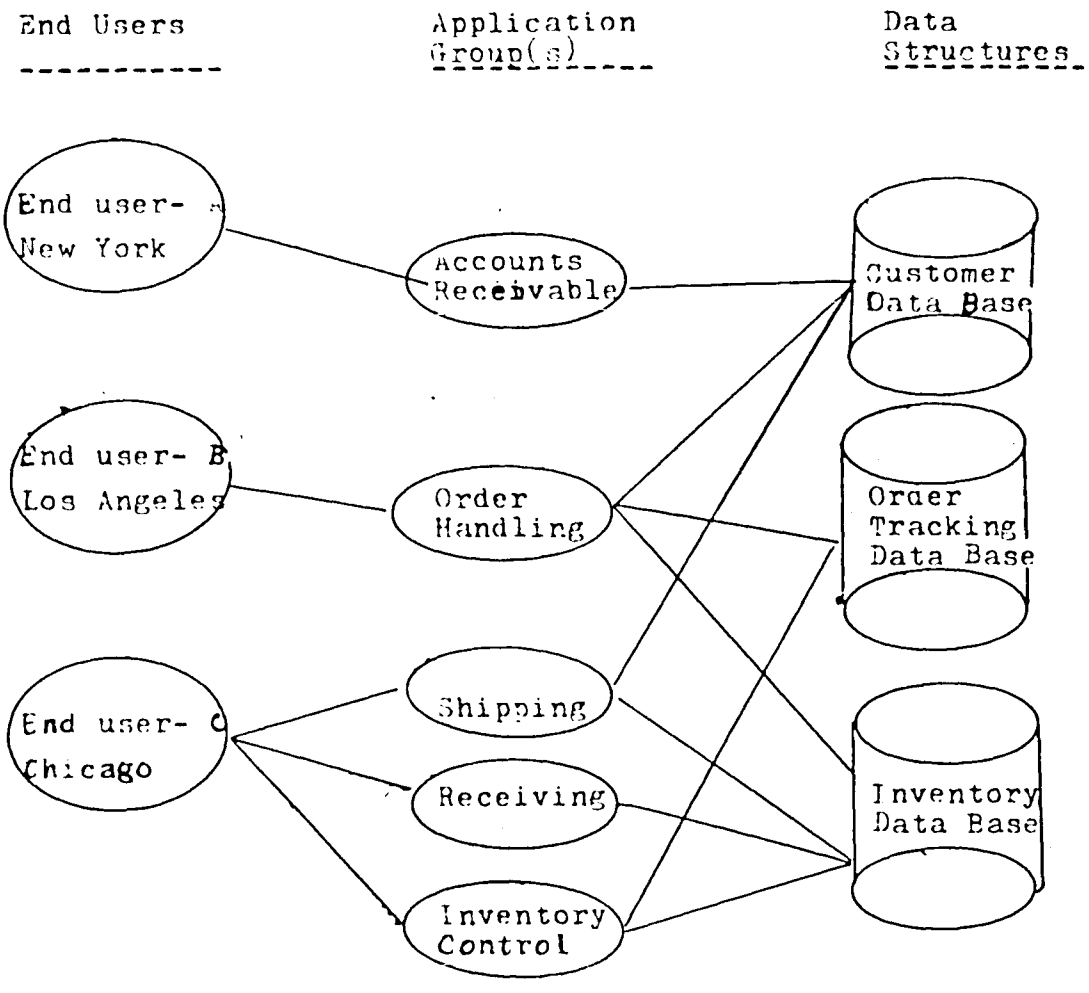


Figure 4.4

Logical Data Structures Network

The logical data structure network is useful in the preliminary design phases. The primary reason to use such a diagram is to identify which end users use which data. The diagram also illustrates the basic fact that application programs frequently require data from two or

more data bases. Data bases, in turn, frequently supply data to two or more application programs. [Martin 81] Once the interdependencies among application systems and data structures are seen, the designer can begin to postulate where to physically locate data structures and/or files. One of the most complex problems associated with the design of a Distributed Data System is the determination of where to locate data. The problem is commonly known as the "file allocation problem."

4.3.2. The File Allocation Problem

The file allocation problem was first investigated by W. Chu [Chu 73] Since Chu's model was developed, many mathematical models have been developed to determine:

1) The number of copies of each data base file to store in the network
2) The location of these copies
3) Which copy of a file to use when processing a given query.

The objective is to minimize the amount of data transferred in a network in order to satisfy a given set of queries subject to certain processing constants. [Elam and Fisher 79]

The computational models available today are, in general, highly complex and very subjective in nature. No model yet realistically represents the major design issues. [Rothnie and Goodman 77, Elam and Fisher 79] Recent works by Chen and Akoka (81), Coffman, Gelenbe, and Plateau (81), and Elam (79) are trying to move closer to a generally applicable model to solve this problem.

The key point is that the designer of a DDBS must be cognizant of the data quantities in the system. The major objective must be to minimize data transfer. Minimization of data transfer can be acomplished by either replicating data or partitioning data. Both techniques have costs and benefits associated with them. These must be analyzed in the light of a particular situation. Constraints to both include: concurrence of update, security, data integrity, and recovery problems. The overall optimization criteria should be based on a mixture of costs, performance, benefits to the users, and responsiveness.

4.3.3 Further Alternatives

To complicate matters further, a DDBS designer must consider not only a static case of assigning data structures to particular nodes in a system but the cases when application programs and/or users need access to data at another site. The problem is how the program and data should be brought together. There are three basic methods currently available to accomplish this: [Booth 76,81]

1) Move the program or process to the data;program migration. A process is a program in an execution state.
2) Move the data to the process; data migration
3) Some combination of these two methods ; split processing

Migration can be either a dynamic process or a reasonably static one. The dynamic case would include situations where movement can occur at the time a remote access request is made by either an application program or user. The static case would include situations where data and/or programs are moved at a time before they are needed in preparation for an expected request. These are the

proactive approach and the reactive approach respectively.

4.3.3.1 Program Migration

There is a possibility that in a DDBS system a program may require data stored at another location. A solution to this situation may be to move a program to the site where the data required is stored. This would be a practical resolution particularly in a case where the program size is small compared to the data required. The actual movement of the program could be user initiated or software initiated.

Figure 4.5 is an example of such a situation. User A runs a program which requires data in location B. Program A is sent to location B to be executed and when completed, the output and program A will be returned from location B [Booth 81].

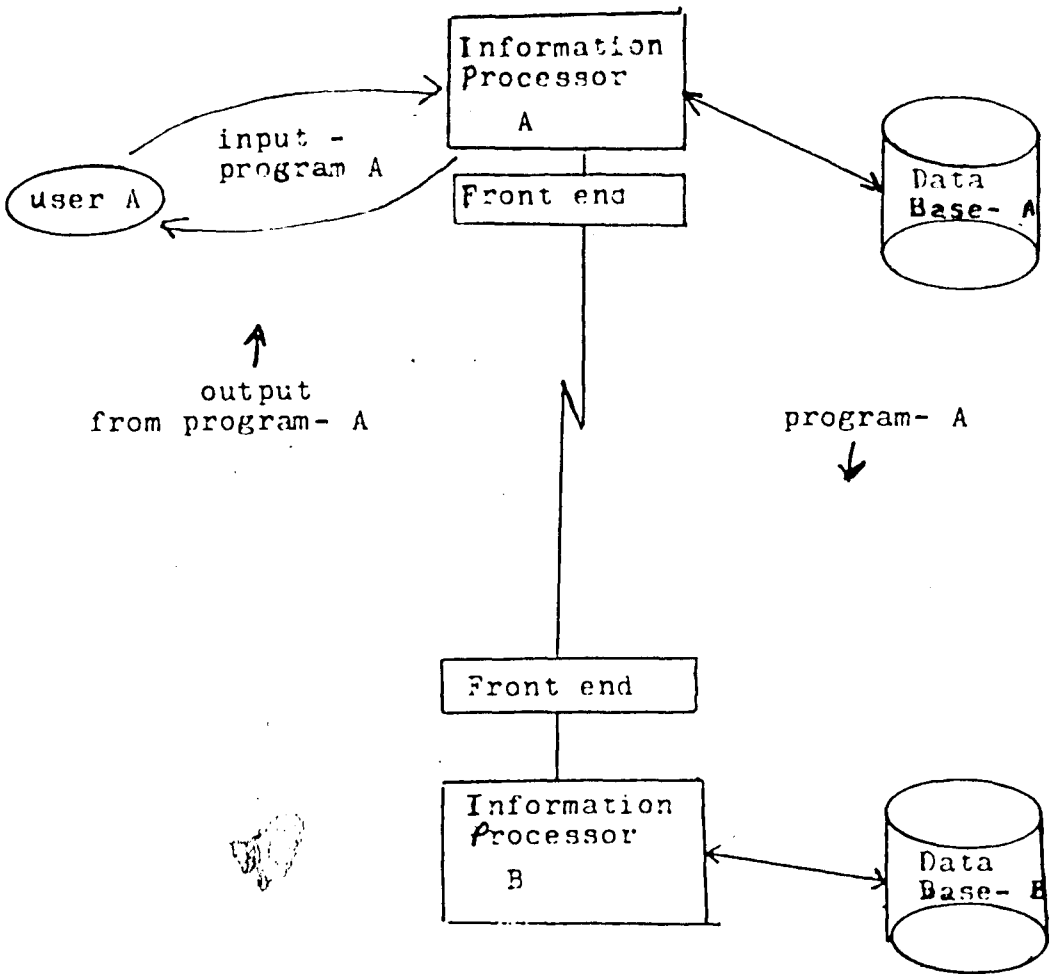


Figure 4.5

Program Migration

The process becomes far more complicated if more than 2 nodes exist in the system. If more than 2 nodes exist, either the user or application developer must know the exact location of data needed and how to reach it, or if a global directory system is used, each information processor can know where all data base elements are.

4.3.3.2 Data Migration

If only a small portion of a data base stored at another location is required, it would make sense to move the data to where it is needed. This approach would be most applicable to time sharing or transaction processing systems. [Booth 81]

As in the case of program migration, the movement can be either software controlled or application program controlled. The software controlled case makes the movements transparent to the user, whereas the applications program could make the movement totally visible to the users.

Figure 4.6 illustrates data movement controlled by sortware. The executing process requests data. The data requested is not found at the local site. The global

directory determines where the needed data are stored and initiates a remote access request. If the data needed are not being used by another application that denies access, the requested data can be retrieved and sent to the location where it is needed. This may sound easy, however, the software required to perform such a process is available only for identical computers using identical data base management systems [Booth 81].

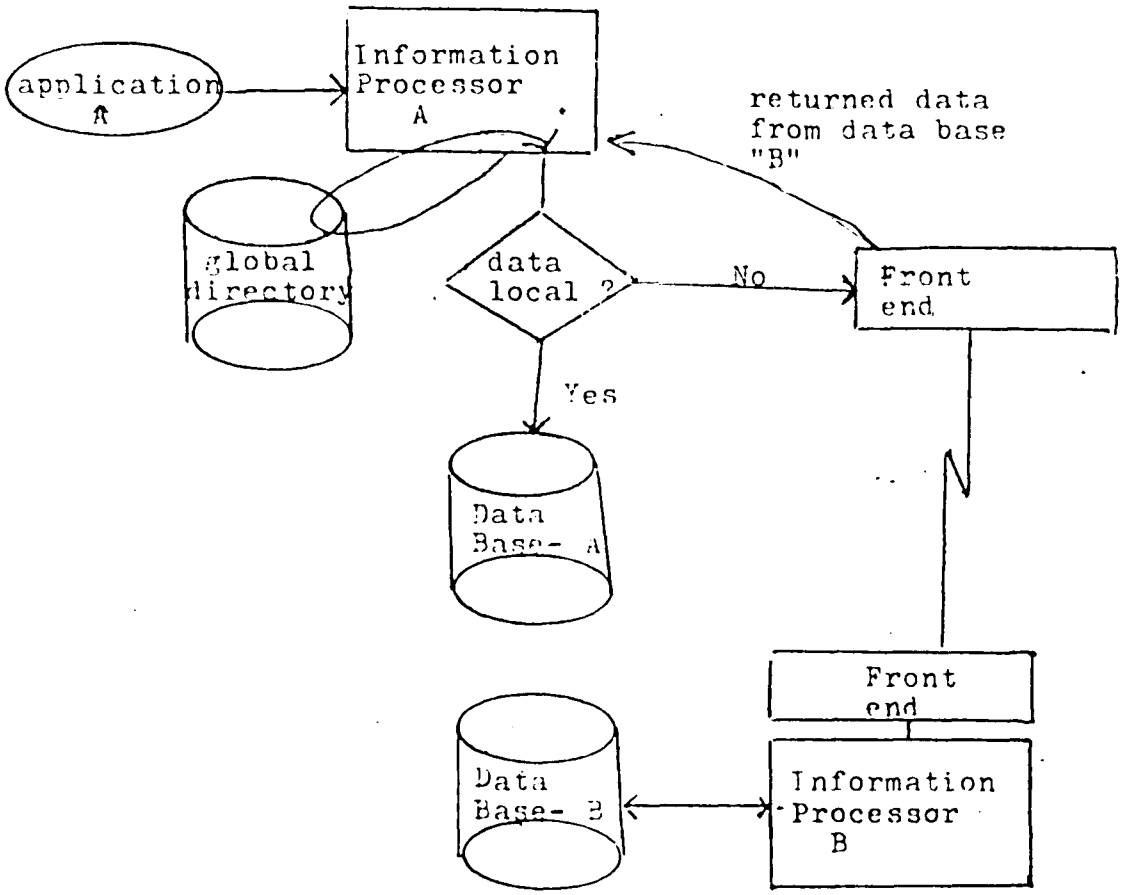


Figure 4.6

Data Migration

4.3.3.3 Split Processing

There probably will exist a few complex situations where there is a need to access data or programs stored at more than two locations. One example may be the situation in which a company requires a periodic report necessitating consolidated input from all partitions in a DDBS. This type of situation is not easily handled.

Unfortunately, there is no specific solution to such a problem other than using a combination of both data migration and program migration. Again, the objective is to minimize transfer of data and/or programs.

4.3.4 Program/Data Structure Graph

In section 4.3.3 the need to examine application programs was discussed. One application can require interaction with multiple data structures as well as need interaction with other application programs located at different nodes in the system.

In order to examine the effect of application distribution, Martin suggested the division of application programs into four classes.

1) Class 0 ---> a class 0 program has both the application it serves and all the data it requires in its own location.

2) Class 1 ---> a class 1 program is located in a location diferent from the application it serves but has all the required data at its own location. This is the conventional teleprocessing case.

3) Class 2 ---> a class 2 program has the same location as the application it serves, however, some or all of the required data are in a remote location.

4) Class 3 ---> a class 3 program would be the rare case of having programs which are distant from both the application and some of the data they use. [Martin 81]

These classes are summarized in Figure 4.7.

	Application node	Data Structure
Class 0	same location	same location
Class 1	different location	same location
Class 2	same location	some or all
Class 3	different location	are in a different location

Figure 4.7

Summary of Program Classes

Class 0 programs are the most desirable to have. Class 3 programs should be avoided. Programs not of class 0 must be resolved by using of one of the following:

1) Program Migration
2) Data Migration
3) Split Processing
4) Replication of Data Structure
5) Replication of Application Programs

Network charts can be developed to illustrate what application programs require what data. They enable the designer to illustrate what classes particular application programs are. Figure 4.8 illustrates such a Network chart. This Figure is based on Figure 4.4, the logical data structure network diagram [Martin 81].

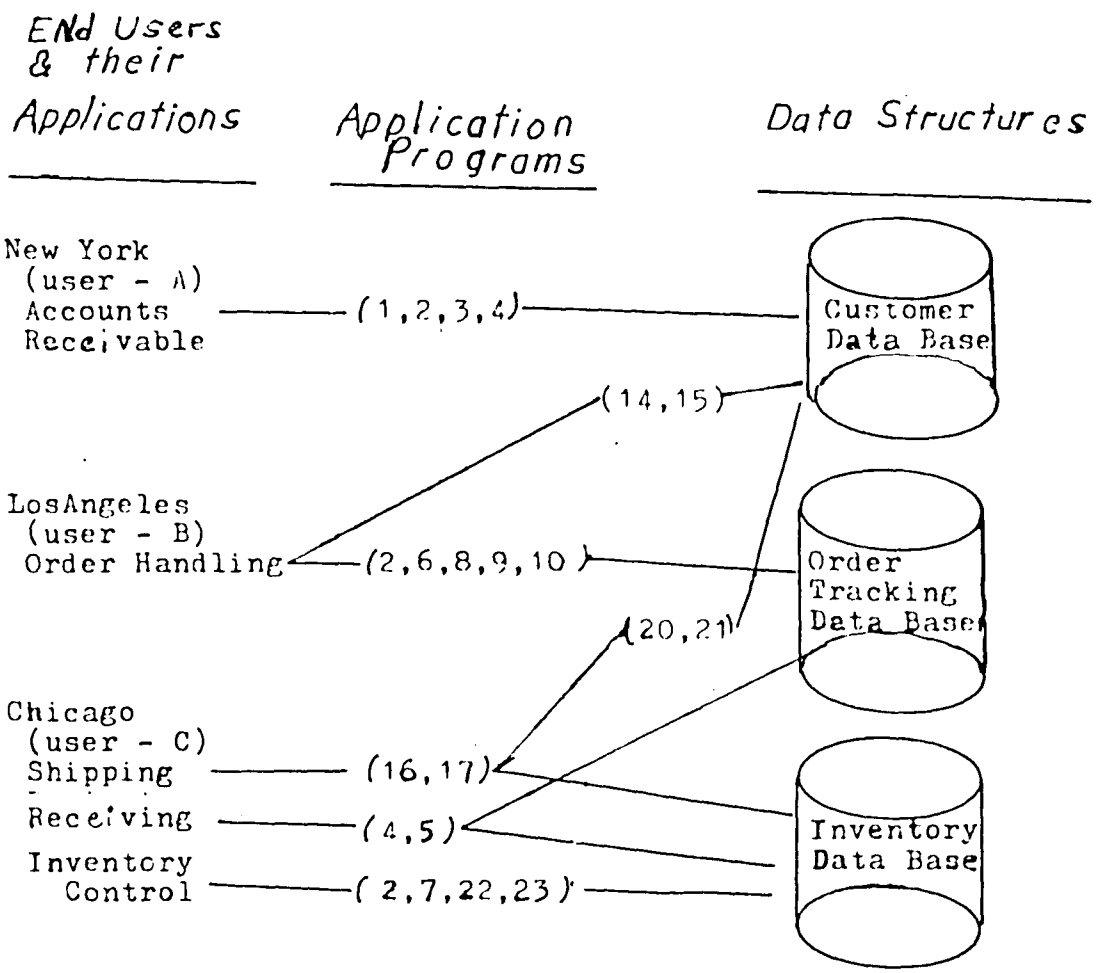


Figure 4.8

Program/Data Structure Network

From Figure 4.8, a designer can count the number of application programs in each class. For example, application 14 and 15 would be classified as Class 1 because they are distant from the application group they serve but all data are in the same location. Applications 4 and 5 would be classified class 2. Numbers 20 and 21 would be class 3. Figure 4.8 is meant to be used by the designer to minimized the occurrences of class 2 and 3 programs.

The designer must look closely at the costs of replicating data, partitioning, transmission costs and traffic when attempting to design an efficient and effective system model. Many variations may have to be attempted before a suitable configuration is found.

4.4 Summary

This chapter has presented and discussed a number of techniques which can be utilized in pursuing a Distributed Data base design. Two approaches were discussed. A qualatative approach looked at the nature of the system parts which are used to help determine which applications might be good candidates for distribution, decentralization, or centralization. A quantitative

approach was discussed to illustrate further considerations of which a designer must be aware. The quantitative approach is aimed at having a designer look at the quantities of data and program transfers needed in a system. The object should be to minimize transfers, yet still design a system which is optimal to the organization it serves.

Both techniques constitute only a beginning for designing a distributed data base system. The complexity of a DDBS system and the recency of its development ensures any designer challenges as well as many opportunities to be innovative. There are no rules, as such, that can applied to all possible design situations other than keep it simple.

5.0 Introduction

In the preceeding chapters, several major facets of designing a Distributed Data Base System were addressed. The objective of this research effort was to accomplish the following:

1) Define precisely what a Distributed Data Base is in terms its component parts.

2) Define what alternative structures a Distributed Data Base System can take.

3) Describe the relevance of design methodologies used to generate centralized Data Base Management Systems to designing a Distributed Data Base Management System.

4) Describe a general method which can be used to begin designing a Distributed Data Base System.

In this chapter the significance of this work is summarized. Further, several avenues of future research are identified.

5.1 Significance

Current literature on Distributed Data Bases reflects the immaturity of this technology in that there is no single publication, to this authors knowledge, that deals with the subject coherently and in depth. Most studies available on the topic of DDBS design and technology are limited in scope. These studies are aimed at examining various aspects of the topic, however, fail to provide a comprehensive overview of what alternatives are available in DDBS's , how these alternatives are related to each other, and how any of the alternatives available are attainable.

The subject matter covered in Chapters one through four provides a useful starting point for system designers interested in achieving the benifits associated with a Distributed Data Base System. As distributed processing systems emerge from their infancy, more detailed and standardized methods will be developed to design such

systems.

5.2 Future Research

The completion of the research reported here has given insight into several related areas which might be fruitfully investigated.

5.2.1 Data Communication Networks

The data communication networks upon which DDBS's are based give rise to many technological and design issues. Issues of particular concern in the area of DDBS are the architecture and interconnection schemes of a network, and the value of mathmatical modelling and simulation in the design of DDBS network models.

5.2.2 File Allocation

The questions of how files and processes should be allocated to the nodes in a network and where certain critical control functions should be performed are

extremely important. Several attempts have been made to derive a mathmatical model that could be used in allocation of resources in a network, however, none to date have been able to account for the great number of variations possible in a DDBS environment.

5.2.3 Data Base Integrity

In Distributed Data Base Systems, serious problems are associated with maintaining data base integrity when the same data are stored at different sites. To date, no method has been sucessful in addressing this problem.

Associated with integrity is the problem of reliability. A systems reliability relates to its ability to serve its users in spite of component failures. The problem is how to restore a failed node in a DDBS so that it is consistent with the rest of the system.

5.2.4 Summary

The areas of possible future research are only a fraction of the areas that need attention in Distributed Data Base Systems. Those noted are significant to this author. Practical solutions would enable system designers to design more pragmatic Distributed Data Base Systems.

5.3 Concluding Remarks

The construction of a Distributed Data Base is a difficult task. In this thesis, basic design issues have been addressed. Through the illustrations and text, a system designer has been provided with a means to evaluate whether a DDBS is a viable alternative to serve user's information processing needs. Further, the design process could be started. Naturally, much further information is required to complete the design given specific organizational circumstances. Many challenges remain as distributed systems continue to evolve.

BIBLIOGRAPHY

[Adiba 79]

Adiba,M.,"Issues in Distributed Data Base Management Systems:A Technical Overview," Proceeding 4th International Conference on Very Large Data Bases, September 1979.

[ANSI 76]

"ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report,75-02-08," FDT-Bulletin of the ACM SIGMOD, June 1976, pp. 97-137.

[Atre 80]

Atre,S., Data Base, Structured Techniques for Design, Performance, and Management, John Wiley _Sons, New York, New York, 1980, pp. 124-191.

[Baker 80]

Baker,C.T., "Logical Distribution of App-

lications and Data," IBM Systems Journal, Vol.19,No.2, 1980, pp.171-191.

[Booth 81]

Booth,G.M., Honeywell Information Systems, The Distributed System Environment-Some Practical Approaches, McGraw Hill, New York, New York, 1981.

[Borkin 80]

Borkin,S.A., Data Models: A Semantic Approach For Data Base Systems, MIT Press, Massachusetts, 1980.

[Buchanan 80]

Buchanan,J.P. and Linowes,P.G., "Making Distributed Data Processing Work," Harvard Business Review, Vol.58.,No.5, September-October 1980, pp. 143-161.

[Buchanan 80]

Buchanan,J.R.. and Linowes,R.G., "Understanding Distributed Data Processing," Harvard Business Review, Vol.58,No4, July-August 1980, pp. 143-153.

[Champine 77]

Champine,G.A., Six Approaches to Distributed Data Bases," Datamation, Vol.23,No.5, May 1977, pp. 69-72.

[Champine 79]

Champine,G.A., Application Partitioning and Data Base Machines," INFOTECH State of the Art Report, Vol.1, pp. 135-136.

[Chen 81]

Chen,P., and Akiba,J., "Optimal Design of Distributed Information Systems," IEEE Transactions on Computers, Vol.C-29,No.12, December 1980, pp. 1068-1079.

[Chu 73]

Chu,W.W., "Optimal File Allocation in a Computer Network," Computer-Communication Networks, Prentice-Hall, New Jersey, 1973, pp. 82-94.

[Codd 70]

Codd, E.F., "A Relational Model for Large Shared Data Banks," Communications of the

ACM, 13:6, 1970, pp. 377-387.

[Coffman 81]

Coffman,E.,Jr., Gelenke,E.,and Plateau, B., "Optimization of the Number of Copies in a Distributed Data Base," IEEE Transactions on Software Engineering, Vol.SE-7 1, January 1981, pp. 78-84.

[Cypser 78]

Cypser,R.J., Communications Architectures for Distributed Systems, Addison-Wesley Publishing Company, California, 1978.

[Date 79]

Date,C.J., "Locking and Recovery in a Data Base System: An Application Programming Tutorial," Proceedings of the 5th International Conference on Very Large Data Bases, Rio de Janeiro, October 1979, pp. 1-15.

[Davenport 79]

Davenport, R.A., "Design of Distributed Data Base System," INFOTECH State of the Art Report, Vol.2, pp. 89-114.

[Davenport 77]

Davenport,R.A., and Palmer,I., "Distributed Databases," INFOTECH State of the Art Report, On-Line Databases, C.H. White,editor, 1977.

[Davenport 81]

"Design of Distributed Data Base System," The Computer Journal, Vol.24 1, February 1981, pp. 31-42.

[DeMarco 78]

DeMarco,T., Structured Analysis and System Specification, Prentice-Hall, Inc., New Jersey, 1978.

[Denning 76]

Denning,P., "A Hard Look at Structured Programming," INFOTECH State of the Art Report-Structured Programing, D.Bates, editor, 1976.

[Eckhouse 78]

Eckhouse, R.H., "Issues in Distributed Procesing: An Overview of Two Workshops," COMPUTER, January 1978, pp. 22-26.

[Edelberg 78]

Edelberg, M., "Issues of Distributed Data Base Systems," Proceedings of the 4th International Conference on Very Large Data Bases, 1978, pp. 159-163.

[Elan 79]

Elan,J.J., and Fisher, ., "Use of Mathematical Models in Distributed Data Base Design," INFOTECH State of the Art Report, 1979, pp. 117-125.

[Elan 78]

Elan,J.J., "A Model for Distributing a Data Base," Working Paper, Department of Decision Sciences, University of Pennsylvania, Philadelphia, 1978.

[Enslow 78]

Enslow, P.H., "What is a Distributed Data Processing System?," COMPUTER, January 1978, pp.13-21.

[Freeman 78]

Freeman,P., "Design Fundamentals," INFOTECH

State of the Are Report, Structured Analysis and Design, J.Hosier, editor, 1978.

[Germano 80]

Germano.F., "Automatic Transaction Decomposition in a Distributed CODASYL Prototype System," Working Paper, Department of Decision Sciences, University of Pennsylvania, Philadelphia, May 1980, pp. 1-19.

[Green 79]

Green, P.E., "An Introduction to Network Architectures and Protocols," IBM Systems Journal, Vol.18,No.2, 1979, pp. 202-221.

[Joyce 78]

Joyce,J., "Principles of Data Base Management in a Distributed System," Computer Communications, IEEE Transactions on Computers, New York, New York, 1978.

[Langefors 75]

Langefors,B., and Sundgren,B., Information Systems Architectures, Petrocelli/charter, New York, New York, 1975, pp. 201-225 and

253-323.

[Long 79]

Long,L., Data Processing Documentation and Procedures Manual, Reston Publishing Company, Inc., Virginia, 1979, pp. 15-19.

[Lorin 79]

Lorin, H., "Distributed Processing:An Assessment," IBM Systems Journal, Vol.18, No.4, 1978, pp.582-602.

[Lorin 81]

Lorin,H., "DDP: How to Fail," Datamation, February 1981, pp.60-64.

[Martin 76]

Martin, J., Telecommunications and the Computer, Second Edition, Prentice-Hall, New Jersey, 1976.

[Martin 77]

Martin,J., Computer Data-Base Organization, Second Edition, Prentice-Hall, New Jersey, 1977, pp.22-81.

[Martin 81]

Martin, J., Design and Strategy for Distributed Processing, First Edition, Prentice-Hall, New Jersy, 1981, pp. 224-375.

[Mohan 79]

Mohan,C., and Yeh,R ., "Distributed Data Base System-A Framework for Data Base Design," INFOTECH State of the Art Report, Vol.2, 1979, pp.239-256.

[Nijssen 78]

Nijssen,G.M., "The Next Five Years in Data Base Technology," INFOTECH State of the Art Report, Data Base Technology, 1978.

[Orr 77]

Orr,K., Structured Systems Development, Yourdon Press, New York, New York, 1977.

[Patrick 80]

Patrick,R.L., Application Design Handbook for Distributed Systems, CBI Publishing Company, Inc., Massachusetts, 1980, pp.3-26.

[Potier 80]

Potier,D., and Leblanc,P., "Analysis of Locking Policies in Data Base Management Systems," Communications of the ACM, Vol.23, No.10, October 1980, pp.584-592.

[Pouzin 78]

Pouzin,L., "A Tutorial on Protocols," Proceedings of the IEEE, Vol.66,No.11, November 1978, pp.1346-1368.

[Ries 79]

Ries,D., and Stonebreaker,M.R., "Locking Granularity Revisited," ACM Transactions on Data Base Systems, Vol.4,No.2, June 1979, pp.210-221.

[Rothnie 77]

Rothnie,J.B., and Goodman,N., A Survey of Research and Development in Distributed Data Base Management, Proceedins of the 4th International Conference on Very Large Data Bases, Tokyo, Japan, 1977.

[Rothnie 80]

Rothnie,J.B.,Bernstein,P.A.,Fox,S.,Goodman, N.,Hammer,M.,Landers,T.A.,Reeves,C.,Shipman, D.W.,and Wong,E., "Introduction to a System for Distributed Data Bases (SSD-1)," ACM Transactions on Data Base Systems, Vol.5, No.1, March 1980, pp.1-17.

[Salter 77]

Salter,J.A.M.,Danielson,B.I.,and Sandvald, A.E., "The Implementation of On-Line Data Base Systems on Nine Machines," INFOTECH State of the Art Report, C.H.White,editor, 1977.

[Scherr 78]

Scherr, A.L., "Distributed Data Processing," IBM Systems Journal, Vol.17,No.4, 1978, pp.324-343.

[Semprevivo 76]

Semprevivo,P.C., System Analysis:Definition, Process, and Design, SRA Inc, Chicago, 1976.

[Severino 77]

Severino, E.F., "Data Bases and Distributed

Processing," Computer Decisions, March 1977, pp.40-42.

[Steel 78]

Steel,T.B., "The Current ANSI/X3/SPARC DBMS Proposals," INFOTECH State of the Art Report, Data Base Technology, 1978.

[Teory 78]

Teory, T.J. and Fry, J.P., "Logical Data Base Design: A Practical Approach," INFOTECH State of the Art Report, Data Base Technology, 1978.

[Ullman 80]

Ullman,J.D., "Principles of Data Base Systems," Computer Science Press, New York, New York, 1980, pp.6-22.

[Weber 79]

Weber,H.,Baum,D.,Popescu-Zeletin,R.,"The Design of a Distributed Data Management System on a Heterogeneous Computer Networks," INFOTECH State of the Art Report, Vol.2, 1979, pp.281-311.

[Whitby 79]

Whitby-Strevans,C., "Distributed Data Base Research in Europe; A Personal View," INFO TECH State of the Art Report, Vol.2, November 1979, pp.313-326.

[White 79]

White,C.H., "Distributed Data Bases," INFO-TECH State of the Art Report, Vol.1, Nov November 1979.

[Wood 80]

Wood,C.,Fernandez,E.B.,and Summers,R.C., "Database Security: Requirements,Policies, and Models," IBM Systems Journal, Vol.19, No.2, 1980, pp.229-251.

[Yen 78]

Yen,R.,Chang,P.,and Mohan,C., "A Multilevel Approach to Data Base Design," Proceedings COMPSAC November 1978.

[Yourdon 79]

Yourdon,E.N., editor, Classics in Software Engineering, Yourdon Press, New York, 1979.

[Ziegler 79]

Ziegler,K., "A Distributed Information System Study," IBM Systems Journal, Vol.18, No.3, 1979, pp.374-401.

[Zimmerman 80]

Zimmerman,H., "OSI Reference Model-The ISO Model for Architecture for Open Systems Interconnection," IEEE Transactions on Communications, Vol.28,No.4, April 1980, pp.425-431.

VITA

Jan Gary Kroc was born April 22,1954. He received a B.S. degree in Economics from Lafayette College in 1980. He has served as a Research Assistant in the Department of Industrial Engineering from May 1980 through May 1981. He is currently employed by the International Business Machines Corporation as a systems analyst. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), the American Institute of Industrial Engineers (AIIE), and the Association for Computing Machinery (ACM).