Query Processing in a Distributed Data Base

by

Leou, Chinchin Leou

B.S., National Chung Hsing University

Taipei, Taiwan, R.O.C. 1976

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1984

Approved by:

[illegible]

Major professor

TABLE OF CONTENT

LISTS OF FIGURES AND TABLES

CHAPTER 1

INTRODUCTION

The growth of distributed data base applications is constantly increasing. It is expected that many future data base implementations will be of the distributed type. This is due to the tremendous boom in mini and micro computers as well as the fast development of computer network technology.

Today, more and more companies can afford to have low-cost, sophisticated mini or micro computers. The branch office and the subsidiary company can work with their own computer systems independently of one another. However, the increasing staff-cost produces the need and desire to build a hardware intensive system. This type of system starts with a combination of powerful small computers and the traditional mainframe. Consequently, the arrangement for geographically separated systems that are able to intercommunicate has become a realistic solution.

Compared to a centralized data base, the distributed data base provides certain advantages such as, a) increasing data reliability and resilience, b) faster localized access of data, and c) flexible performance. Problems and difficulties may also arise from the introduction of distributed data base systems; the time delay caused by the communication between the geographically distributed data resources, the consistency of the redundant data, and the security and privacy control are all problems that should be carefully considered. In particular, the efficiency of a distributed data base system may highly influence the user's interest and acceptance. Query processing is one of those factors which determines this efficiency.

Query processing in a distributed data base corresponds to the translation of a user's request into a sequence of instructions which will retrieve data stored in various locations. Recent studies on distributed data bases have shown that the development of data retrieval strategies for queries is deeply related to the goal of obtaining an efficient performance. This report will attempt to give a comprehensive discussion of the various problems related to query processing in a distributed data base and a feasible solution to the problem of optimization.

The remainder of this paper is organized as follows: chapter 2 presents the definition of a distributed data base system, its classification and considerations. Chapter 3 briefly presents the architecture and structure of a network system and its considerations related to the query processing. Chapter 4 reviews the literature with a categorization of how query analysis, data transfer, data dictionary, and data distribution impact query processing. Chapter 5 outlines the approach of optimization. Both Unger et al. and Hevner & Yao's strategies are presented, and a comparison and evaluation is also given. Finally, Chapter 6 indicates the conclusion and recommendations.

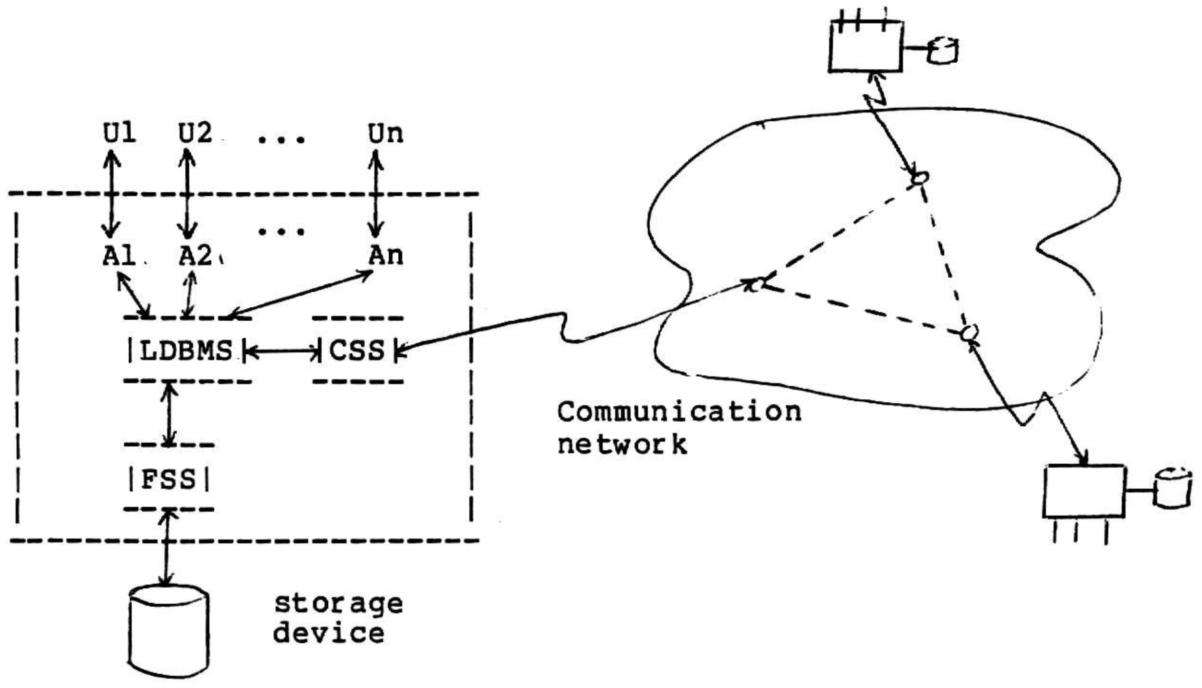
CHAPTER 2

DISTRIBUTED DATA BASE

2.1 Definition of distributed data base

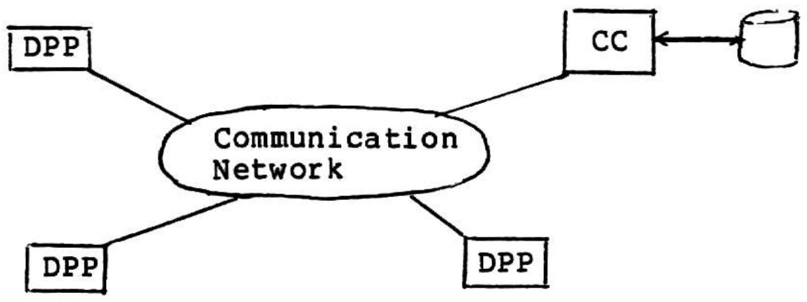
A Distributed Data Base (DDB) can be defined as a logically integrated data base which is distributed over several physically distinct but linked sites. In other words, the Distributed Data Base Management System (DDBMS) is a software system which maintains a data base where the data is distributed among subsets (called local data bases), stored at various sites, and geographically dispersed but interconnected by a communication network. These subsets can be either overlapped or disjointed. Figure 2.1 shows one of these local data bases, also called nodes in the communication network. User programs (applications) interact with a module called Local Data Base Management System (LDBMS) which implements all of the functions for accessing remote data via the communication subsystem and local data via the local file subsystem. The network provides the configuration of computer systems and communication facilities in which data is stored, DBMS is operated and users access data.

The system shown in figure 2.1 can be viewed as a fully distributed environment, i.e., the processing power, data base, and control are all distributed. However, this approach is still in an experimental stage in university and

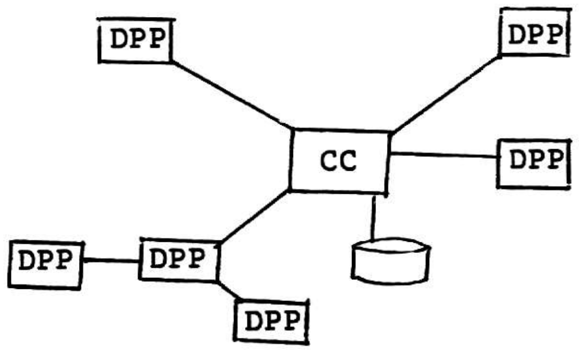


LDBMS	-	Local Data Base Management System
CSS	-	Communication Sub-system
FSS	-	File Sub-system
Ui	-	User i
Ai	-	Application i

Figure 2.1 Operation environment for Distributed Data Base System.



(a) Centralized data base with decentralized data processing through communication networks



(b) Centralized data base with decentralized data processing through hierarchical control.

DPP	=	Decentralized Processing Power
CC	=	Central Computer with system control functions

Figure 2.2 Alternative architecture of distributed system.

industrial research environments. In the real world, most applications of distributed data bases seem to stay in a partially distributed environment as showned in figure 2.2 (a) & (b). The system with distributed processing power, in the form of intelligent terminals or minicomputers, is connected either in a star structure or through the communication network to a single large central computer with its own data base.

2.2 Considerations in DDB

The distributed data base provides certain advantages over the centralized data base, such as an increase in data reliability and resilience, faster localized access of some data, and flexible performance. A distributed approach allows rapid reconfiguration since the modular satellite systems can be easily moved to another location. Any subsystem can be replaced or removed without disturbing the remainder of the system. Some problems, which are either unique to a distributed data base system or else more difficult than they are in a centralized environment, also arise from the introduction of DDBS; therefore, some considerations must be examined first. For successful operation, a distributed data base should possess attributes as described:

(1) User transparency

From the user's point of view, DDB works just like a single integrated data base. The user does not need to know the location of specific files, relations or fragments. Meanwhile, data can be moved from time to time without directly affecting the user.

(2) local autonomy

Some degree of local autonomy, i.e. users' participation, is desirable, depending upon the nature of DDB. Where the data base is essentially a union of autonomous groups, local autonomy is a paramount feature. Where the data base is a single corporate entity, this feature plays a lesser part.

(3) Data distribution

With the possibility of data redundancy, data may be placed physically at one or more locations. The locations of data are highly related to the data retrieval strategies. More details about this topic will be discussed in chapter 4.

(4) Efficient operation

The DDB should be able to achieve much higher efficiency in terms of utilization of machine and man power than large multi-programming, multi-tasking centralized systems, since the small computers within the distributed system are normally dedicated.

(5) Data base reliability and integrity

Duplicated data provides greater reliability of the data base in case of machine failure. However, an error possibility arises when data is stored at two or more physically remote sites. It is apparent that redundancy will have major impact on consistency, integrity and performance. More attention must be given in order to keep the diversed data accurate and consistent.

(6) Security and privacy

In addition to the consideration of security and privacy in a single site data base with local users, the communications among the remote sites in DDB network represent further potential dangers. The message can be lost, changed or even sent to an unauthorized receiver during transmission in networks.

Attribute	Design Issue
User Transparency Local Autonomy	System Architecture
Data Distribution	Resource Allocation
Efficient operation	Query processing
Reliability & Integrity	Synchronization Concurrency
Security & Privacy	Data Protection

Table 2.1 DDB Attributes and Design Issues

Each of these attributes presents a particular area of concern in DDB design. Table 2.1 shows the correspondence between these six attributes and data base system design issues. The problem of query processing is closely related to the efficient operation in a DDB. In chapter 4 we will discuss query processing and the problems arising from it.

2.3 Classification of DDB

Distributed data bases can be classified as either homogeneous or heterogeneous. The definition of each is given. The background and distinctions between them are also mentioned. Further problems associated with query processing in each of these two types of DDBs will be discussed in chapter 4.

2.3.1 Homogeneous DDB

A homogeneous DDB is one where several identical data base management systems exist at sites of a computer network. The hardware of each site may be identical, and permits each data base management system (DBMS) to cooperate in presenting either a global or local external schema to the users. [13].

2.3.2 Heterogeneous DDB

A heterogeneous distributed data base can be defined as a distributed data base where several dissimilar (local) data base management systems exist at sites of a computer network. The hardware at each site may be identical, but the software is dissimilar. On the other hand, a heterogeneous DDB allows the local team of data administrators and system designers to choose their own management and implementation techniques. This type of DDB is usually derived because of the following reasons:

(1) The variety of software data base systems produced in past years provides the possibility of different systems on a distributed environment. A recent issue of COMPUTER DECISIONS indicates over 30 products, as shown in table 2.2, on the market.

(2) A heterogeneous DDB may come up from an existing state of operation, which is performed under different software products, if users want to accomplish a data integration and sharing. In these circumstances, users have already made a high investment in application programs and this must be saved by allowing local installations to retain the data models and data languages which are currently in use. [19]

(3) A heterogeneous DDB is developed for the purpose of pursuing a policy of differentiation of tools and methods. Different software systems (as well as hardware systems) are selected at the very beginning for research and comparison. This approach can also avoid the inflexibility in a changing environment and the strong dependency from a single computer manufacturer. [13]

COMPUTER DECISIONS 1984 February
DBMS Software

Vendor	Package	Requirements	Price
Information Builders	Focus	IBM370, 43xx, PCMs under VM/CMS MVS/TSO, VS1	$66,000
		IBM PC XT	$1,595/unit
Intel	System 2000	IBM, Sperry, Control Data mainframes	$40,000 and up
Interactive Tech.	RDM	DEC PDP-11 under RT-11, RSX, RSTS/E	$2,495
		DEC VAX under VMS	$4,795
	RDM 300	DEC Pro 350	$995

+ many more

Table 2.2 Examples of Software Data base Systems in 1984

CHAPTER 3

NETWORK

A communication network is a system used to move information over short or long distance by electronically transmitted signals. The electrical transmission may be by wire, cable, microwave, optical fiber, satellite, etc. Because the network is the key factor in distributed data base systems, this chapter will review the basic structure and architecture of a network system and the considerations related to the query processing in distributed data base systems.

3.1 Network structure

The network system contains a collection of machines intended for running users' (i.e., application) programs. These machines, called hosts, are connected by the communication subnet also called subnet (i.e., transport system, transmission system), The job of the communication subnet is to carry messages from host to host. The communication subnet consists of two basic components: 1) transmission lines (path) and 2) switching elements. Transmission lines are often called circuits or channels. The switching elements are called IMPs (Interface Message Processors) in ARPANET[4] (which was created by the Advanced Research Projects Agency of the U.S. Department of Defense), and are sometimes referred to as communication computers, packet switches, or

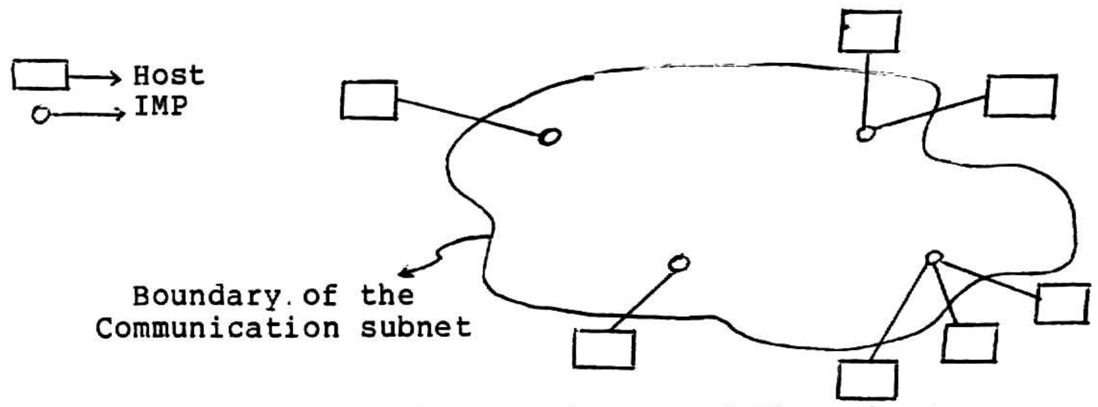


Figure 3.1 Relation between hosts and the subnet

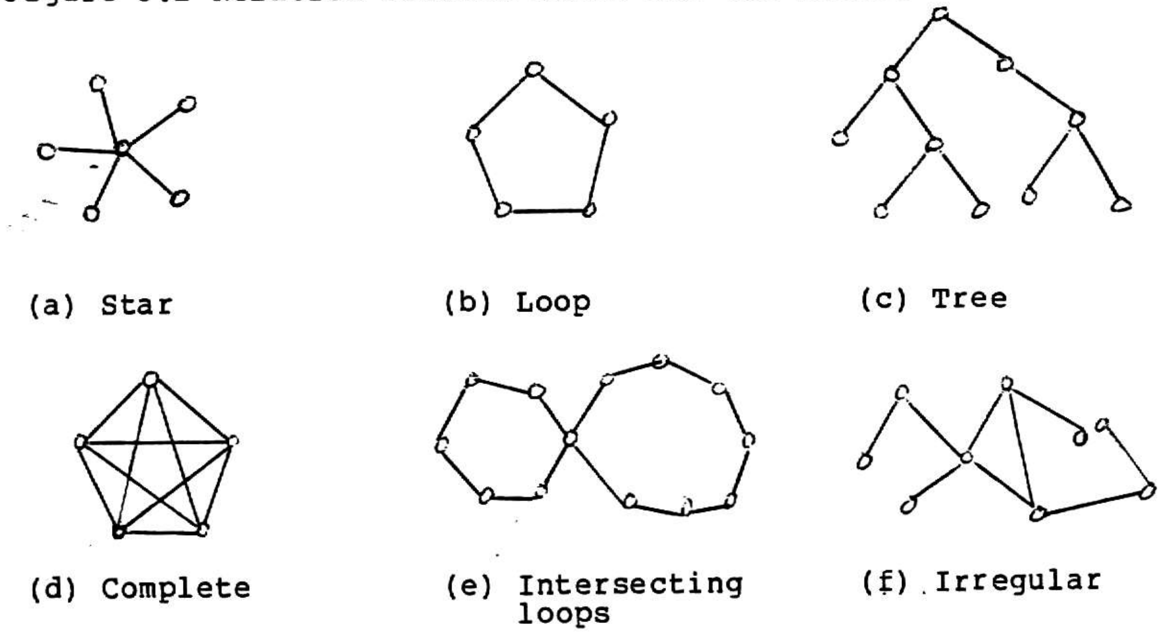


Figure 3.2 Some possible topologies for point-to-point subnet. (a) Star (b) Loop (c) Tree (d) Complete (e) Intersecting loops (f) Irregular

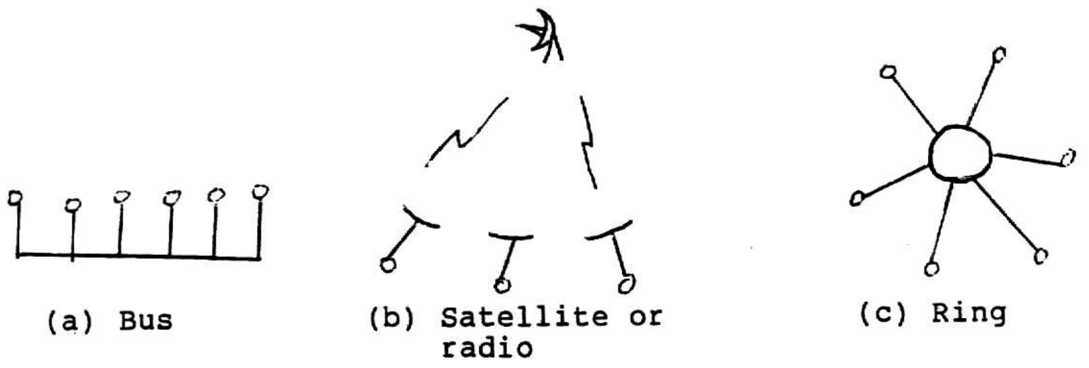


Figure 3.3 Some possible topologies for broadcasting subnet. (a) Bus (b) Satellite or radio (c) Ring

nodes. Figure 3.1 shows the relation between hosts and the communication subnet. All traffic to or from the host goes via its IMP.

The transmission path in the communication subnet can be a point-to-point or broadcast channel. In a point-to-point subnet, the network contains numerous cables or telephone lines, each one connecting a pair of IMPs. If two IMPs do not share a cable they must use other IMPs to communicate with each other. Several possible IMP interconnection topologies for a point-to-point subnet are shown in Figure 3.2. In a broadcast communication subnet, only a single communication channel is shared by all IMPs. If any IMP sends a message, it will be received by all other IMPs; therefore, the message, itself, must specify an IMP so that the other IMPs can ignore it. Some of the possible topologies for broadcast subnets are shown in Figure 3.3.

Broadcast subnets can be further divided into either static or dynamic subnet, depending on how the channel is allocated. Static allocation means to divide time into discrete intervals and time-division multiplexing, allowing each IMP to broadcast only during its individual time slot. Dynamic allocation methods for a common channel are either centralized or decentralized. A centralized allocation uses a single entity to determine which will be next while the decentralized allocation lets each IMP decide, itself, whether to transmit or not.

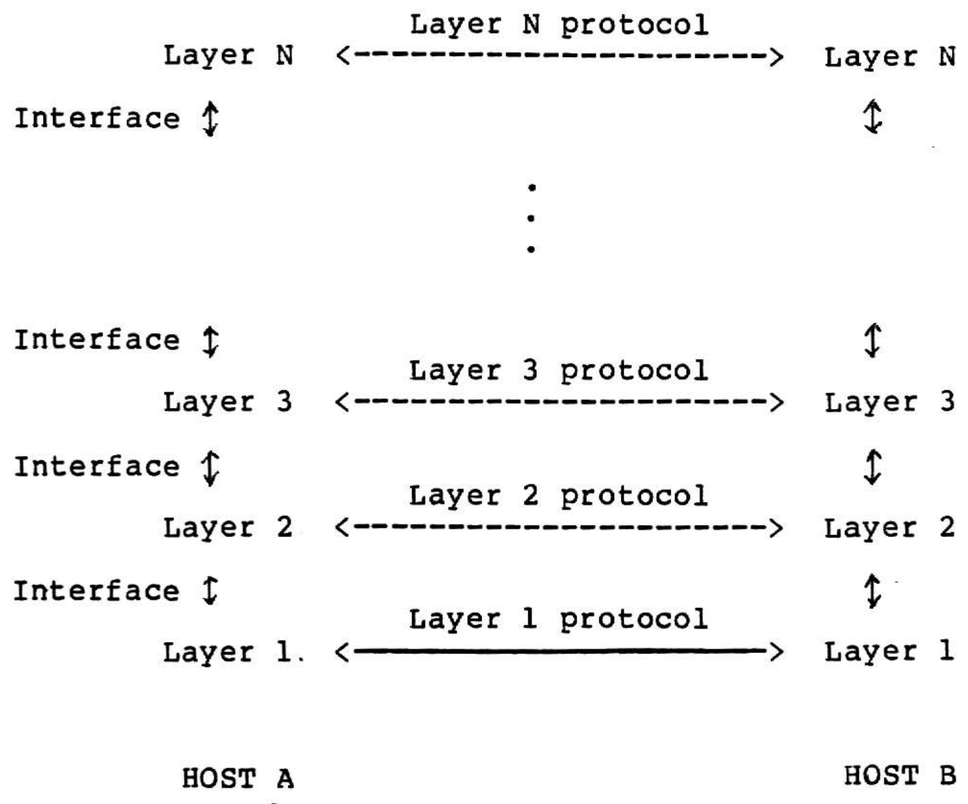


Figure 3.4. Layer, protocols, and interface.

3.2 Network architecture

Most networks are organized as a series of layers or levels as shown in figure 3.4. Layer N on one machine carries a conversation with layer N on another machine through a protocol. However, no data is, in fact, directly transferred from layer N on one machine to layer N on another machine except in the lowest layer. Only at the lowest layer is there a physical communication with the other machine. There is a virtual communication path between two peer entities at each higher level. In other words, each layer passes data to the layer immediately below it until the lowest layer is reached; then the data is passed through the physical lines up through the protocol layers to the destination entity.

Messages must be broken into smaller units before the lowest level is reached in order to transmit data through the communication line. This must be done in such a way that these units can be sent in parallel and assembled at the destination in order to save response time. These units will concatenate with a header/trailer which contains control information, such as sequence numbers, to make sure the destination machine will receive the same messages.

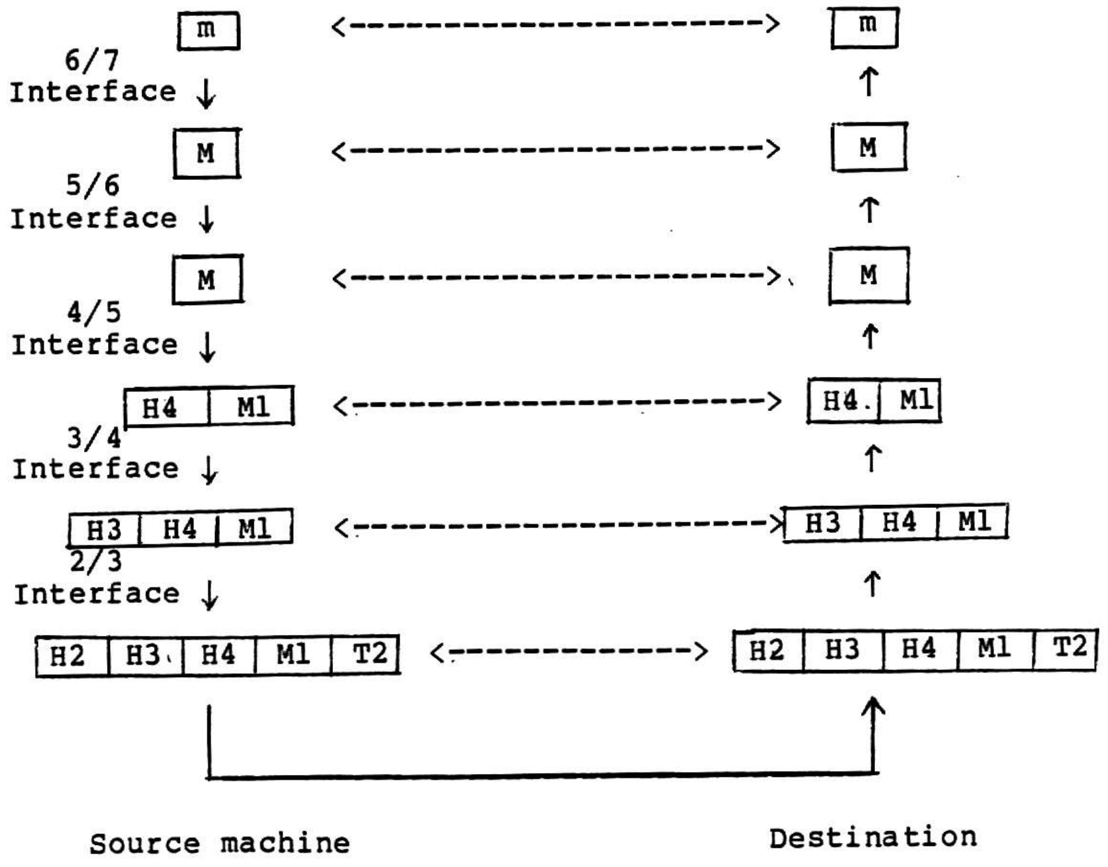


Figure 3.5. Actual information flow supporting virtual communication in layer 7.

For example, in figure 3.5, for a message to pass in layer 7 of host A to layer 7 of host B, it actually follows the solid line instead of the dotted line. i.e. the message is passed to its lower layer across the interface until the lowest level is reached for physical transmission. At the receiving machine, a reverse process occurs from layer to layer, with headers being stripped off.

There are various designs for protocol in network systems. The model shown in figure 3.6 is based on the Open Systems Interconnection recommendation developed by the International Standards Organizations [4]. Each layer performs a well-defined function. The basic purpose of layered protocols is to reduce complexity, provide for peer-to-peer layer interaction across nodes, and allow changes to be made in one layer without affecting others.

3.3 Considerations in DDB

The structure and architecture of a network system as mentioned above, implies that data transfers through the network are much more costly and slow than traditional transfers between secondary and central memory. The network may also lose messages, send duplicate messages, and deliver messages out of order. All these potential dangers to messages as well as other problems, such as the security in broadcasting, standards on communication systems, and time delay in transmission, should be well examined in the network communication system.

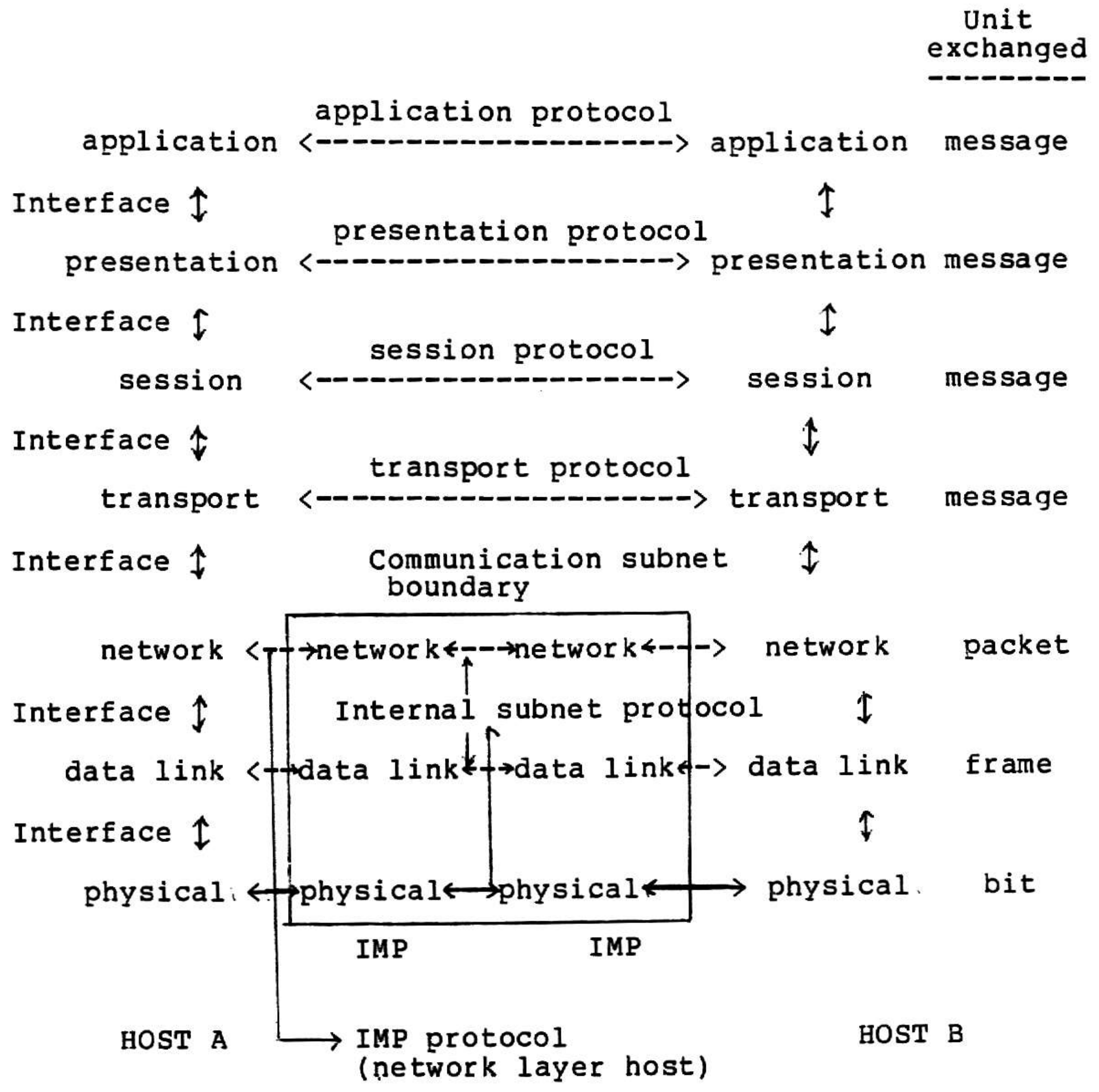


Figure 3.6. The network architecture based on ISO OSI reference model

The problem which will directly affect the query processing optimization in distributed data base systems is the time delay caused by traffic. When network traffic is light, time delay is primarily due to the time each node needs to process the message. However, as traffic increases, the principle delay becomes the queuing delay within each node plus the necessary processing time, since messages must wait their individual turns to be processed on a heavily-used channel. The technique used for message transmission and detection is also important, since any re-transmission for a lost message will make the time delay even worse. Another factor affecting the query processing is the use of satellite. Among the different transmission paths, satellite communications are unique. This technology provides a large communication capacity, and the cost and time required to transmit a message are independent of the distance between sender and receiver. To send a message through satellite to a place 10 miles away takes the same time as to a place 1000 miles away. This property can simplify the optimization of query processing in DDB by eliminating distance as an essential cost consideration.

CHAPTER 4

QUERY PROCESSING

4.1 Definitions

A query is a request to the data base to restrict its contents to some item or a set of items [6]. Query processing in a distributed system, which requires data from two or more distinct nodes in the network, consists of an arrangement of local data processing and data movements among diverse nodes.

4.2 The architecture of queries

The evolution of a query is represented by the global software architecture in figure 4.1. It shows that query processing in a DDB corresponds to the translation of requests formulated by users in a high-level language on one computer of the network, into a sequence of elementary instructions which could retrieve data stored in the distributed data base.

This diagram can be subdivided into two parts, linked by the communication network. The lower part of the architecture corresponds to the functions of a local DBMS which can be considered as a centralized data base. The upper part, the node where the query is generated and analyzed, includes functions like supervision and control of operations directly related to the distributed environment.

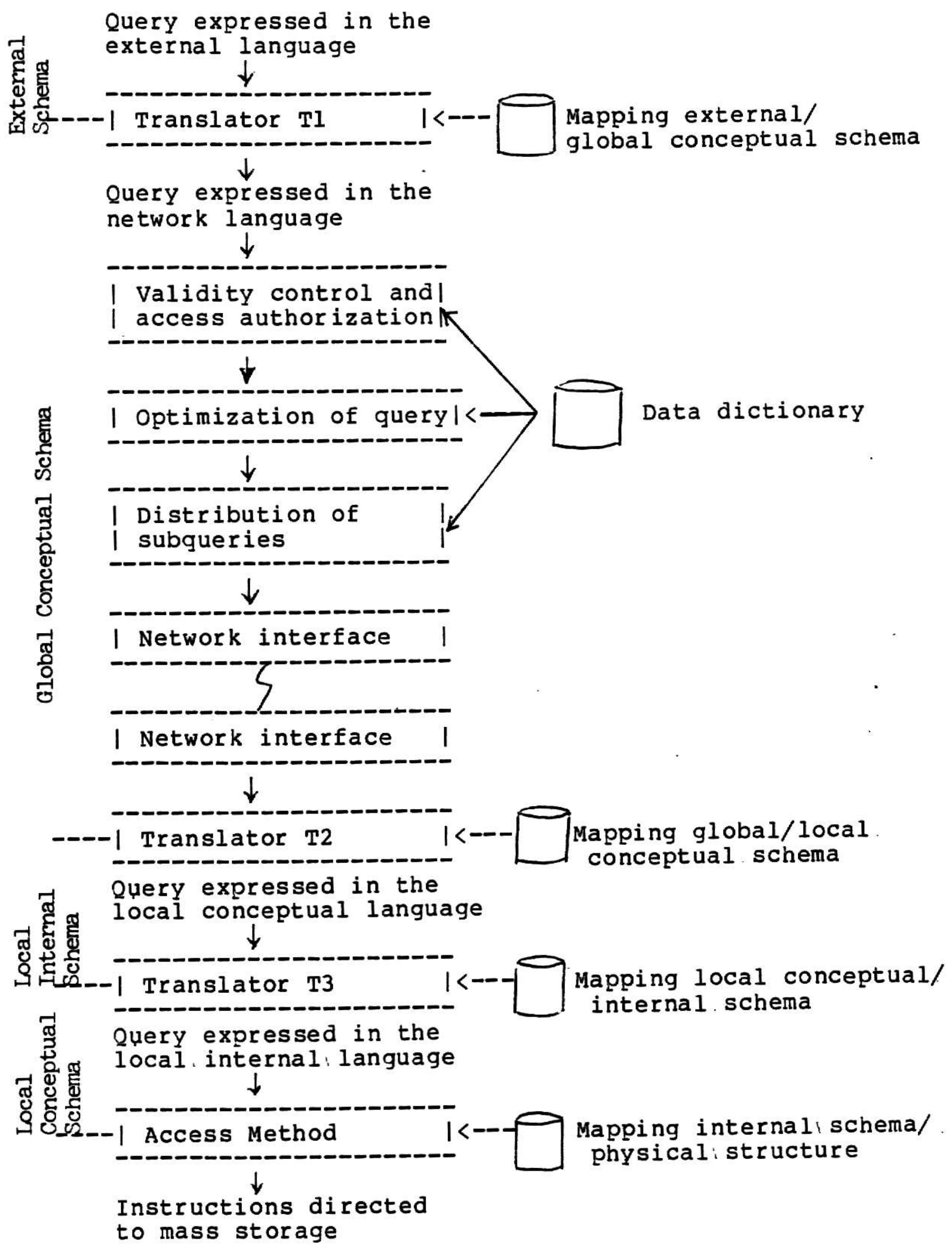

Query expressed in the external language
External Schema
Translator T1
Mapping external/ global conceptual schema
Query expressed in the network language
Validity control and access authorization
Optimization of query
Data dictionary
Distribution of subqueries
Network interface
Global Conceptual Schema
Network interface
Translator T2
Mapping global/local conceptual schema
Query expressed in the local conceptual language
Local Internal Schema
Translator T3
Mapping local conceptual/ internal schema
Query expressed in the local internal language
Local Conceptual Schema
Access Method
Mapping internal schema/ physical structure
Instructions directed to mass storage

Figure 4.1 Global software architecture of a DDB

The optimization process in the upper part, according to the strategy chosen, produces the optimal sequence of operational commands to the local data bases. Each command is further analyzed by the local DBMS of the remote computers, and then the optimal local data retrieval strategy is performed.

As discussed in chapter 2, the DBMS at each node is identical in homogeneous DDBs. The translator T2 at each node (see figure 4.1), is rather simple. In heterogeneous DDBs, this translator will be more complex, and this complexity is highly related to the type of system involved. The translator cost is likely to be high and, in some cases, unacceptable. The choice between homogeneous and heterogeneous DDB should be carefully considered. The cost of past investments should be computed as well as the cost of developing a new system. Re-using the existing environment provided with additional translators may be more expensive than developing a new system independent of the existing one.

4.3 Data characteristics

Some aspects of data which are related to query processing are described. Information on data distribution, data redundancy, data directory, and data dictionary is very important for the processing, optimization and routing of queries. They are the essential parameters when query processing strategies are examined.

R

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b2	c2	d1	e1
a3	b1	c3	d2	e2
a4	b2	c3	d2	e3
a5	b3	c4	d2	e4
a6	b3	c4	d2	e4

(a) Partitioning by Occurrence

SITE X

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b2	c3

SITE Y

R2

A	B	D
a4	b2	d2
a5	b3	d2
a6	b3	d2

(b) Partitioning by Structure

SITE X

R1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b1	c3
a4	b2	c3
a5	b3	c4
a6	b3	c4

SITE Y

R2

D	E
d1	e1
d2	e2
d2	e3
d2	e4

Figure 4.2 Example of Data Distribution in
(a) Partitioning by Occurrence
(b) Partitioning by Structure

4.3.1 Data distribution

In a distributed data base environment, data can be stored by partition. Partitioning may be by occurrence (horizontal splitting) or by structure (vertical splitting). In partitioning by occurrence, specific occurrences are allocated to specific sites. For instance, in figure 4.2 if we take relation A, then occurrences 1,2,3 may be stored at site X while occurrences 4,5,6 may be stored at site Y. In partitioning by structure. specific relations are allocated to specific sites. For instance. relations A,B,C may be allocated to site X, while relations D, E may be allocated to site Y.

The purpose for data distribution is to store data on the computers where it will be used most often. In such cases, it will reduce data transmissions between sites. If relations are both partitioned by occurrence and by structure over several machines, then the execution of a distributed query will be more difficult. This is especially true when data is duplicated. For example. if all occurrences of relation A. B, C are stored at site X, occurrences of relation A, C, D are stored at site Y, and all occurrences of relation C, D are stored at site Z, then. if the user's query needs the data for relation A. C, D, he will have several options for obtaining the required data.

4.3.2 Data redundancy

In a distributed data base system, data can be duplicated in a planned way. When data is stored at more than one site, the availability of the data is higher than if it is stored in only one place. Duplicated data enhances locality of reference by satisfying more read requests locally. This will reduce the response time for a request and will also reduce traffic on the communication network.

Meanwhile, it offers the opportunity to distribute the workload on a frequently-accessed file.

Another advantage of duplication is that it provides greater reliability through back-up copies from which data can be recovered in the event of machine failure. The complexity of 'updating' to achieve consistency in a DDB will increase, as well as the query processing optimization. Since it provides the opportunity of choice among various sites, the efficiency, utilization, and other important characteristics of each site should be taken into account upon query processing optimization.

4.3.3 Data dictionary and data directory

A data directory is a set of tables which drives the internal DBMS programs, providing mapping information between one level of schema and another, whereas data dictionaries provide the schema information to the user. The data dictionary must contain all information needed for the execution and optimization of queries. This information should contain the information as follows:

- the general characteristics of data:
  relation and domain names, attributes and field length, etc.
- data distribution:
  the allocation of data
- data access control:

read, write, update, etc.

- statistical information on the distribution of values of data: selectivity
- other information describing the state of the DDB

The correctness and distribution of the data dictionary/directory are deeply related to the efficiency of query processing. Statistical estimators such as selectivity, an estimator of the volume of data (number of tuples) resulting from the processing a specific subquery, allow the prediction of the expected size of local query results. This predictor is an essential parameter in query processing optimization and must be correct and up-to-date.

An effective criterion for distributing data dictionary/directory is to keep it close to the programs that will use it. Moreover, the distribution criteria of the data dictionary/directory is influenced by network topology and storage cost. Some users' computers, such as personal computers, are probably unable to keep this extra information in their limited storage. Furthermore, the redundant data dictionary/directory also brings more overhead on updating and consistency. In a hierarchical DDB where a central computer supervises the distributed system, it might be more convenient to store the whole data dictionary/directory on the central computer where the query optimization takes place.

4.4 Query language

Query language is the notation for expressing queries in order to retrieve required data. Some query languages such as ISBL, QUEL, SEQUEL and SQUARE have been produced and are widely used in data base systems. Details of their usage are discussed in [15]. Since the choice of a specific language is irrelevant for the optimization of query processing, their features are not discussed in this paper. However, we need an equivalent representation of the query showing its semantic implications. A subset of Codd's relational algebra is introduced as the basis for both the application level (user) data base access language and as a common network-wide access language in the distributed environment.

The unit of data is a relation. a two-dimensional table in which each row is a record, or tuple. of the same type. Each column of the table consists of a set of values that is called a domain of an attribute of the relation. The query processing is operated by using the Relation Query Language (RQL) which is limited to the PROJECTION. RESTRICTION and JOIN operations defined as follows :

PROJECTION

Let R(A) be a relational variable. Let $B \in A$. The Projection of R over B, denoted R[B] is a relation defined as

$$R[B] = \{ x.B \mid x \in R \}$$

JOIN

Let R(A), P(B) be two relational variables. Let x denote tuples over A.B. The join of R and P, denoted R*P, is a relation defined as

$$R*P = \{\ x \mid x.A \in R.\ x.B \in P\ \}$$

RESTRICTION

Let R be a relational variable and C a qualification variable. The restriction of R by C, denoted R|C is a relation defined as

$$R|C = \{\ x \in R \mid (x.c.E)(c.\theta)(c.v),\ c \in C,\ x \in R\ \}$$

where C, E is the name of a qualified attribute in a relation, c.θ is a boolean condition operator and c.v is a qualified value.

CHAPTER 5

QUERY PROCESSING OPTIMIZATION

5.1 The problem of query processing in DDB\

The problem of query processing optimization can be seen in the following :

"For a given user's query, the system has to determine how to transfer relations among sites and the ordering of local operations within site. Most of all, make it efficient."[14]

The following example can illustrate how this problem is involved in a distributed environment. In a computerized library system, the library data base consists of relations containing data on books, borrowers, libraries, publishers, and transactions involving these entities. This data is stored in diverse locations. The relations and their attributes referred to in the example are shown in figure 5.1. Consider the following query:

"Get the names and locations of libraries in the state of Kansas which have books on data base theory"

Such a query might be important to a publisher who wishes to offer these libraries a new book on data base systems. A graphical representation of the query is shown in Figure 5.2. The incoming arrows specify the local restrictions, the outgoing arrows specify the output domains and the double lines connect the join domains.

SITE A The LIBRARY relation

LIBRARY	LIBRARY NO.	LIBRARY NAME	LOCATION	STATE
	10	LIB1	MANHATTAN	KS
	15	LIB2	BOSTON	MA
	20	LIB1	ROCHESTER	NY
	.	.	.	.
	.	.	.	.

SITE B The BOOK relation

BOOK	ISBN	TITLE	KEYWORD	AUTHOR
	100-3	Data base Design	101	Smith
	.	.	.	.
	.	.	.	.

The CLASSIFICATION relation

CLASSIFICATION	KEYWORD	DESCRIPTION
	100	Computer Algorithms
	101	Data Base theory.
	.	.
	.	.
	.	.

SITE C The LIBRARY-BOOK relation

LIBRARY-BOOK	LIBRARY NO.	ISBN
	10	100-3
	15	156-0
	.	.
	.	.
	.	.

Figure 5.1 Relations of a Library DDBMS

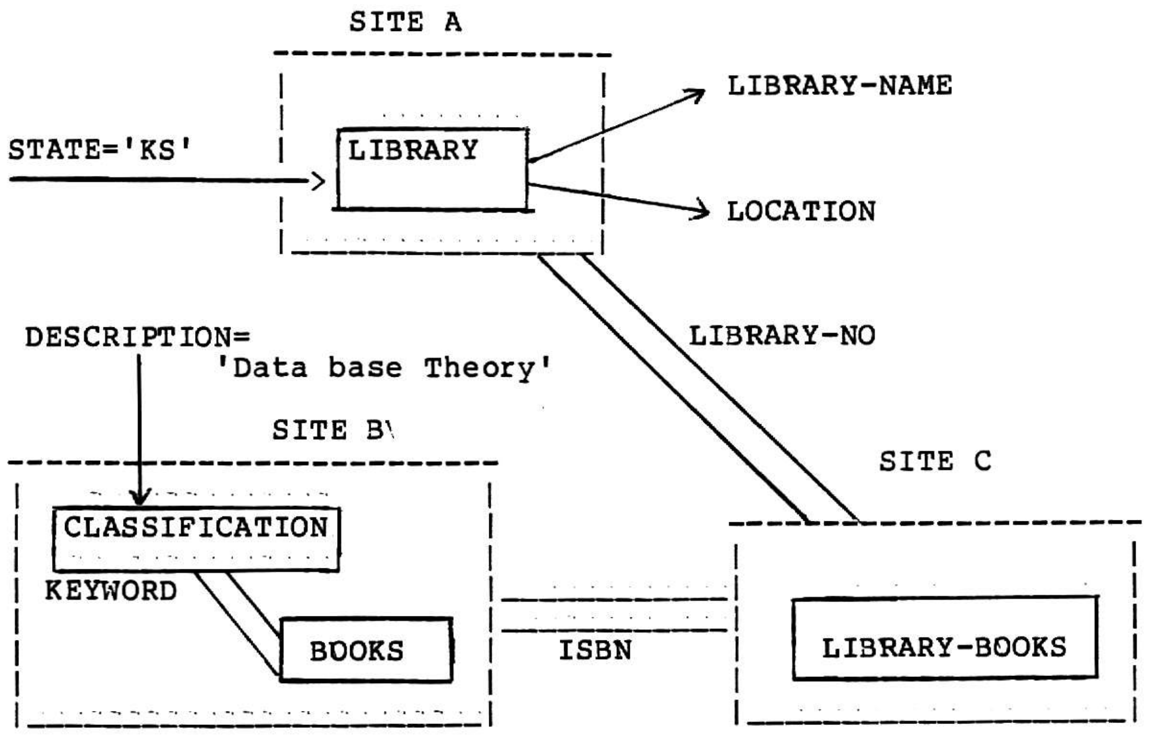


Figure 5.2 A Graphical Representation of the Publisher's Query:

Assume that the user query is located at a different site. After the local processing has been done, there are many ways to join relations between sites. Three possible alternatives are presented as follow:

<u>1st alternative</u>

1. transfer the resulting ISBN relation from SITE B\ to SITE C.
2. Transfer the restricted library relation from SITE A to SITE C.
3. Join LIBRARY, ISBN and LIBRARY-BOOKS relations and project the result at SITE C.

2nd Alternative

1. Transfer the restricted LIBRARY relation from SITE A to SITE C.
2. Join LIBRARY and LIBRARY-BOOKS relations at SITE C and Project over LIBRARY-NAME, LOCATION and ISBN.
3. Transfer the resulting relation to SITE B\.
4. Join and project the result at SITE B\.

3rd Alternative

1. Transfer the resulting ISBN relation from SITE B\ to SITE C.
2. Join ISBN and LIBRARY-BOOKS relations at SITE C and project over LIBRARY-NO.
3. Transfer the resulting relation to SITE A.
4. Join and project the result at SITE A.

Each of these alternatives is acceptable without considering the actual cost and response time. However, if the information of relations shown in table 5.1 is available and taken into account, then the result will be much different.

It is assumed that transmission costs are proportional to the number of bits transferred, and that transmission costs dominate processing costs. It is also assumed that the transmission delay is given by a constant equal to 0.25 seconds, plus a portion which is proportional to the number of bits transferred and is given by volume(in bits)/50,000 bps.

Relation	Cardinality	Attribute	Size(Bits)
LIBRARY	20,000	LIBRARY-NO.	40
BOOK	1,000,000	LIBRARY NAME	160
CLASSIFICATION	10,000	LOCATION	120
LIB-BOOKS	100,000,000	STATE	16
		ISBN	120
		TITLE	160
		KEYWORD	24
		AUTHOR	160
		DESCRIPTION	160

5,000	Libraries have books on Data Base Theory.
500	Books are on Data Base Theory.
2,000	Libraries are in KS state.
20,000,000	Books are available in KS state's libraries.

Table 5.1 Characteristics of the relations

Alternative	Transfer	Transmission time	Response time
1	Site B > Site C	1.45 sec.	
	Site A > Site C	13.05 sec.	13.05 sec.
2	Site A > Site C	13.05 sec.	
	Site C > Site B	~ 45 hrs.	45 hrs.
3	Site B > Site C	1.45 sec.	
	Site C > Site A	4.25 sec.	5.7 sec.

Table 5.2 Performance of the transfer policies

The performance of the three transfer policies is shown in table 5.2, which indicates that the second alternative is totally unacceptable and the third alternative has both the minimum cost and shortest response time. The second alternative consists of two serial transfers. In the first transfer, the result relation of SITEA, which consists of the attributes LIBRARY NO., LIBRARY NAME and LOCATION, of the 2000 tuples with STATE = 'KS', is transferred to SITEC. The transfer time equal to 0.25 + 2000(40+160+120)/50,000 = 13.05 seconds. In the following transfer, all the LIBRARY NAME, LOCATION, and ISBN of 20,000,000 books in KS library should be transferred to SITEB. The transfer time will be 0.25 + 20,000,000(160+120+120)/50,000 = ~ 45 hrs. The calculations for the other alternatives are performed in a similar manner. This example illustrates the important and essential need to pursue an acceptable query processing algorithm in the distributed data base system.

A reasonable strategy for query processing should begin with all local processing, which consists of RESTRICTION, PROJECTION and JOIN operations referred into a query at a single site, followed by the operation between sites. The purpose of local processing is to reduce the amount of data that needs further processing. PROJECTION eliminates unneeded domains from relations. RESTRICTION selects rows of a relation that satisfy specified data conditions. JOIN combines relations and, in the process, eliminates rows whose domain values do not match between relations. This operation is also known as a JOIN RESTRICTION.

The decomposition procedure, which analyzes the query given by users into a series of subqueries, should be done before local processing. These subqueries are passed to the node, where the consequent relations are allocated for local processing. Following local processing is the specification of an execution sequence which constitutes an ordering of the above query nodes. To determine the most efficient ordering of execution becomes the critical part of query processing optimization. Hevner & Yao[1,2,3] developed an algorithm to generate efficient ordering to accomplish query optimization. Unger et al.[6] presented a paper, recently, that took more consideration of the local processing in order to determine consequent routing. Their research is reviewed in the following section.

5.2 Current research

Many research efforts have been conducted in pursuing an optimal strategy for query processing [7,19,21], but in the specific area of the ordering of execution after local processing, only Hevner , Yao and Unger et al. concentrate on this problem. Their strategies are reviewed, and differences between them are compared and evaluated.

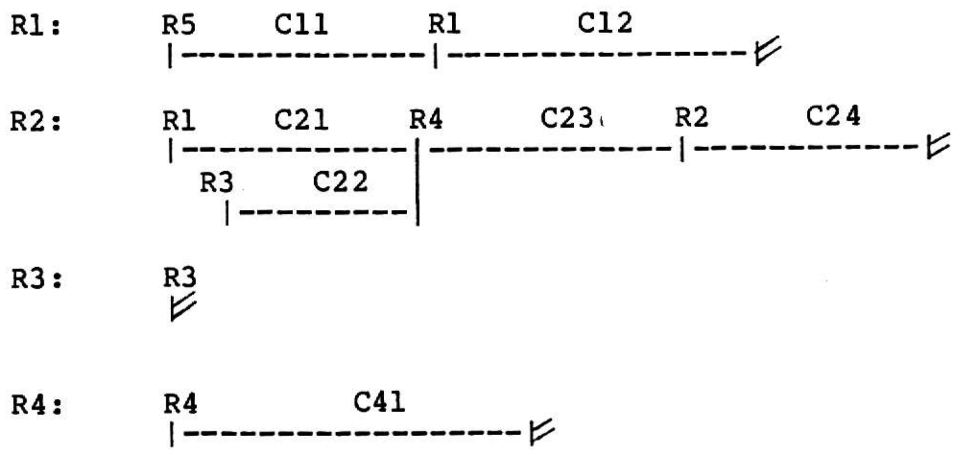
5.2.1 Hevner & Yao's algorithm

Hevner & Yao proposed an optimization strategy defined as the arrangement of data transmissions and local data processing based on two cost measures: response time and total time. Response time is the time from the start of transmission until the required relation is available at the result node (user Node). Total time is the sum of all transmission time required in the strategy.

The cost is measured in units of time. The data transmission cost between any two nodes is defined as a linear function $C(X) = C0 + C1X$, where X is the amount of data transmitted. The constant C0 represents an initial start-up cost for each separate transmission. An important property of $C(X)$ is that if $X < Y$ then $C(X) < C(Y)$. i.e., the cost and time of a query is dependent on the amount of data.

It is obvious, for transmission media of equal cost and speed, that minimizing the cost of a data transmission is equivalent to minimizing the amount of data transmitted. The transmission between nodes in a network is significantly slower and more costly than the data movements and processing within local nodes. Compared to the cost of network transmission, the local processing costs are considered to be insignificant. Furthermore, the distance of transmission is not considered as a factor to the optimization as the advances in communication technology, such as satellites, will stabilize its influence.

In order to clearly visualize the structure and interaction of the algorithm, a cost graphic notation of a distribution strategy, as shown in figure 5.3, is adopted, in which the horizontal line (i.e. $|\overset{C(X)}{\text{------------}}|$) represents each data transmission. In figure 5.3, each relation is originally at a separate node. R3 is at the result node. For the relation R2 in the cost graph, data is transmitted in parallel from R1 and R3 to R4. Relation R4 is reduced in size and then data from it is transmitted to R2. Relation R2 is also reduced in size and transmitted to the result node.



R3 is at the result node.

Response time = C21 + C23 + C24
Total time = C11 + C12 + C21 + C22 + C23 + C24 + C41

Figure 5.3 Example Cost Graph of a Distribution Strategy

The basic tactic considered in constructing such an optimal strategy is the 'small to large' tactic, in which data transmission is done from small relations to large relations. Another tactic is to make the most advantageous use of parallelism on the network. The operations that occur in parallel with existing operations do not contribute to the response time with this strategy. The following example illustrates this methodology for finding a distribution strategy.

Example : Assume a query requires data from relations R1, R2, R3, R4, which are located at separate nodes. Let the size after local processing and selectivities be :

Relation	Size(Si)	Selectivity(Pi)
R1	200	1/5
R2	500	1/2
R3	500	1/2
R4	1000	1

Let C(x) = 20 + X

The initial feasible solution cost graph is :

```
             220
    R1: |-------|>
             520
    R2: |-----------------|>
             520
    R3: |-----------------|>
             1020
    R4: |-----------------------------------|>
```

Response time = 1020

Total time = 2280

The initial feasible solution is to transfer all the required data to the user's node where the query is created, after each local processing. This is the most nonbeneficial strategy for processing a query since before the user's node is reached, the amount of data can be reduced by applying the operation of JOIN and/or RESTRICTION to any two relations. Hevner & Yao's algorithm takes this strategy as a basis, and then examines each relation for better substitutes, until the optimal solution is achieved. Since relation R1 is the smallest relation, data transmission from R1 is beneficial. Instead of transmitting relation R1 to the result node, transmit its result after local processing to R2 for further processing. After the operation of JOIN, the new cost graph for schedule R2 is :

```
       R1      220     R2  120
 R2:   |------------|-------⊨
```

The same strategy is used for relation R3 and R4. Finally, the optimal ordering of the relations (nodes) is obtained:

```
         R1      220     R2  120  R4
 R4 :    |-------------|--------|    70
         R1      220     R3  120  |------⊨
         |-------------|--------|
```

Response time = 220 + 120 + 70 = 410

Total time = 220 + 120 + 220 +120 + 70 = 750

The final strategy is to transmit relation R1 to both locations, R2 and R3, do the local join, and transmit the result to location of relation R4. After the local processing of R4,(i.e. the join of R2, R3 and R4,) transmit the result to the result node (user node). Compared to the initial feasible solution, both the response time and total time have significantly dropped.

If the cost objective is to minimize the total time only, then an ordered serial strategy is applied, i.e. eliminate the parallel operation. The final cost graph for the above example will be:

```
    R1    220       R2   120   R3 70  R4 70
R4: |--------------|----------|------|-------⊭
```

Response time = 480
Total time = 480

This shows that the response time is higher than the other one, but the total time has dropped significantly. The choice of cost objective depends upon the load in the distributed system. In a lightly-loaded system with few queuing delays, response time minimization would be preferable. On the other hand, in a more heavily-loaded system, queuing delays may make total time minimization the better objective.

The above optimization algorithm is used for a simple query which contains only one domain, the common joining domain; however, in the real world the query may not be so simple. In a general query, each required relation may have any number of joining domains and output domains, and each node may contain several required relations. The algorithm presented above can be extended for such situations. Joining domains within each relation are assumed to be independent. An exhaustive search with the same tactics can be applied for each relation, each incoming domain, and each possible domain candidate until the most beneficial candidate is obtained.

5.2.3 Unger et al. algorithm

Unger et al.[6] proposed a query processing model which takes more consideration of the factors, such as redundancy, security, languages and processor's utilization of a distributed environment. This model consists of three steps defined as follows:

Step1.Selection of user topology. Since redundancy of the resources in a distributed environment is possible, a set of nodes and communication links in the network are selected which can satisfy the request of the query.

Step2.Restrictions of user topology. Based upon certain user-defined constraints, such as security, time, languages, the DBMS model used, and processors, the nodes and links of the user topology which cannot meet such constraints are eliminated.

Step3.Application of the optimization algorithm. According to the cost and time profiles of all the candidate nodes and links, select an optimal retrieval pattern.

The cost and time profiles mentioned in step 3 are the critical factor used for query processing optimization. This temporal profile, a graph of the features at a particular point in time [6], considered both the communication links of the network, (i.e. the transmission between nodes), and the local processing.

The profile for a communication link contains both the cost and time required, as determined by the factors of volume, distance, and type of transmission. Presently, and in the foreseeable future, a transmission that must travel a longer distance is charged a higher total rate as opposed to one which travels a shorter distance [6], requires the distance of a transmission to be taken into account.

Furthermore, the distinctions among types of lines, such as line speed and whether the line is dedicated or shared, are also considered in the temporal profile. The cost of a dedicated line is higher than the cost of a shared line .

For local processing, the possibility of data redundancy is considered, (i.e. if the query has a choice of processors (nodes), then the optimization algorithm should be able to select the one with the least total expenditure). Such expenditure should involve the cost and speed of the processor, main memory, secondary memory and file retrieval. The queuing wait and the utilization rate is also included in the expenditure.

The characteristics of primary memory and secondary storage among various nodes may be different. These differences will also affect the efficiency of query processing. Furthermore, the queuing techniques, file retrieval and organization will all affect the utilization of a local node. A temporal profile of each node consists of the above cost factors as well as the speed of each factor. The total node resource and a particular communication line expense will be the function of several costs and times defined as follows:

$NODE_c$ = f1(PC, MMC, SMC, FRC)

$NODE_t$ = f2(PC, MMS, SMS, FRS, UR)

$COMM_c$ = f3(volume, distance, line rate)

$COMM_t$ = f4(volume, distance, line speed)

where PC = processor cost

MMC = main memory cost

SMC = secondary memory cost

FRC = file retrieval cost

PS = processor speed and queuing wait

MMS= main memory speed

SMS = secondary memory speed

FRS = file retrieval speed

UR = utilization rate

When a query is given, the cost of each local processing, as well as the cost of transmission, will be calculated and used as the factor in the optimization algorithm. Both of their performances will determine the ordering of each local processing before the result node is reached. The optimal strategy will be the Minimum ($COMM_c$, $COMM_t$, $NODE_c$, $NODE_t$).

5.3 Evaluation

In a distributed data base system, the need for data retrieval optimization is evident. At the same time, the strategies should be manipulated as simply as possible for efficient implementation. The simplicity of this strategy is important as well as its optimization. Since the time and cost required for generating such optimal algorithm should also be included in the optimization, those factors with less influence and little optimal improvement should be eliminated.

Hevner & Yao's algorithm is developed in such a way that it is simple to use and is computationally efficient. The possibility of data redundancy is not considered in their algorithm. If the user has the choice of two or more data resources, then the characteristics of each resource should be evaluated in order to select the most cost beneficial one. Meanwhile, query processing can not work alone in the distributed data base system. It is interdependent with other subsystems such as integrity, security. The restrictions for a certain processor, language, and communication link should also be considered before an optimal and feasible action can be taken. The step 1 and step 2 in Unger's query processing model have seriously considered this situation and they are also the essential procedures before applying Hevner and Yao's algorithm.

After considering the above situation, i.e. the locations of data resources for each subquery are achieved, how do we manage the following execution among these subqueries to be an optimal one? The differences of such execution between Hevner & Yao's algorithm and the third step of Unger's model are examined as follows:

Hevner & Yao

Step 1 : Order relations from small to large for relation Ri, i = 1 .. n
i.e. R1 < R2 < ... < Rn

Step 2 : Based on the initial feasible solution [1,2,3], find the better substitute strategy for each relation.

```
R1   ---> R1'
R2   ---> R2'
     ...
Rn   ---> Rn'
```

Step 3 : According to the cost object, compare the response time and/or total time and select the optimal one.

Unger et al.

Step 1 : Collect the up-to-date utilization of each node that relations located and the performance of each local processor.

Step 2 : Compute the functions of $NODE_c$ and $NODE_t$ for all the nodes involved and functions of $COMM_c$ and $COMM_t$ for all the possible communication links between any two nodes.

step 3 : Compare all the possible user topologies, select the optimal one.
i.e. Minimize ($COMM_c$,$COMM_t$,$NODE_c$, $NODE_t$)

The information required for Hevner & Yao's algorithm consists of the size of data domain (for step 1) and the selectivity of data domain after processing a subquery (for step 2). This information is part of a data dictionary which is easy to get and apply in the algorithm.

The information required for Unger's model comes from temporal profiles. Each temporal profile has to provide the current status of the node utilization as well as the other criteria, such as the speed and cost of the local processors, and memories, upon request. The temporal profiles can be updated either automatically after some interval of time has passed or at the end of step 2 of Unger's original algorithm. When automatic updating of temporal profiles is performed, the information from the profile is immediately available when needed in query optimization. The temporal profile needs to be updated when such information is required. The choice between these two ways of updating depends upon how often a query will be generated in the network system. If queries come and go so often, then the system should update the temporal profile automatically instead of doing so for every query. Since the overhead for updating this temporal profile consists of a request to the network for such action, the queuing wait, and the necessary time for updating and sending the result back to user, it will increase the cost and delay the time for processing such a query.

The other information included in the temporal profile is the evaluator of the local processor among different nodes. The techniques used for evaluating such performance among different machines are difficult or too costly to achieve. Since most of the DBMS software and hardware have their advantages and disadvantages, it is very difficult to evaluate the performance among different machines due to the large number of possible interactions. Imperfect information makes the estimate result unreliable.

Furthermore, the execution time required for generating these two so-called optimal strategies is considerably different. The step 2 of Hevner & Yao's algorithm is processed by the 'small to large' tactic. When we look for the substitute strategies for each relation, only the relations which are smaller than the current one will be considered and the algorithm will be run up to N times for N different relations. It is assumed that each relation is located at a different node. (It is actually calculated for N-1 times, since there is no substitute necessary for the smallest relation.).

In the step 2 of Unger's algorithm, the execution time required for calculating the functions of $COMM_c$,$COMM_t$,$NODE_c$,$NODE_t$ is large. The possible ways in which a temporary query result can be passed among N node are N!. Therefore, we need to calculate N! times for the functions of $COMM_c$ and $COMM_t$, i.e. N! different results of $M(COMM_c, COMM_t, NODE_c, NODE_t)$ are computed. It is also assumed that each relation is located at a separate node. The execution time can be longer if we consider the possibility of parallelism in the network and duplicate data resources, (i.e. the alternative nodes). Unger's strategy will be more expensive and less efficient than Hevner & Yao's.

Unger's query processing model is more of a general guildline than a specific algorithm. It can be broken into less complicate algorithms for different situations and cost objects. The cost and time for a communication link always conflict with each other, i.e. the faster the speed, the higher the cost. This trade-off also happens among different processors, memories and file retrieval techniques. The query itself should decide its priority, i.e. the time or the money, or the constraints on each. If a query is generated under emergency situation, users only consider the time it requires, then the optimal strategy will be selected based on MINIMIZE $(COMM_t, NODE_t)$. On the other hand, for routine queries where the primary cost object is the money, the optimal strategy will be MINIMIZE $(COMM_c, NODE_c)$.

In a homogeneous DDB system, where the hardware and software of all the local data bases (processors) are identical, the differences among the various processors, memories, and file retrieval techniques will be small and should be tolerated. The major difference among these processors will be the queuing wait. If a processor associated with many commonly used resources has a work backlog, then an alternative action should be taken. In such cases, the function of $NODE_t$ can be simplified as $NODE_t = f2(UR)$.

(utilization rate)

If we examine the weight among the three factors of communication link, we can find that the volume of data is still the most valuable factor. Because the major goal of a network system is to eliminate the problem and difficulty that the distance would create, the low cost and distance independent communication technology will possibly be accomplished in the near future. Meanwhile, in the local area network, the distance among any two nodes is not significant to be considered as a major factor. It can be dropped from the functions of $COMM_c$ and $COMM_t$.

Although the algorithm with more considerations will produce a more accurate evaluator to calculate the optimal solution, the less efficient performance would also come along with its increasing complexity. Solution time which increases the overhead of the query processing beyond a tolerable range is not acceptable. When we study the query processing optimization, it will be a controversy of optimal versus near-optimal algorithms. In most cases heuristic solutions will be used rather than optimal solution. A simple, heuristic, and user-friendly algorithm that is less optimal might be more welcomed than a complicated and exhaustive strategy with little optimal improvement.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

The introduction of distributed data base systems brings varied study and research in this field. Topics include data base theory, system architecture and design resource allocation, query processing, consistency, security, synchronization, and etc. This report has specific interest in query processing and its optimization.

A general distributed data base architecture has been presented together with an approach for formulating and coordinating query processing strategies. The structure and architecture of a communication network was also given. The considerations in network for a query processing approach were outlined. The problem and difficulties impact the query process, such as data transfer and data character were also reviewed. The information of data such as data allocations, data redundancy, statistical estimates, and etc., is all the basic resource that query processing requires.

Efficient query processing is an essential function of a distributed data base system. In this report, the problem of such efficiency is discussed. The strategies proposed by Hevner & Yao and Unger are both examined and evaluated. Hevner & Yao produced a query optimization algorithm which can be implemented computationally, and the information it needs are the size of domain and the selectivity. Unger produced a comprehensive guildline for query processing optimization. The utilization of each node as well as the communication link are well examined.

Query processing in a distributed data base system which implements the query optimization algorithm, is inter-dependent with other subsystems such as integrity, reliabil-ity, and concurrency. The query processing optimization algorithm needs the other subsystems to provide the accurate information; it may have to implement under some constraints from other subsystems. This report is intended to provide the comprehensive knowledge of distributed data base systems in case of query processing optimization. It seems that the solution of optimization problems lies in the development of good heuristics.

REFERENCES

[1] A. R. Hevner & S.B. Yao,
Optimization of Data Access in Distributed Systems
Computer Science Dept. Purdue University TR-281, 1978

[2] A. R. Hevner & S.B. Yao,
Query Processing in Distributed Database System
IEEE Transactions on Software Engineering, Vol. SE-5, NO. 3,
May 1979

[3] A. R. Hevner & S.B. Yao,
Query Processing on a Distributed Database
Computer Science Dept. Purdue University Tech. Rep. 1978

[4] Andrew & Tanenbaum
Computer Networks
Prentice - Hall, Inc. 1981

[5] B.W. Lampson, M.Paul, H.J. Siegert(editor)
Distributed Systems - Architecture & Implementation
Springer - Verlag, 1981

[6] E. A. Unger, et al.
Query Processing on a Distributed Database
Computer Science Dept. Kansas state University
Manhattan, KS, Tech. Rep. 1984

[7] Eugene Wong & Karel Youssefi
Decomposition - A Strategy for Query Processing
ACM Transactions on Database Systems Vol. 1. No. 3 Sept 1986

[8] George a. Champine
Distributed Computer System
North-holland Publishing Co. 1980

[9] Gregore Von Bochmann
Concepts for Distributeed Systems Design
Springer - Verlag, 1983

[10] H. Chao
Query Processing in a Distributed Environment
Master Report, Kansas State University
Manhattan, Kansas 1982

[11] Howard L. Morgan & K. Dan Levin
Optimal program and Data Locations in Computer Networks
Communications of the ACM, Vol 20, NO 5. May 1977

[12] H. J. Schneider et al.
Distributed Data Bases
Proceedings of the second International symposium on Distributed Data Base
North-holland Publishing Co. 1982

[13] I. W. Draffan and F. Poole
Distributed Data Bases
Cambridge University Press 1980

[14] J. Akoka et al.
Management of Distributed Data Processing
Proceedings of the International Conference on Management of Distributed Data Process
North-holland Publishing Co. 1982

[15] Jeffery D. Ullman
Principles of Data Base
Computer Science Press 1979

[16] James Martin
Computer Networks and Distributed Processing
Prentice-Hall, Inc.,1981

[17] J. D. Scharf,
Department of Defense Computer Network Security:
An Assessment of the State of the Art.
Master's Report, Dept. of Computer Science,
Kansas State University, Manhattan, KS, 1980

[18] Paul Fisher
Communications, Computers, Distributed Processing,
Networks, Protocols and other miscellania
Dept. of Computer Science, Kansas State University
Manhattan, Kansas 1981

[19] R. P. Van de Riet and W. Litwin et al.
Distributed Data Sharing Systems
Proceedings of the Second International Seminar on
Distributed Data Sharing Systems
North-holland Publishing Co. 1981

[20] R. K. Shultz and R. J. Zingg
Response Time Analysis of Multiprocessor Computers for Database Support
ACM Transactions on Database System, Vol. 9, NO. 1, March 1984

[21] S. K. Chang and B. H. McCormick
Intelligent Coupling of the User to Distributed Database Systems
Medical Information Systems Laboratory 1979
University of Illinois at Chicago Circle, Chicago, Illinois

[22] S. Ceri and G. Pelagatti
Correctness of Query Execution Strategies in Distributed Databases
ACM Transactions on Database System, Vol. 8, NO. 4, December 1983

[23] DBMS Roundtable
Computer Decisions February 1984

Query Processing in a Distributed Data Base

by

Leou, Chinchin

B.S., National Chung Hsing University

Taipei, Taiwan, R.O.C. 1976

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1984

ABSTRACT

This Master's report describes the query processing in a Distributed Data Base System (DDBS). A general distributed data base architecture is presented. Each DDBS can be classified as either homogeneous or heterogeneous. The difficulty of query processing in a heterogeneous DDB is outlined. The structure of a communication network as well as its relationship to query processing in such environment are also examined.

The criteria of the query processing, such as data distribution, data dictionary, data directory, and data redundancy are also discussed. The algorithm chosen for query optimization is very important to the performance of the query processing in the distributed data base system. The efficiency of such algorithm should also be considered in pursuing optimization. The strategies proposed by Hevner & Yao and Unger et al. are both summaried and evaluated.