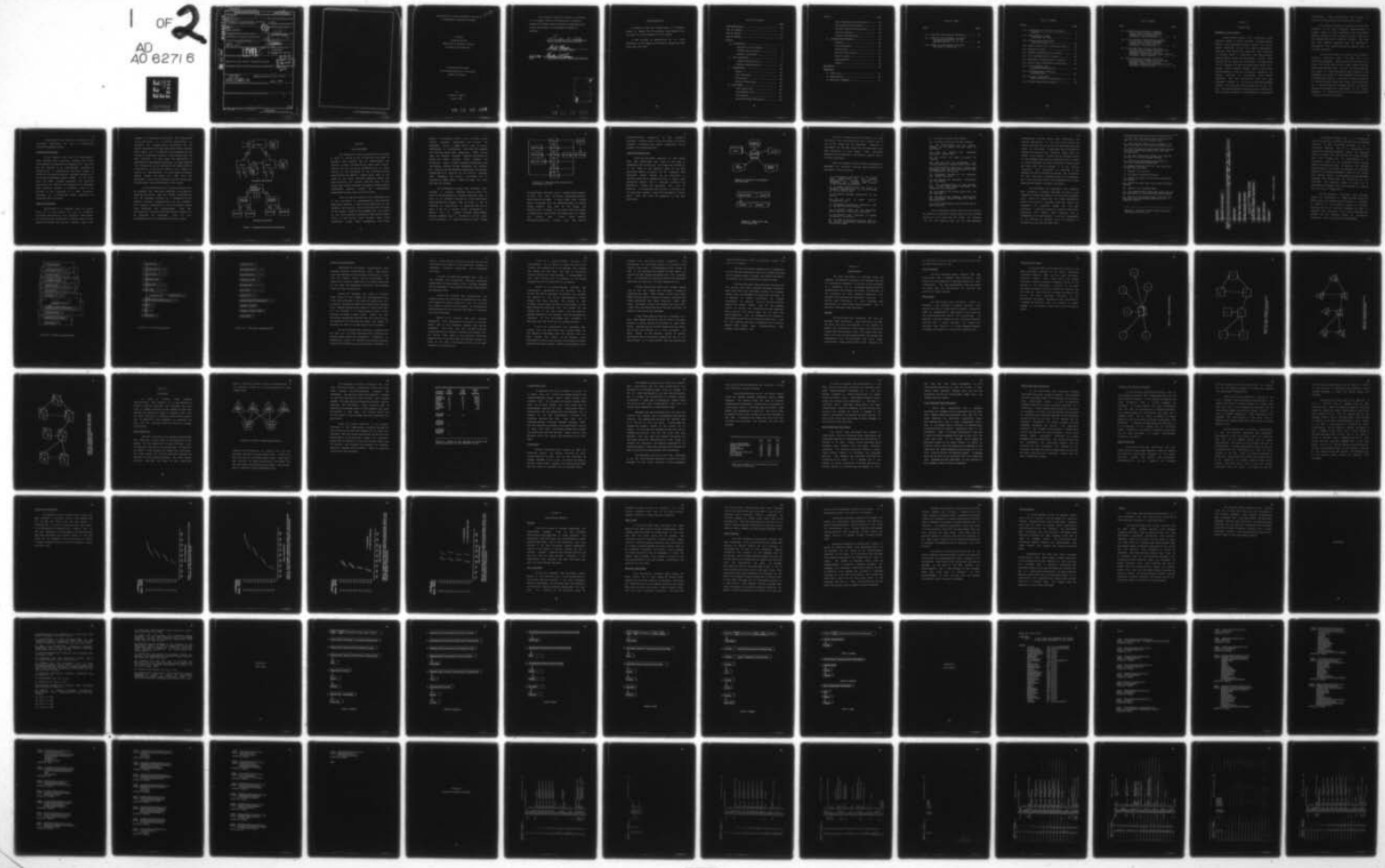
AD-A062 716 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
DEVELOPMENT OF A DATA DICTIONARY: FOR USE IN A DISTRIBUTED INTE--ETC(U)
AUG 78 G K POWELL

UNCLASSIFIED AFIT-CI-79-84 NL

1 OF 2
AD
A062716



SIFIED

1 OF 2

AD
AO 62716

DDC FILE COPY

AD A062716

1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

AD A062716

DDC FILE COPY

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CI 79-84
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE *(and Subtitle)*: Development of a Data Dictionary: For Use in a Distributed Integrated Database
5. TYPE OF REPORT & PERIOD COVERED: Thesis
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Gordon K. Powell
8. CONTRACT OR GRANT NUMBER(s): (12) 98p.

9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT Student at Brigham Young University
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (14) AFIT-CI-79-84

11. CONTROLLING OFFICE NAME AND ADDRESS: AFIT/CI, WPAFB OH 45433 (9) Master's thesis, (11)
12. REPORT DATE: August 1978
13. NUMBER OF PAGES: 88

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*:
15. SECURITY CLASS. *(of this report)*: Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

LEVEL II

16. DISTRIBUTION STATEMENT *(of this Report)*:

Approved for Public Release, Distribution Unlimited

DDC RECEIVED DEC 29 1978 F

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*:

18. SUPPLEMENTARY NOTES:

APPROVED FOR PUBLIC RELEASE AFR 190-17.

JOSEPH P. HIPPS, Major, USAF
Director of Information, AFIT

DEC 15 1978

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*:

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*:

012 200 JOB

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

DEVELOPMENT OF A DATA DICTIONARY: FOR USE IN A DISTRIBUTED INTEGRATED DATABASE

79-84

A Thesis

Presented to the

Department of Computer Science

Brigham Young University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Gordon K. Powell

August 1978

78 12 26 138

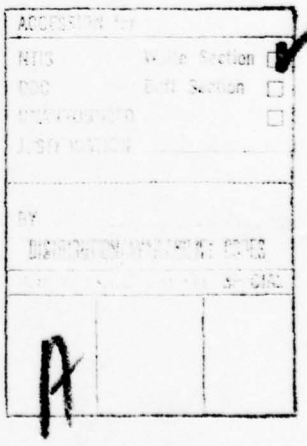
This Thesis, by Gordon K, Powell, is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the thesis requirement for the degree of Master of Science.

Gordon E. Stokes
Gordon Stokes, Committee Chairman

Bill Hays
Bill Hays, Committee Member

July 4, 1978
Date

Theodore A. Norman
Theodore A. Norman, Department Chairman



78 12 26 138

ACKNOWLEDGEMENTS

I extend my thanks and appreciation to Professor Gordon E. Stokes for his guidance, encouragement, and criticism in the development of this thesis.

I also express my appreciation to my wife, Stephaney, for her support and patience through the long hours away from home.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF GRAPHS	viii
Chapter	
1. INTRODUCTION	1
Statement of the Problem	1
Problem Justification	3
Problem Limitations	3
2 DATA DICTIONARY	6
Logical Representation	9
Physical Representation	20
3 DISTRIBUTION	25
Central	25
Fully Redundant	26
Partitioned	26
Partially Replicated	27
4. SIMULATION	32
Disk Access Time	32
Transmission Time	36
Limitations	36
Centralized Data Dictionary	39

Chapter Page

Fully Redundant Data Dictionary. . . . 40
Partitioned Data Dictionary. 41
Partially Replicated Dictionary. . . . 42
Results Analyzed 42
Further Considerations 45

5. DISTRIBUTION PROBLEMS. 50
Central. 50
Fully Redundant. 50
Partitioned. 51
Partially Replicated 51
Other Problems 52
Recommendation 55
Summary. 56

REFERENCES. 58

APPENDIXES

A. USER VIEW. 61
B. IMAGE SCHEMA. 68
C. SIMULATION PROGRAMS 77

LIST OF TABLES

Table		Page
2.1	Canonical schema design procedure.	14
4.1	Number of disk accesses to perform the desired operation for the specific entity	35
4.2	Number of milliseconds to perform the specific operation.	38

LIST OF FIGURES

Figure		Page
1.1	Horizontal and vertical distributed systems.	5
2.1	Relationships of data dictionary entities.	8
2.2	Activity of an Integrated Data Dictionary	10
2.3	Simple user's view of dictionary data . .	10
2.4	Canonical schema.	15
2.5	CODASYL representation of schema.	17
2.6	DL/I representation of schema	18
2.7	Relational representation of schema . . .	19
3.1	Central data dictionary configuration . .	28
3.2	Fully redundant data dictionary configuration	29
3.3	Partitioned data dictionary configuration.	30
3.4	Partially replicated data dictionary configuration.	31
4.1	Partial Image schema diagram.	33

LIST OF GRAPHS

Graph		Page
4.1	Centralized configuration response curve. First fraction represents updates and the second fraction represents inserts/deletes.	46
4.2	Fully redundant configuration response curve. First fraction represents updates and the second fraction represents inserts/deletes.	47
4.3	Partitioned and partially replicated configurations response curve with 20% messages remote. First fraction represents updates and the second fraction represents inserts/deletes.	48
4.4	Partitioned and partially replicated configurations response curve with 80% messages remote. First fraction represents updates and the second fraction represents inserts/deletes.	49

Chapter 1

INTRODUCTION

Statement of the Problem

Large computer systems have historically been centered around a single centrally located computer system. Users have been serviced through the central system regardless of their local needs or geographical location. More recently the users have been effective in bringing about a recognition that their problems might be better solved with distributed computer power because they as a group are distributed. Although there is still argument for servicing distributed users with a central single computer system, there is now a trend toward systems in which the information processing and storage functions are distributed among several computers. With this distribution there has been increased interest in the use of distributed database systems. The reason for this interest lies in the fact that distributed database systems provide a solution to some very real problems of a geographically distributed

organization. These organizations must maintain a unified information-sharing and processing system.

The computing industry is becoming more and more cognizant of the vital corporate resource--data. A problem shared by both centralized and distributed database systems is that of keeping track of what is in the database and where it might be located. The best and most recently developed tool for assisting a corporation in the location and use of company data is the data dictionary.

The data dictionary is a facility which contains defining information about the data held in an information database. Data dictionaries have taken many forms from single card files to large, complex computer stored and accessed files. The last few years have seen the growth of data dictionaries used to assist in the management and control of large database systems running on centralized systems. In as much as data dictionaries in this environment become very large, a data dictionary is a database about the database [1,2,3]. As computer systems and databases are distributed it will become essential to understand the use of the data dictionary in the distributed environment.

The purpose of this thesis is to develop a data dictionary description for use in a distributed integrated database system.

Problem Justification

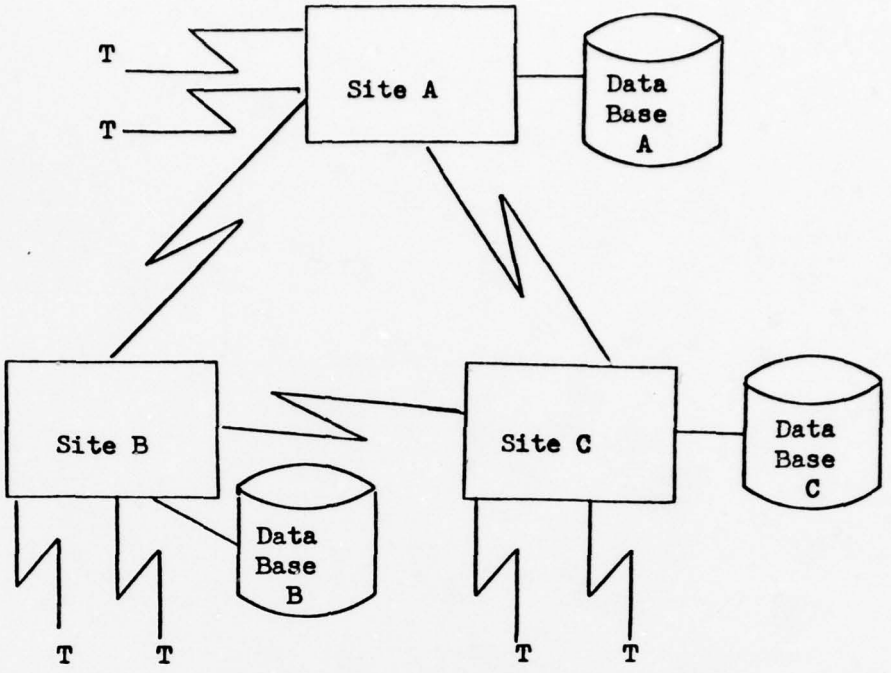
At the present time there are approximately seven marketed data dictionary systems. Only one of these systems is termed fully integrated [4]. The other six are referred to as free-standing. According to Lefkovits [5], a free-standing dictionary system is unknown to the operating system, language processors, or the Database Management System (DBMS). On the other hand, the DBMS, language processors, and operating system are fully aware of the existence of an integrated data dictionary system. In addition, there is no evidence in the literature of a data dictionary developed for a distributed system, especially an integrated data dictionary.

Problem Limitations

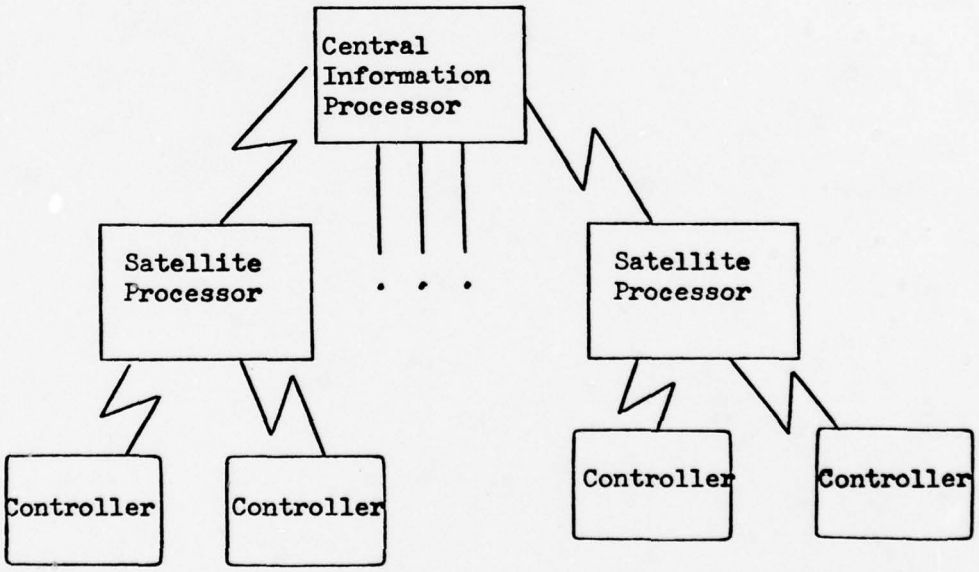
Distribution of a database takes on several forms and in this thesis will thus be limited to horizontal distribution. In a horizontal distribution the computers may be physically different and of unlike capacity and power, but they are logically equal with

regards to a hierarchical structure. They interconnect on the same level. In a vertical distribution, the computers are hierarchically distributed with the smaller computers being more specifically task oriented. (See figure 1.1.) In addition, the discussion will be limited to homogeneous systems. These systems have the same hardware. In most cases the application is applicable to heterogeneous networks. In these networks the hardware may be very different from installation to installation. It is not the intent of this thesis to provide an implementation of a distributed dictionary system. Rather, the thesis will deal with the description of the data dictionary and the various ways the dictionary may be distributed in the system.

The use and definition of a data dictionary will be proposed with particular attention to the specific features and content. Next the logical representation will be discussed followed by a discussion of five possible physical representations for the database. Four proposed organizations will be presented listing their advantages and disadvantages. These four configurations have been simulated and the results will be presented and discussed. Last of all, recommendations will be made and conclusions drawn.



Horizontal Distribution



Vertical Distribution

Figure 1.1 Horizontal and Vertical Distribution

Chapter 2

DATA DICTIONARY

The purposes of a data dictionary are to provide a means of control on how the data are to be used, to provide a more complete form of documentation, and maintain certain characteristics about each entity. There is not just one way to accomplish this. In fact, certain of the attributes of the dictionary may be determined by the specific system being used or the needs of the users. Generally speaking, however, the data dictionary will contain entities such as Data Files, Data Fields, Programs, Databases, Systems, Users, Departments, Security Levels, and relationships established between these entities [6,7,8].

Each of these entities will have associated with it the attributes or characteristics that have been determined important. These will be determined by the Database Administrator after conferring with the users of the system. Lefkovits [9] suggests that the domain of the data dictionary consists of three entity types. First, a data entity, which includes data items (elements), groups (data aggregates), and files.

Second, a processing entity, which includes system modules, programs, subprograms, and systems; and databases. Third, a usage entity, such as users, departments, or organizatons. Cullinane [10] suggests that there are seven major categories. These include users and systems, programs, elements (consisting of groups and items), records, files, classes, and attributes. Users include projects, departments, and individuals. Systems include subsystems and systems; and programs include programs, subprograms and modules. Classes deal with security of the entity, language, mode, frequency, and privacy. Attributes will be discussed at length below. Figure 2.1 shows how these entities are related.

The attributes of a data item (element) might include: 1. Official name--the name by which it is known to the dictionary, 2. Synonym name--the name to which it is commonly referred, 3. Alias names--other names besides the synonym name to which it may be referred, 4. Description, 5. Usage--how the item is used. It may be a numeric field, alphanumeric, or another form, 6. Program Language Names--COBOL, Fortran, Assembly, etc., 7. Version--a version number of the dictionary, 8. Security--security level needed

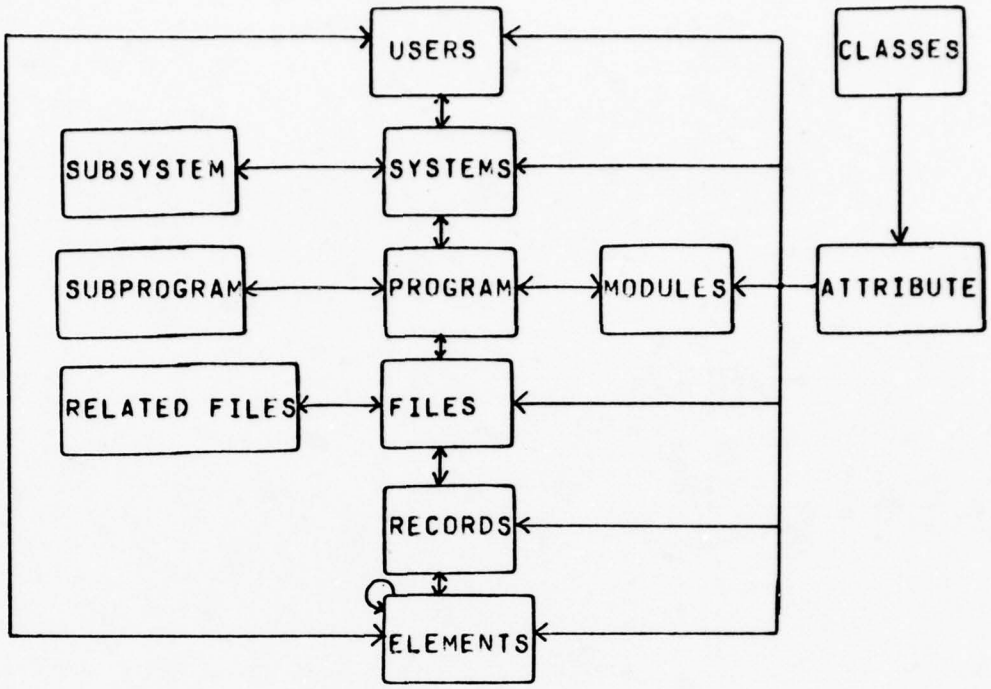


Figure 2.1 Relationships between Data Dictionary entities

to access data instances, and 9. Classification--batch, on-line, etc. Other attributes considered essential could also be included. A data group might include other attributes such as Contents--names of item(s) and/or group(s) that make up the group, and Position--the alignment of the parts within the group. A record would include other attributes such as Access and Primary Key. Files would specify Structure--sequential, hashed, etc., and Sorting

Order--ascending, descending, or most frequently accessed. Programs would include attributes like Source Language, Programmer--the person responsible for the program, and Characteristics.

Logical Representation

Once the attributes essential to this system have been determined, user views of the data or subschemas must be drawn up. This means how the data is to be viewed by the specific users of the data dictionary system. The users for an integrated data dictionary system consist of the following: 1. Terminal users making requests on-line, 2. Batch users generating reports, and 3. Programs compiling and assembling. Figure 2.2 represents this kind of activity. The dictionary must be able to interact with all three user classes. Figure 2.3 represents a simple user's view that might be supported by the data dictionary.

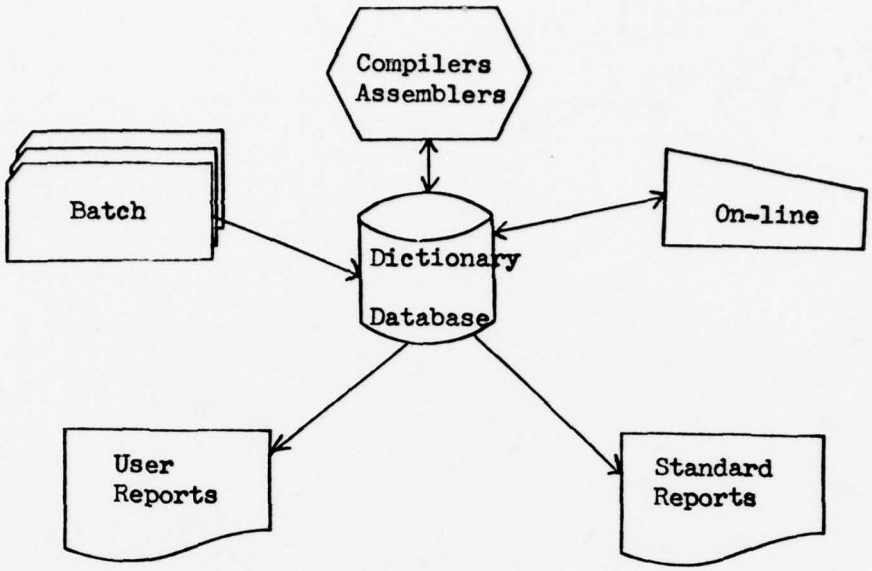


Figure 2.2 Activity of an Integrated Data Dictionary

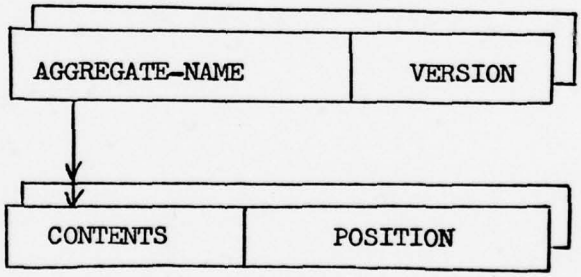


Figure 2.3 Simple user's view of dictionary data

Once all the user views of the data in the data dictionary have been determined then a logical view of the entire system can be developed. Appendix A represents all the user views that are supported in the data dictionary in this thesis. Each individual subschema is represented in the over-all logical view of the data dictionary.

After the logical view of the data dictionary is developed the following entities and information are available in the dictionary:

1. A dictionary name--this is the entity name determined by the database administrator. Entities consist of databases, data elements, data records, files, programs, and systems.

2. A synonym name--the name the entity is commonly referred to by. It may be the same as the dictionary name.

3. An English language description of the entity.

4. How the item is used: numeric, alphanumeric, etc.

5. The number of digits or characters used in representing data elements.

6. A version number for the dictionary specifications of the entity concerned.

7. The security level necessary to access instances of the entity.

8. The name of the entity as it is used in the various programming languages, such as, Fortran and COBOL.

9. The date the entity was created.

10. The date the entity was last updated.

11. The classification of the entity, whether it is batch, on-line, test, or production.

12. How the records are accessed; sequential, random, etc.

13. The primary key used to access the records.

14. How the file is structured; i.e. indexed sequential, hashed, binary tree, etc.

15. The contents of each file, record, and aggregate, and how the contents are aligned or positioned in the entity.

16. The source language of the programs; i.e. BASIC, APL, etc.

17. The name of the programmer responsible for the program.

18. The characteristics of the program; i.e. files used, output generated, and processing time required.

19. The names of the various users that use the dictionary.

20. The name of the database administrator and information about him that will help in contacting him.

21. Alias names--other names entities may be referred to by.

This information allows a user, who is writing a new program, to ascertain whether specific items already exist and, if so, where they are found. For example, list all the programs that use the data aggregate

home-address, printing source code, COBOL-name, and program description. If the user is not sure of the correct name then he may ask for the dictionary name and supply an alias name. A department may be considering changing the length of the inventory number from six digits to ten digits. A report can be produced listing all the programs that would be affected by this change. A system programmer may need to contact all users using the Heap sort routine because it is not sorting correctly. This information is contained in the database and can be obtained on demand. The dictionary not only serves as a tremendous source of information, but it also provides control and standardization of data names across systems, programs, and users.

The attributes and subschemas were organized and, following a process described by Martin , were reduced to a canonical schema of the dictionary database [11]. Table 2.1 summerizes the steps presented by Martin to arrive at a canonical form and figure 2.4 represents the canonical schema for this data dictionary. A canonical schema is a model of data which represents the inherent data structure but is independent of any application. It is also independent of the software or hardware which is employed in representing and using the data [12].

1. Take the first user's view of the data and draw it in the form of a bubble chart.

2. Take the next user's view, represent it as above and merge it with the first user's view.

3. From the resulting graph distinguish between attribute nodes and primary keys. Mark the primary keys.

4. For each association between keys, add the inverse if it is not already present.

5. Examine the association and remove any genuinely redundant associations.

6. Repeat steps 2-5 until all user views are merged into the graph.

7. Identify the root key.

8. Eliminate isolated attributes.

9. Modify the graph to eliminate intersecting attributes.

10. Redraw the data items into groups (records, tuples).

11. Identify all secondary keys.

12. The unconstrained "canonical" schema may now be represented in a particular software package.

13. Return to the original user views and make sure they are represented correctly in the canonical schema.

Table 2.1 Canonical schema design procedure according to Martin [11]

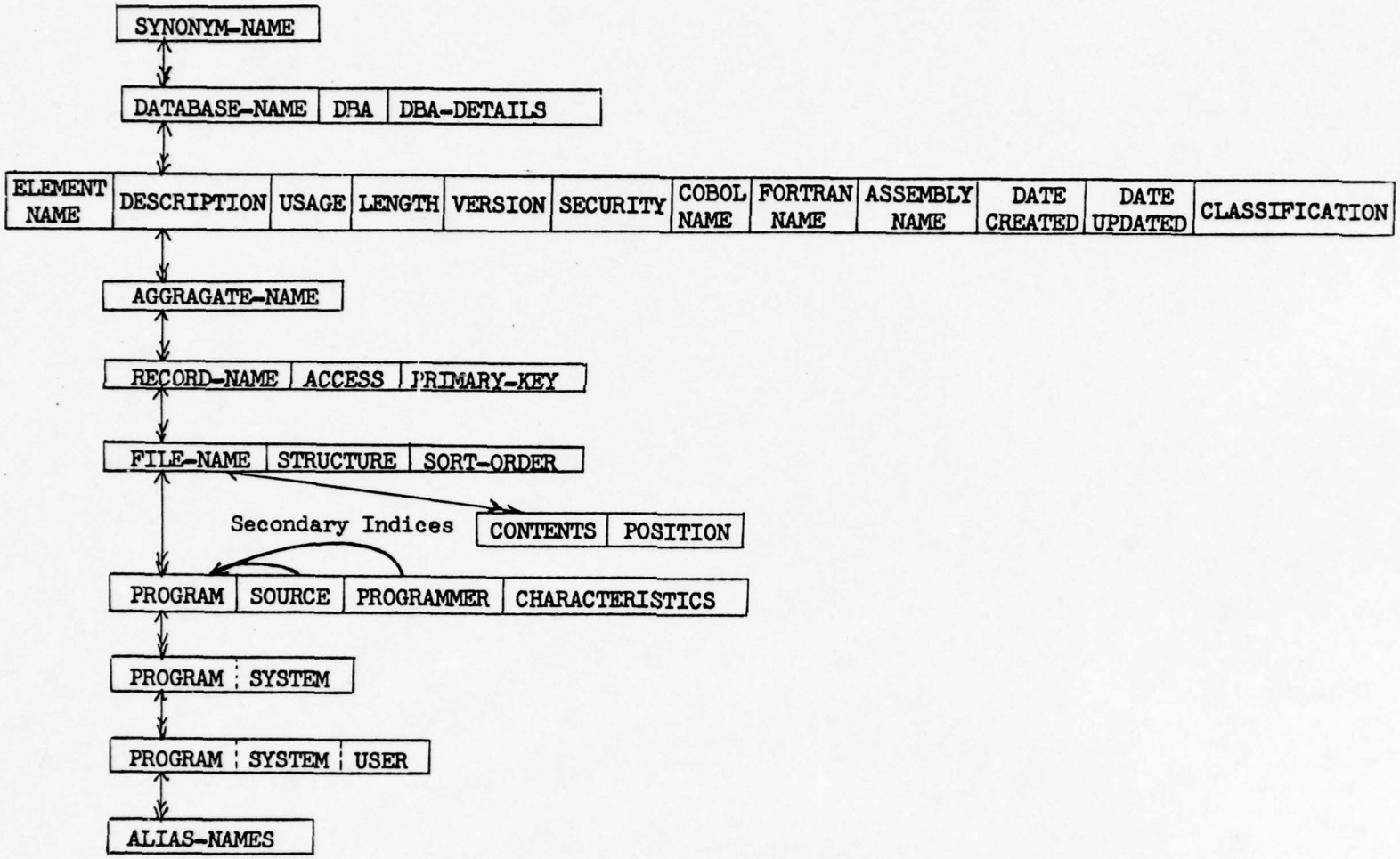


Figure 2.4 Canonical Schema

The canonical schema, when it is implemented, can be represented in CODASYL Data Base Task Group-based software, in IBM's DL/I-based software or as a relational database. When it is represented in an appropriate database management system, the canonized schema provides the best facility for future changes. Most future changes can be accommodated by an incremental growth of the canonical schema without massive restructuring [13]. In addition, the database administrator of any system can represent this canonical schema in the database management system of his choice. This applies equally to homogeneous and heterogeneous networks, which demonstrates the flexibility of this technique. The canonical schema of figure 2.4 is represented in CODASYL, DL/I, and relational, in figures 2.5, 2.6, and 2.7 respectively.

For the purpose of simulation, this canonical schema was implemented in Image, the DBMS for the HP 3000 version 2 at Brigham Young University as presented in Appendix B. The canonical schema is a minimal structure and not all software packages are able to represent this structure. Thus it may be necessary to deviate from the canonical schema by introducing redundancy. Such is the case with Image/3000.

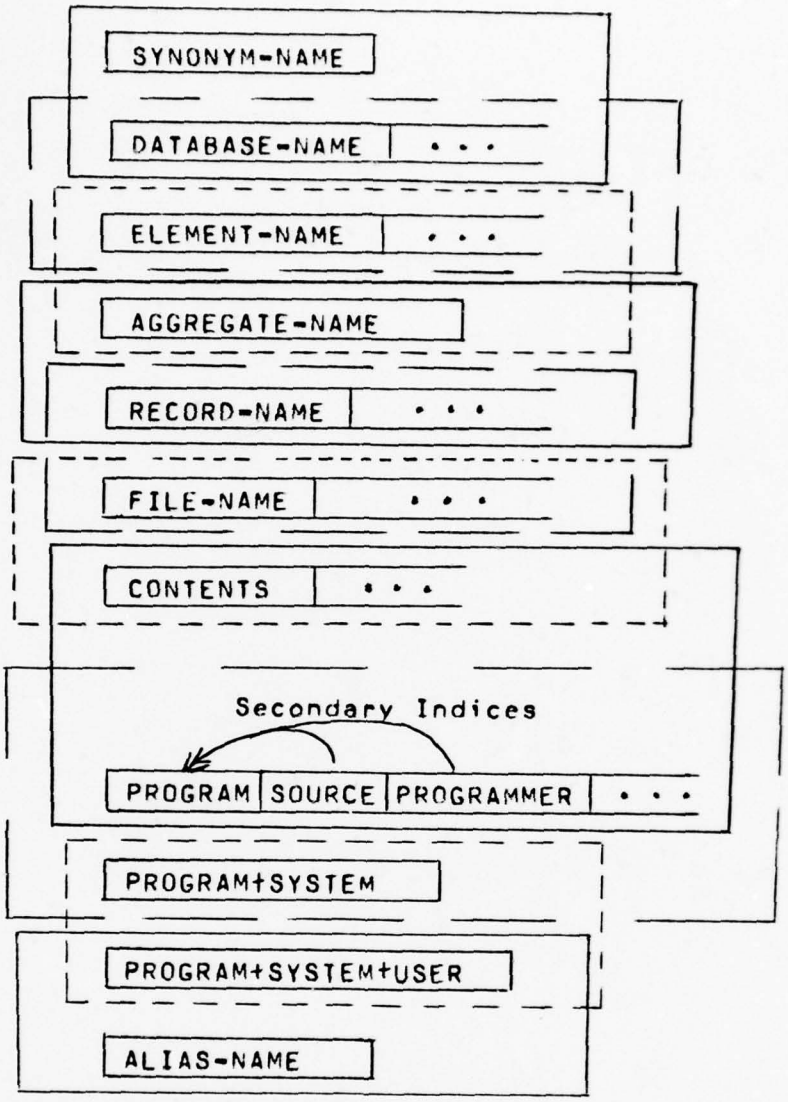


Figure 2.5 CODASYL Representation

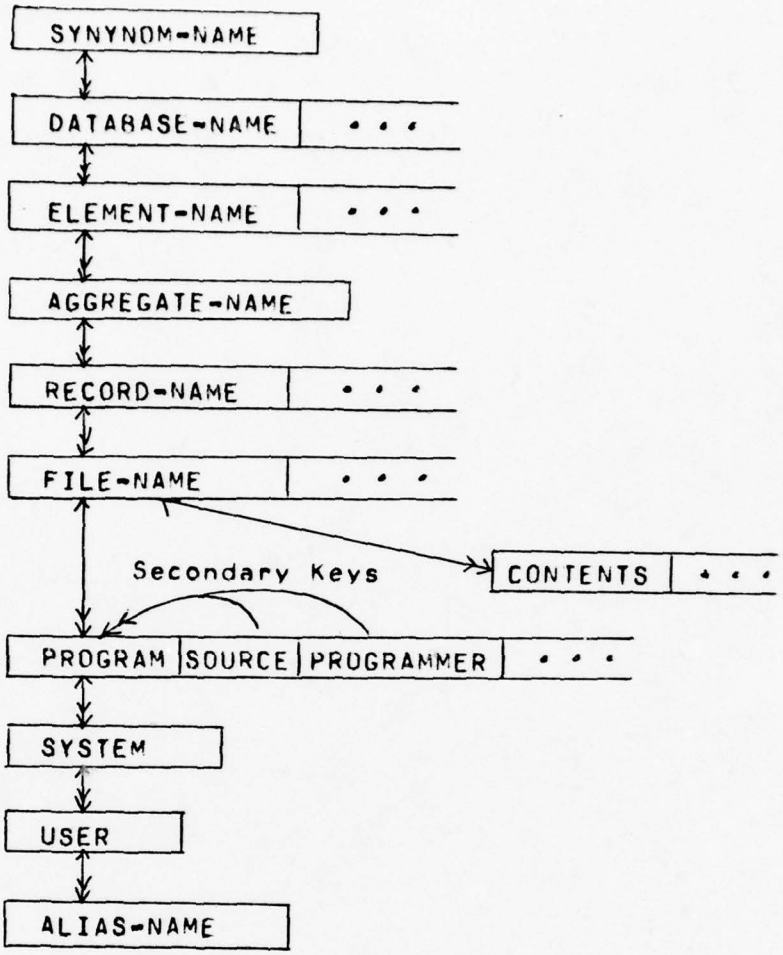


Figure 2.6 DL/I Representation

SYNONYM-NAME

DATABASE-NAME	. . .

ELEMENT-NAME	. . .

AGGREGATE-NAME

RECORD-NAME	. . .

FILE-NAME	. . .

CONTENTS	. . .

PROGRAM	SOURCE	PROGRAMMER	. . .

PROGRAM	SYSTEM

PROGRAM	SYSTEM	USER

ALIAS-NAME

Figure 2.7 Relational Representation

Physical Representation

Regardless of the logical representation used, numerous physical organizations exist. Care must be taken when determining the physical organization, since this will greatly affect machine performance. The selection of physical organization is largely determined by the need for operational efficiency, fast response times, and cost minimization.

Martin [14] suggests four steps to identify those areas of high usage and fast-response paths. First, mark all paths in the canonical schema which will be used on interactive systems and require fast response time. Second, determine the number of times a user path will be followed in a given amount of time. Third, estimate the length of each group. Fourth, for each one-to-many association estimate how many there will be. The results of this analysis may effect the choice of structure or result in a modification to the schema.

For an integrated data dictionary, response time and cost are of utmost importance and, therefore, may dictate random access over sequential. In addition, an organization should be selected that handles numerous insertions and deletions, and facilitates expansion or

growth. Other factors influencing the desired physical organization are blocking, data compaction, frequency of reference, clustered insertions, and multiple-key retrieval.

In order to handle the numerous user views in this database and multiple-key retrieval, a physical organization tailored toward a fully inverted database will be necessary. Five possible physical organizations are presented here.

First is an inverted list organization. The indexes contain the entire key as opposed to a key range which results in larger indexes but does not increase the number of entries in the indexes. It increases the resolving power of the index and the number of inquiries that can be answered.

Second is an inverted lists with indirect addressing. This organization is similar to the first except none of the secondary indexes have machine addresses. This means that when the data files are reorganized and the physical addresses are changed only the primary index needs to be rewritten. Also more questions can be answered using the indices instead of going to the data files because the full key is used instead of a truncated key.

Third is a bucket-resolved inverted list organization. In an attempt to reduce the size of the indices, the indexes point to the buckets that contain the values for that key. The size of the bucket directly affects the index savings. A disadvantage is that fewer queries can be answered in the indexes because keys may be abbreviated or truncated.

Fourth is a bucket-resolved inverted list organization represented by bit strings. The bit strings are used to represent the buckets the contents are located in. Bit string representation is more compact than other methods. The savings is also affected by the number and size of the buckets. If the data records are sorted so that like values for the indexes are in the same bucket, then the number of buckets searched will be reduced. Sorting increases the complexity of file maintenance and, therefore, is desirable only when files are not updated on-line.

Fifth is an organization with secondary keys removed from the data files into a chained index. In this method, the secondary keys are removed from the data records and stored in the indexes. This organization permits more queries to be answered without searching the data records. Chains are contained in the

indexes which facilitate quicker searching. Also maintenance is simplified because it is handled in the smaller index files. A disadvantage of this method is that if the pointers are damaged it might result in a loss of the data record associated with that secondary key. To eliminate this potential problem the secondary keys could be carried in the data records as well.

Another determining factor that affects access time is how the indexes are searched. A number of techniques are used. These include: serial scan, block search, binary search, binary-tree search, balanced tree index, unbalanced tree index, algorithm index, hash index, look-aside buffer, and sequence set chain [15]. A hash index was determined to be the best technique to support a data dictionary database.

A hash index does not have the advantage of a pure hash because records may not be found in one seek. However, it allows records to be stored by some other order. Records could be stored sequentially by primary key or with their parents in a tree structure. The empty spaces in the buckets, typical of hashing techniques, are in the smaller indexes and not in the data storage. It is less wasteful than pure hashing and

handling overflows is less time consuming because they are in the indexes.

The data dictionary database will be substantial in size and the organization that will provide efficient use of storage, speed in access, and conform the best to the specific unit characteristics should be used.

The data dictionary described here, derived from the various user views, contains important information about every data item, aggregate, record, file, program, system, user, and database an installation may want. It is designed to handle practically any request imaginable. For example: List all the data elements used in the payroll system; List all the programs written in Fortran and containing the data aggregate Date; List all the alias names for the data item social-security; List the contents of all records in the student system and using Name as primary key; List all the programs John Doe is responsible for, printing program name, source code, characteristics, date created, date updated; etc.

Chapter 3

DISTRIBUTION

The data dictionary is meta-data about the database and generally speaking, will approach the size of a corporation database. In as much as the dictionary is a repository for an integrated database which is geographically distributed, the proper distribution of the dictionary for fast response is essential. Four different distributions are presented. They are: 1. Central data dictionary; 2. Fully redundant data dictionary; 3. Partitioned data dictionary; and 4. Partially replicated data dictionary.

Central

The centralized data dictionary has only one complete data dictionary. (See figure 3.1.) In order for any of the other installations in the network to access the dictionary, it must obtain this information through data communication lines. This means that there must be at least one data communication line between the installation with the dictionary and every other installation. Those installations which require a lot

of information from the dictionary may require more than one data communication line.

Fully Redundant

The fully redundant system requires that each installation has a complete data dictionary. (See figure 3.2.) No accessing is necessary beyond the local installation. The data communication lines are needed to insure that the dictionaries are identical and reflect the latest updates.

Partitioned

The partitioned data dictionary system is segmented into physically disjoint sections. (See figure 3.3.) Each of these Local Dictionary Directories (LDD) is geographically distributed in the network at the installations that require this information. The separate sections, because of their interrelationships, logically form a single database. Any request for non-local data results in a remote message broadcast throughout the network to retrieve the desired data.

Partially Replicated

In the partially replicated data dictionary the data dictionary consists of two parts: the Global Dictionary Directory (GDD) and the Local Dictionary Directory (LDD). (See figure 3.4.) Each Global Dictionary Directory contains a complete list of all the entries in the dictionary, a pointer to the Local Dictionary Directory which the details are found in, and the security level required to access instances of the entity. The Local Dictionary Directory is the partitioned dictionary database for that installation. A request for non-local data will generate a remote message to the specific installation where the data is stored.

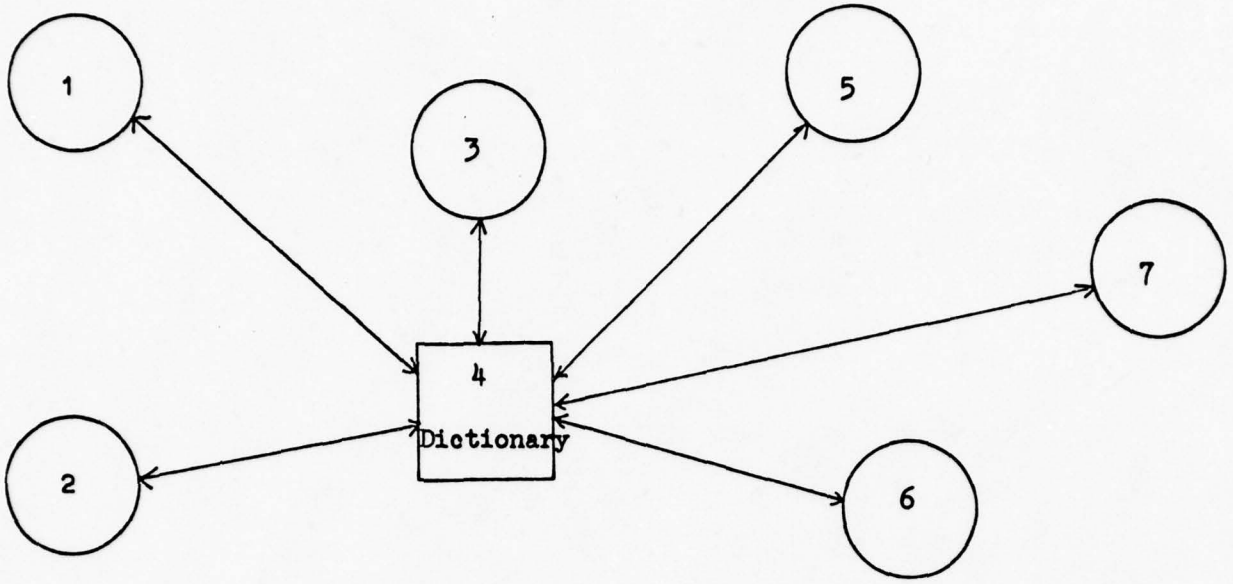


Figure 3.1 Central data dictionary

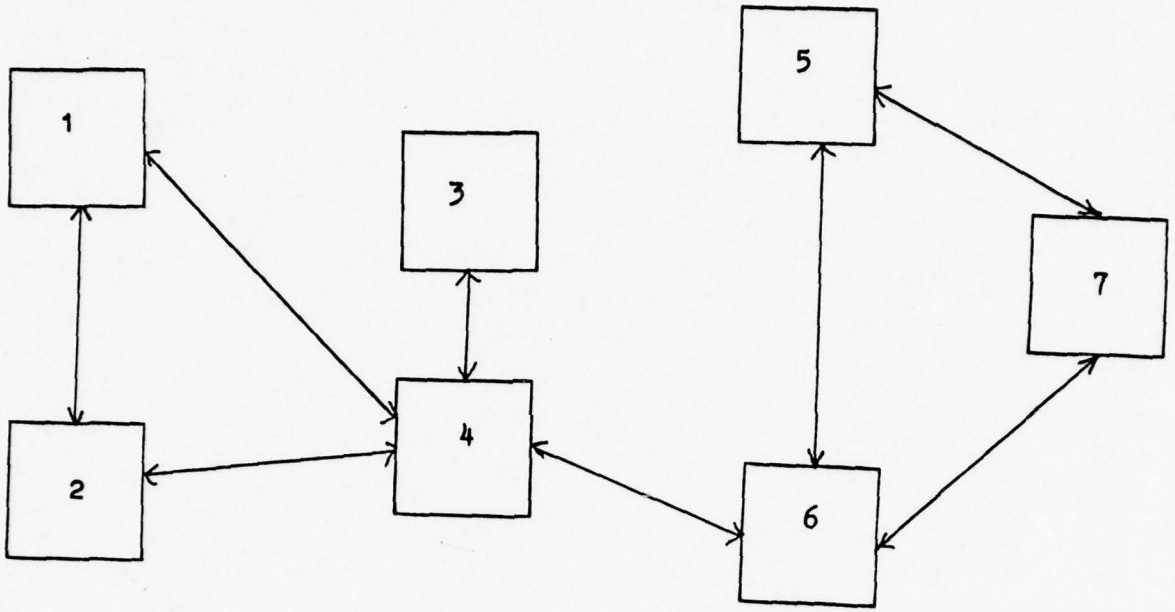


Figure 3.2 Fully redundant data dictionary with a dictionary at each installation

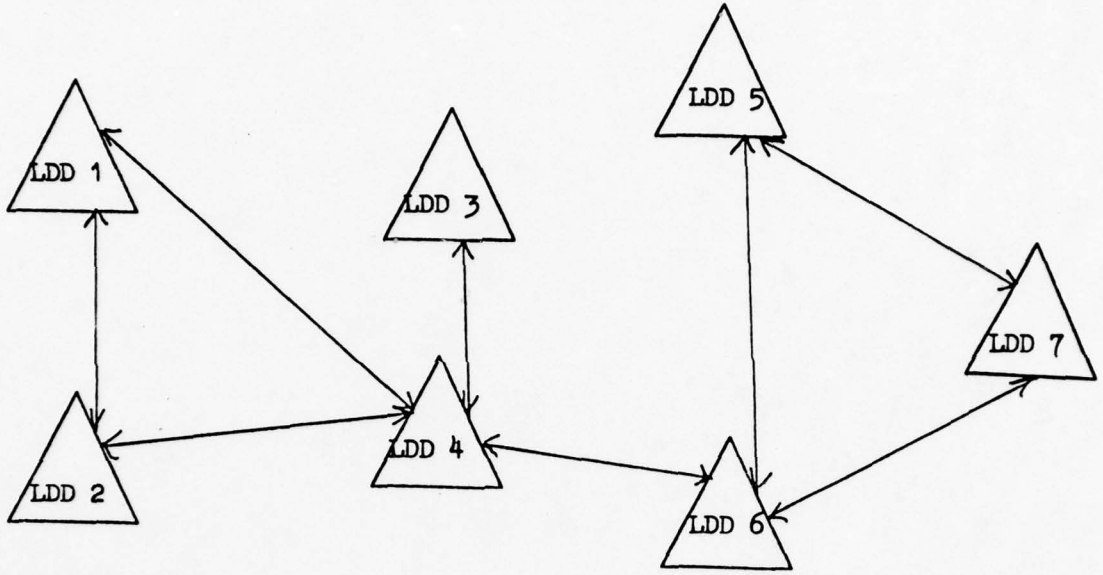


Figure 3.3 Partitioned data dictionary
with Local Dictionary Directory

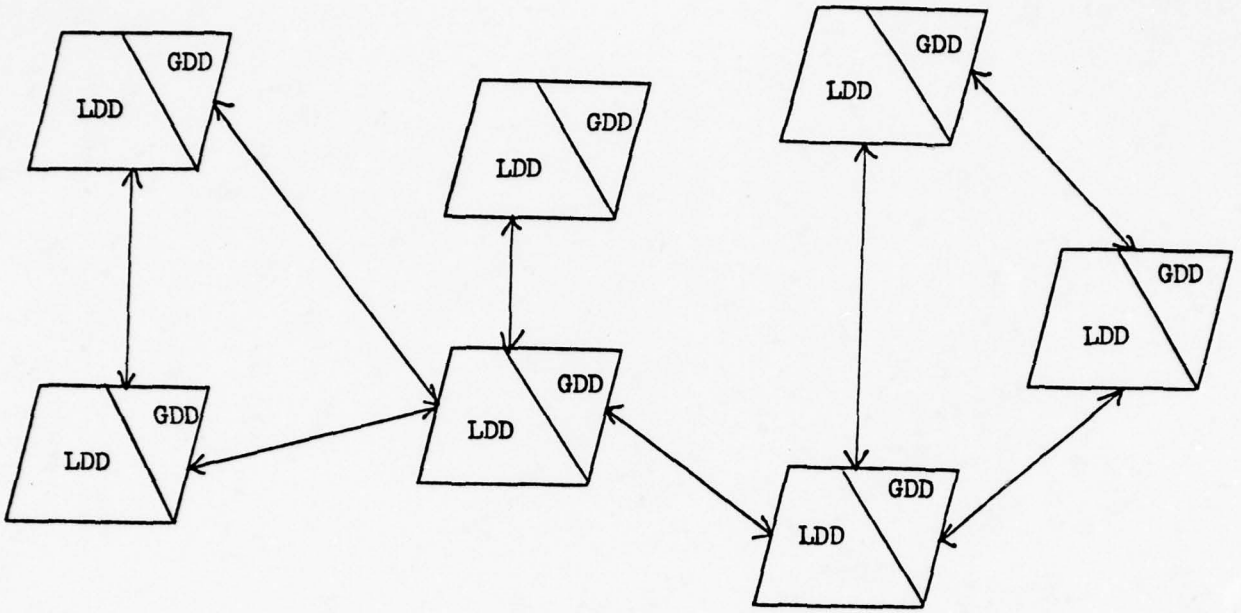


Figure 3.4 Partially Replicated data dictionary with Global Dictionary Directory (GDD) and Local Dictionary Directory (LDD)

Chapter 4

SIMULATION

In order to simulate these proposed configurations, (See Appendix C), it was necessary to answer a number of questions. For instance, how much time is needed to perform a disk access, how many disk accesses are required to insert a complete entry into the data dictionary, to delete an entry, and to modify one. In addition, it was necessary to determine how much time was required to send an inquiry to a remote installation.

Disk Access Time

The time to perform a disk access was determined from observing a large number of I/O operations on the system. The time ranged from 40 msec to 80 msec. Fifty milliseconds was used in the simulations. To determine the number of disk accesses for updates and retrievals, the Image schema for the data dictionary database was analyzed for disk access operations. The Image schema structure provides two types of sets: master and

detail. Figure 4.1 depicts a partial representation of the canonical schema of the data dictionary in Image diagram form.

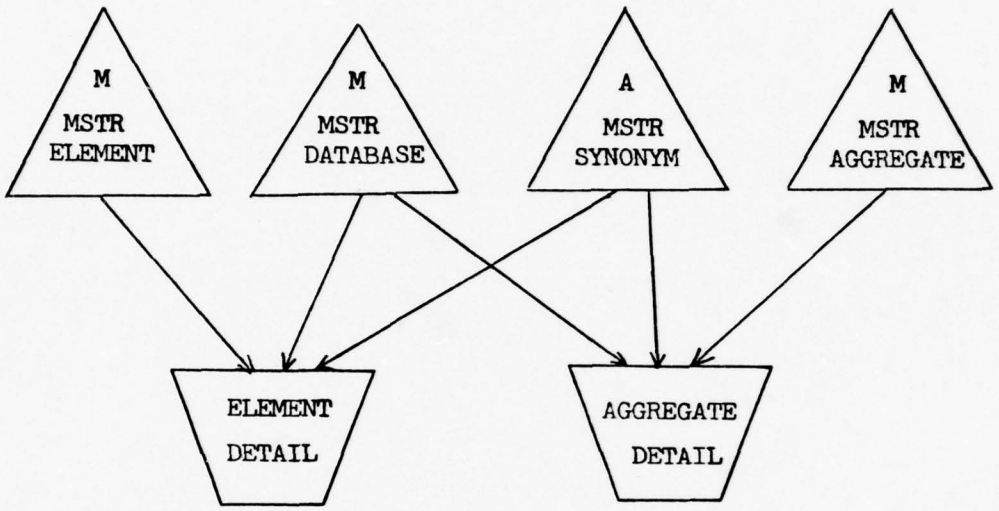


Figure 4.1 Partial Image schema diagram

A master file is basically an inverted file and may reference from one to sixteen detail sets. A detail set may have from one to sixteen search items. Image uses a hash function to store the entries in the master file and then chains the pointers to the detail occurrences. Data is frequently stored redundantly.[16]

The simulated dictionary consisted of: four data entities--elements, aggregates, records and files; three process entities--programs, systems, and databases; and one user entity--individuals. A number of inquiries from each of these entities were checked against the Image schema representing deletes, inserts, modifies, and retrievals. These inquiries were to determine the average number of disk accesses that could be expected in each case. The minimum number and maximum number of disk accesses were also determined for each entity.

Using the amount specified in the capacity attribute of the Image schema, a weighted average for the minimum and maximum disk accesses for an insert were computed. Both the minimum and maximum values were then multiplied by the 50 msec disk access time previously determined, to provide a minimum and maximum access time for inserts. This procedure was repeated for deletes, modifying updates, and retrievals. Table 4.1 shows the results that were obtained.

INSERTS/ DELETES	MIN DISK ACCESS	MAX DISK ACCESS	NUMBER OF ENTRIES
ELEMENT	9	12	100000
AGGREGATE	10	13	50000
RECORD	10	13	10000
FILE	10	13	10000
PROGRAM	10	13	1000
SYSTEM	4	5	100
DATABASE	6	6	10
USER	5	6	1000
WEIGHTED AVERAGE	9	12	
MODIFY WEIGHTED AVERAGE	3	6	
RETRIEVE WEIGHTED AVERAGE	3	4	

Table 4.1 Number of disk accesses to perform the desired operation for the specific entity.

Transmission Time

To determine the time to transmit an inquiry to a remote site, the traffic and message delays of the Advanced Research Project Agency (ARPA) network were examined [17]. For a single full packet message (1000 bits) to be sent between two close sites and be acknowledged required 50 msec. Acknowledge means a notification that the message sent was received. For a message to be sent cross-country involving five locations required 190 msec. Each site has a store-and-forward Interface Message Processor (IMP). The time required to process a store-and-forward packet is about 0.35 msec. An 8-packet message sent between two close sites required 195 msec. These results were obtained while the network was operating with a light load.

Limitations

Certain limitations were imposed to keep the simulation simple. The network consisted of seven installations all of which were up and available to process messages at all times. No attempt was made to simulate failed nodes. However, the problems associated with down nodes in a network will be discussed later.

The messages simulated were single full packets. Each installation had one data communication line operating in a half-duplex mode. This, of course, is not the ideal configuration in a real world situation, but it allowed the simulations to proceed without introducing a great deal of complexity. Ideally there may be one or more data communication lines with 64 logical paths operating in a half-duplex mode.

Response time was considered to be the time from entering the request until an acceptable response was received. The response may be either the desired data or a failure to find the data. In some cases one request may trigger a search of the network for an answer. In this case the response was not sent until a response was received from all the sites that were queried. This may seem like a severe limitation, but the system having the information will take longer to respond and therefore, the negative responses will be completed sooner. Since each configuration followed the same limitations no descrepancies were introduced.

The processing time for an entry was considered to be the time computed necessary to perform the disk accesses for that entity activity. Actual processing

time, aside from disk accessing was considered minimal and, therefore, was not included.

The simulations were run on the DEC-10 system using the General Purpose Simulation System (GPSS) language. The advance block was used to simulate communication/transmission delays and the processing of a dictionary inquiry. The times used to simulate these delays are represented in table 4.2. The time to enter or delete an entry from the GDD was two disk accesses or 100 msec. A randomness was added which would vary this 100 msec time by 25 msec, but maintain the 100 msec average.

	MIN	AVE	MAX
LOCAL TRANSMISSION	3	10	17
REMOTE TRANSMISSION	50	120	190
RETRIEVAL	150	175	200
INSERT/DELETE	450	525	600
MODIFY	150	225	300
ENTER/DELETE FROM GDD	75	100	125
GDD LOOK-UP	90	100	110
LDD LOOK-UP	110	120	130

Table 4.2 Number of milliseconds to perform the specific operation

In order to consider each configuration in its best situation and worst situation four parameters were used. These parameters consisted of: 1. the time between requests; 2. remote versus local; 3. update versus retrieval; and 4. insert/delete versus modify. Three time intervals were considered: 1. 500 msec; 2. 1000 msec; and 3. 5000 msec. Since there were seven installations inputting messages during the same time interval, the average for having a message in the network was actually one-seventh the time interval. In other words, there were an average of seven messages in the network at the same time.

Centralized Data Dictionary

The central data dictionary was located at installation four, and therefore local transmission was considered practically instantaneous but was delayed an average of 10 msec. Inputs to the dictionary would therefore occur randomly from the seven different locations. The dictionary activity, whether it was an insert, delete, modify, or retrieval, was processed serially. The response was generated following the completion of the query. If a message was in the process of being transmitted from that same site for another request, the response would be queued up until

the line was free before proceeding. In this configuration there was no need to include a local versus remote parameter because there was only one dictionary, and for all sites except number four, the message would be remote.

Fully Redundant Data Dictionary

Since each installation had a complete dictionary all messages could be answered from the local site. Random inputs were generated from each of the seven locations with equivalent interarrival rates. Each message would have a local transmission delay. Then if the request was to retrieve, the look-up would be performed and the results transmitted back to the originator. Again if the data transmission lines were in use, the seizing message would have to wait until the lines were free. If the request was to update the dictionary either for an insert, delete, or modify an existing entry, while the update was in progress at the local site, six remote messages would be sent out to the other sites to perform the desired request. A response would not be sent to the originator until all remote and local messages had been completed. It thus resulted in slow response times on certain occasions.

Partitioned Data Dictionary

In the partitioned data dictionary messages originated from seven locations randomly. Each averaged the same specified number of milliseconds and incurred a local transmission delay. In this configuration messages could be for either the local dictionary or a remote one. A local message could either be a retrieval or an update. In either case the desired function would be performed and a response generated to the originator as soon as the line was free. If the message was for non-local information, six remote requests would be broadcast throughout the network. One additional constraint was imposed in this network. Updates could only be done against the local dictionary. No one could update another dictionary. Therefore, all remote messages were retrievals only. The remote site that had the information was determined randomly and the retrieval was then performed. The response to the originator included the remote transmissions to and from the other sites, plus the dictionary look-up and the local transmission delays.

Partially Replicated Dictionary

In the partially replicated dictionary each of the messages incurred a local transmission delay and was also generated randomly. If the message was against the local dictionary, then a further breakdown was necessary. If it was a request for some information, then the data was retrieved and a response generated. For a modify update the modification was made and a response sent. For an insert or delete, six remote updates to the other global dictionary directories were generated. A response was generated upon completion of all the updates. If the message was for retrieving remote information the specific installation was known and the data retrieved. All remote transmissions incurred a time delay.

Results Analyzed

The results of the test simulations were very revealing but in most cases expected. Graph 4.1 reports the results of running 1000 messages, at five second intervals, against the general data dictionary, configuration A. As was expected the response

time was reasonable as long as there were few updates. But as the updates increased in frequency the response time slowed down substantially.

Graph 4.2 shows the results of running in the fully redundant data dictionary, configuration B. The results are similar to those in graph 4.1, except when more than half of the messages were updates, then the lines became swamped with multiple update traffic. As long as updates comprised a small percentage of the messages, then configuration B was better than A. However, when updates exceed 50%, configuration A became superior to B.

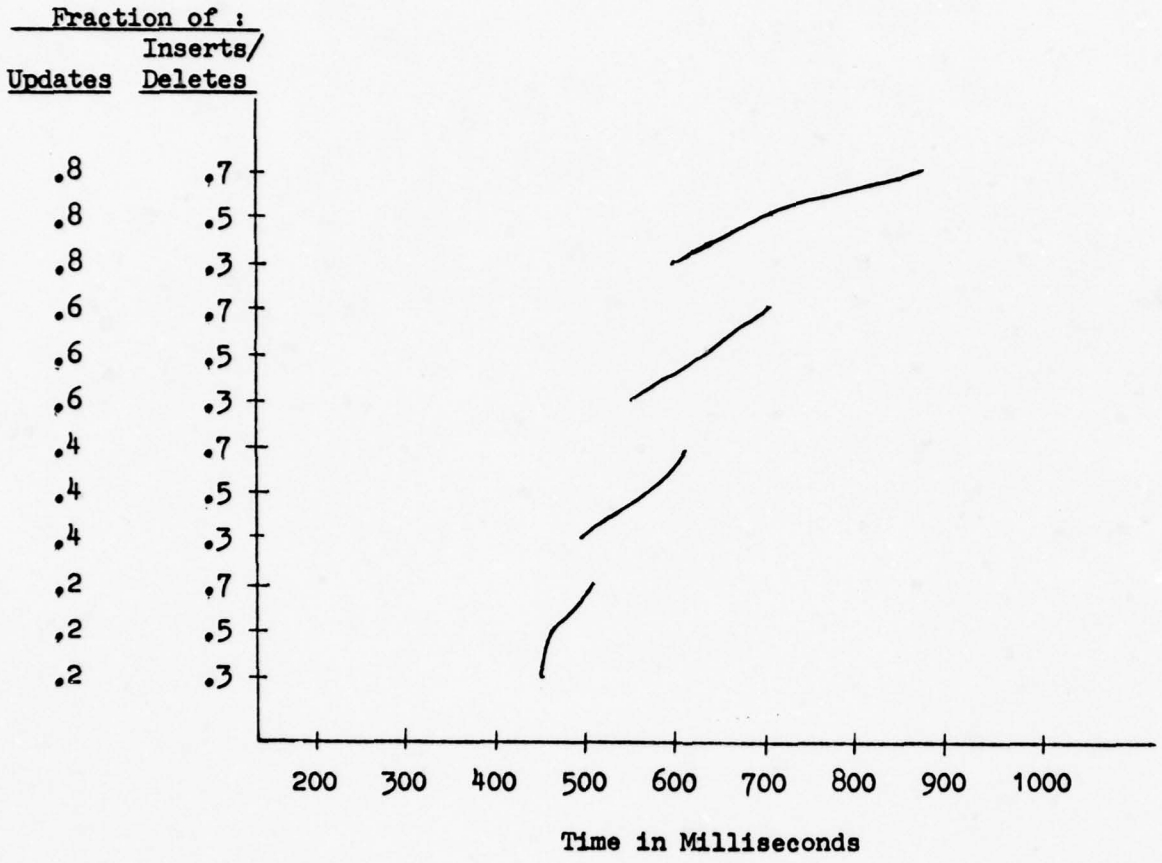
The partitioned data dictionary, configuration C, and the partially replicated data dictionary, configuration D were first tested under the condition that 20% of the message traffic was for remote installations. The amount of time to search the LDD in configuration C was set approximately equal to the time it took to find the installation at which an entry resided in the GDD. Graph 4.3 shows the results of these runs. Notice that both C and D were faster than A or B; especially as updates became the bulk of the traffic. Also notice that C and D were very close, alternating back and forth. The reason for this was

that for every update resulting in an insert or delete in configuration D, seven GDD's were modified. As the updates increased in number the spread became more distinct.

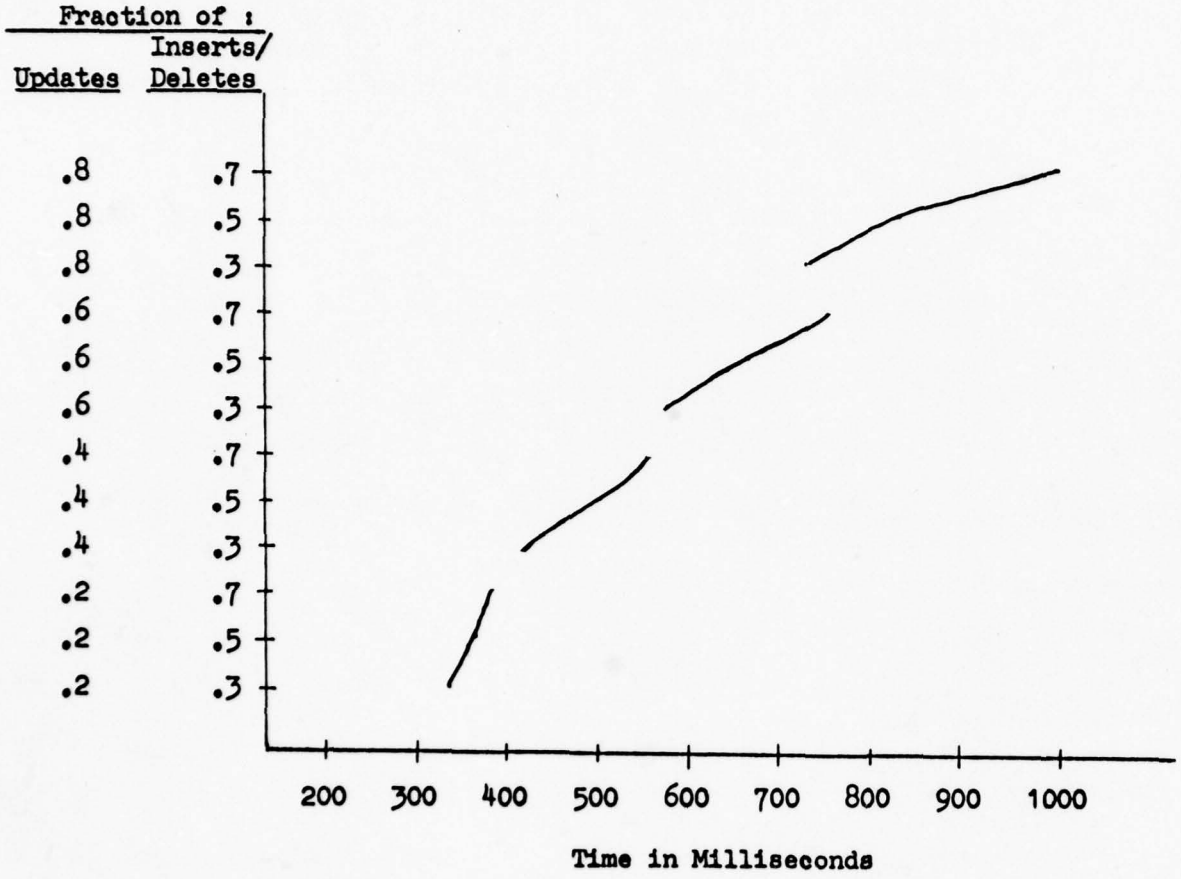
Switching to the other end of the spectrum, we find a definite rise in response time for both configurations. Graph 4.4 shows the results. While the number of updates were small, configuration C and D were not as good as A or B. This is because 80% of the message traffic was for non-local data. This resulted in a search of the LDD in configuration C, and a search of the GDD in configuration D followed by remote transmissions. As the number of updates increased, C and D steadily moved away from A and B, showed an improved response time. At this end of the scale D showed a markedly better response time than C. Also notice that when updating exceeded 50%, D was also better than A or B. Similar results were obtained when the four configurations were run with message intervals at one second and one half second. The difference was as the interval became smaller the response time increased.

Further Considerations

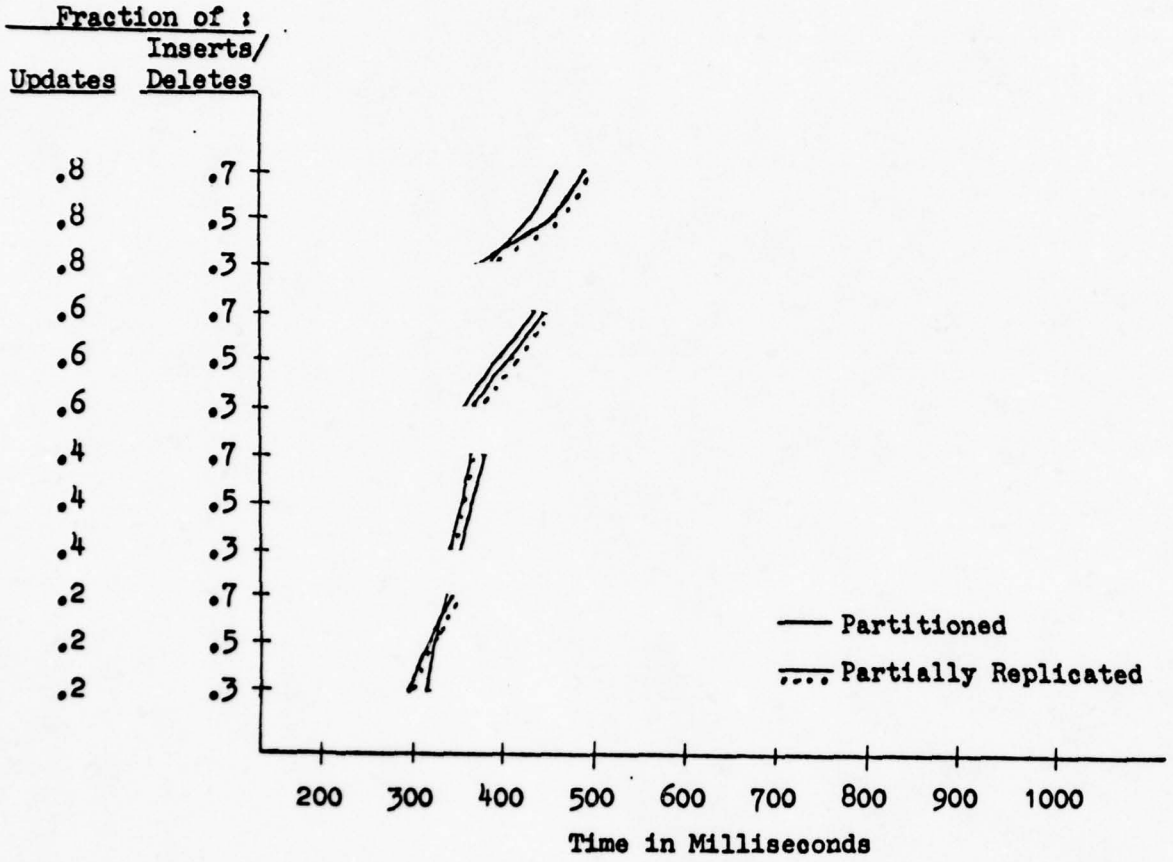
The response time for configuration D could have been improved in two ways. First, if the look-up time for the GDD was faster than the LDD look-up in configuration C, which it should be, this would result in an improvement of response time. Second, when the GDD was referenced, if it pointed to the LDD where the data were contained, then a second search of the GDD would not be necessary. This would result in a small increase in update time, but would tremendously reduce retrieval time.



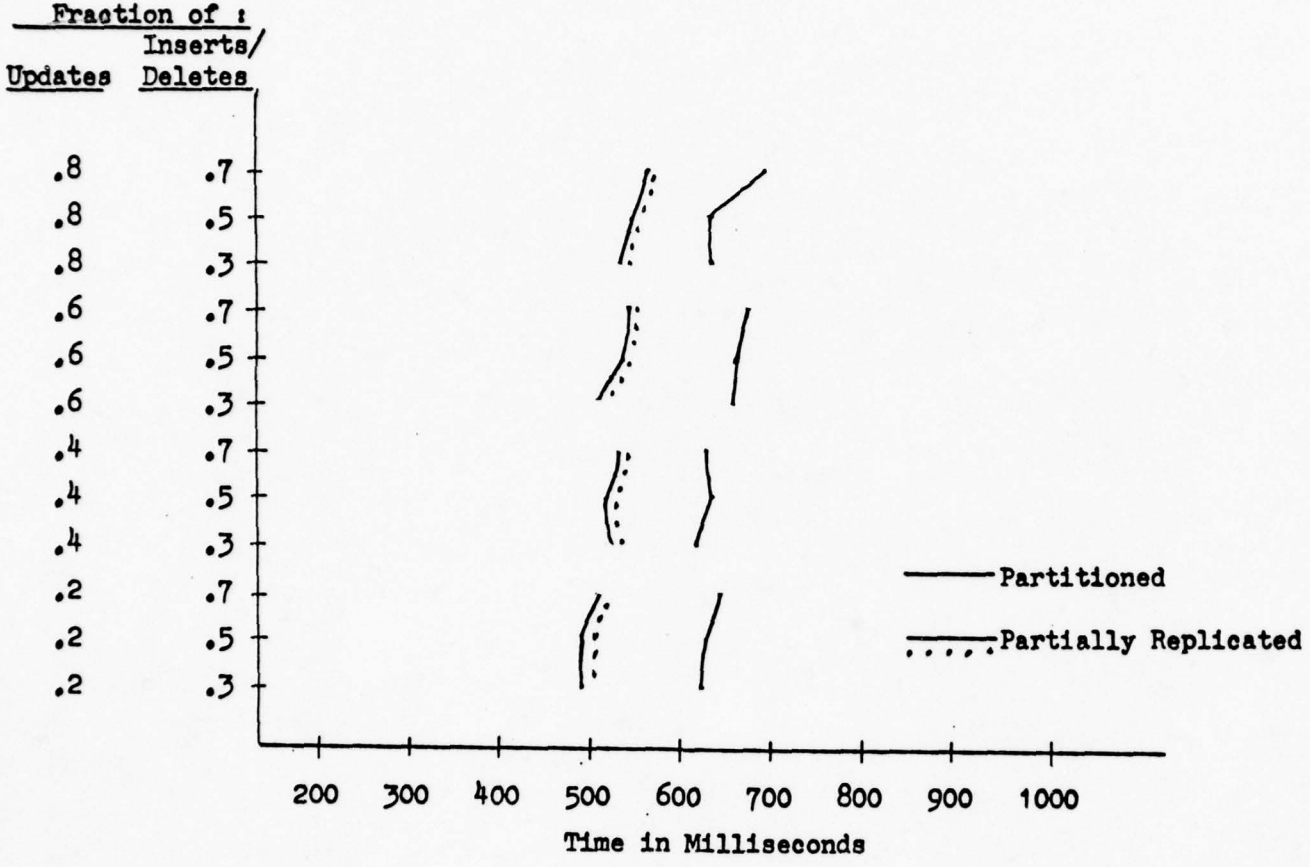
Graph 4.1 Central data dictionary response curve as the fraction of updates and inserts/deletes increase



Graph 4.2 Fully redundant configuration response curve. The first fraction represents updates and the second fraction represents inserts/deletes



Graph 4.3 Partitioned and partially replicated configurations response curve, with 20% messages remote. The first fraction represents updates and the second fraction represents inserts/deletes



Graph 4.4 Partitioned and partially replicated configurations response curve with 80% messages remote. The first fraction represents updates and the second fraction represents inserts/deletes

Chapter 5

DISTRIBUTION PROBLEMS

Central

There are a number of problems associated with distributed systems. Each of the simulated distributions displayed some of these problems. The centralized data dictionary had the advantage of simple updating--one copy. Also if any remote site went down nothing in the data dictionary became inaccessible. Locking in the network to perform updates was not a problem. However, inquiries were handled serially and were very slow at peak periods. Moreover, if the installation that provided the data dictionary went down, the whole network was down.

Fully Redundant

In the fully redundant data dictionary system access is local and very fast. If any installation in the network goes down nothing in the data dictionary becomes inaccessible. On the other hand, any changes to the dictionary requires multiple updates--one for each site. The integrity of the dictionary would be

extremely critical and hard to maintain. If a site failed, provisons would have to be made to provide adequate recovery to insure absolute integrity.

Partitioned

In the partitioned data dictionary any query against local data would be almost instantaneous. Local updates would not affect any other sites. If any site went down the impact would only be minimal. The dictionary integrity would be a simpler problem. Remote traffic, inquiries against non-local data, would result in a slower response because the inquiry would need to be broadcast throughout the network. This is because the local installation has no knowledge of where the remote data is stored. This increase in traffic on the data communication lines would become preponderous and would be far from ideal.

Partially Replicated

This distribution provides quick access for local traffic and is much faster for remote traffic because only one other location is searched. Insertions and deletions are the only updates affecting the remote GDDs which are short and simple. Modify updates affect only the Local Dictionary Directory. Any site lost

would only prevent remote queries from being processed as with the partitioned data dictionary. Like the fully redundant data dictionary system, provisions for handling updates to failed sites would have to be provided for. This last distribution is similar to the system used by Computer Corporation of America in their design of SDD-1 (System for Distributed Databases) [18].

Other Problems

Two other problems in distributed systems also need discussion. First, survivability--the system must continue to operate despite any site failures or inaccessability of one or more databases. Second, reliability and integrity of the database must be maintained. This includes restoring downed sites to the current position by processing any updates that occurred while the installation was down. To increase survivability, each of the seven LDDs in both C and D could overlap with one or more of the other sites. This would allow more remote requests to be answered at the local site instead of being transmitted to another site. It also means less data will be inaccessable because of a node failure. This, however, involves redundant data which introduces a new problem--redundant updating. The easiest method of handling this problem is to lock that

portion of the databases involved in the update. In a distributed system, this would be intolerable.

Thomas [19] proposed a method which reduces the volume of inter-module communications for locking by using a voting protocol. Only a majority of the sites are communicated with. Alsberg and Day [20] suggest working with only one site--the "primary site." All update activity is handled through a single primary site.

Computer Corporation of America [21] sought to avoid all locking in SDD-1. They determined that it is not necessary for all copies to be instantaneously identical at all times. It is sufficient that the same final state be achieved if all update activity were to cease. The update of redundant copies would be accomplished by a specially designed protocol. All messages were assigned a particular class by the DBA. In addition, each message was assigned a timestamp followed by a two digit site number. Once a clock had been read it could not be read again until it had advanced one unit of time. This, along with the two digit site number, gave a globally unique timestamp.

Messages are assigned a class depending on their logical read-set and write-set. Messges of the same class process in timestamp order. Messages of different classes are analyzed by a process described by Bernstein [22] to determine the amount of synchronization that is required. This approach guarantees; first, that after a finite period of time, that the same logical data item will retrieve the same value. This means that all physical copies of a logical data item will converge to the same value. Second, that the interleaved operation of messages is reproducible to an operation that runs serially.

The problem of bringing failed sites up to the current position could be solved by having the receiving installation, as acknowledgement, send a message to the originating installation after having once received the message. In the case of the ARPA network this acknowledgement was a RFNM (Request For Next Message) which took 0.35 msec to generate. If this acknowledgement is never received then the sending installation would re-issue the message.

Recommendation

It is not possible to know in advance exactly what kind of activity will be present on a system or network. Possible activity can be estimated. However, care must be taken to prevent being locked into a configuration that is not flexible. The partitioned data dictionary and the partially replicated data dictionary represent the best response times over a range of message traffic. The partitioned data dictionary, however, clogs up the communication lines rather quickly when requests are made for non-local data.

Regardless of the fact that these simulated configurations were rather simple, a close examination of the characteristics and behavior of each provides firm evidence that a partially replicated data dictionary system is the most efficient and flexible for use in a distributed environment. The most efficient physical organization for this dictionary system is an organization with secondary keys removed from the data files into a chained index. The indexes would be searched by a hash index technique.

Summary

This thesis has explained the development of a data dictionary from the initial step of determining what should be included in a data dictionary.

The data dictionary must include entities such as data items, groups, records, files, systems, programs, and users. Each of these entities will have descriptive attributes documenting their specific features. These will include: how the entity is used, its name, which will include alias names, synonym name, and program names, a complete description of the entity and the security level necessary to access the instances. How these entities relate to each other is an essential part of this dictionary database. This information allows concerned users and management to answer questions like: which programs use the A/R master file, who is authorized to update the customer record file, and how many data elements comprise the inventory record. The data dictionary provides answers concerning existing programs and systems, as well as, information about the affects of future changes.

The canonical schema, representing the numerous views for an ideal data dictionary, was represented in four software packages. Five physical organizations were presented that would best support the dictionary database. Finally, four distributed configurations were simulated and the partially replicated data dictionary containing a Global Dictionary Directory and a Local Dictionary Directory was determined to be superior for general usage in most distributed systems.

REFERENCES

1. Champine, G.A. "Six Approaches to Distributed Data Bases", Datamation, May 1977, p. 69.

2. Ehrensberger, M. "Data dictionary-More on the impossible dream", Proceedings AFIPS 1977, National Computer Conference, AFIPS Press, Vol. 46, 19.77, p. 9.

3. Booth, G.M. "Distributed information systems", Proceedings AFIPS 1976, National Computer Conference, AFIPS Press, Vol. 45, 1976, p. 794.

4. Cullinane Corporation, Brochure, IDMS Integrated Data Dictionary, 1978.

5. Lefkovits, H.C. Data Dictionary Systems, Q.E.D. Information Sciences Inc., 1977, p. 2-1.

6. Plagman, B.K. and Altshuler, G.P. "A data dictionary/directory system within the context of an integrated corporate data base", Proceedings AFIPS 1972, Fall Joint Computer Conference, AFIPS Press, Vol. 41, part II, 1972, p. 1138.

7. Cullinane Corporation, Brochure, Integrated Data Dictionary, p. 6.

8. Ehrensberger, op. cit., p. 10.

9. Lefkovits, op. cit., p. 2-4.

10. Cullinane Corporation, Brochure, IDMS Integrated Data Dictionary, 1978, p. 8.

11. Martin, J. Computer Data-Base Organization, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1977, pp. 274-275.

12. Ibid., p. 248.

13. Ibid., p. 286.

14. Ibid., p. 276.

15. Ibid., p. 255.

16. Image Data Base Management System Reference Manual, Hewlett-Packard, June 1976.

17. Heart, F.E. and others, "The interface message processor for the ARPA Computer network", Proceedings AFIPS 1970, Spring Joint Computer Conference, AFIPS Press, Vol. 40, 1970, p. 564.

18. Rothnie, J.B. and Goodman, N. "An Overview of the Preliminary Design of SDD-1: A System for Distributed Databases", Technical Report No. CCA-77-04, March 31, 1977, Computer Corporation of America,Cambridge, Massachusetts, p. 7.

19. Thomas, R.H. "A Solution to the Update Problem for Multiple Copy Databasesshich Uses Distributed Control", BBN Report No. 3340, July 1975.

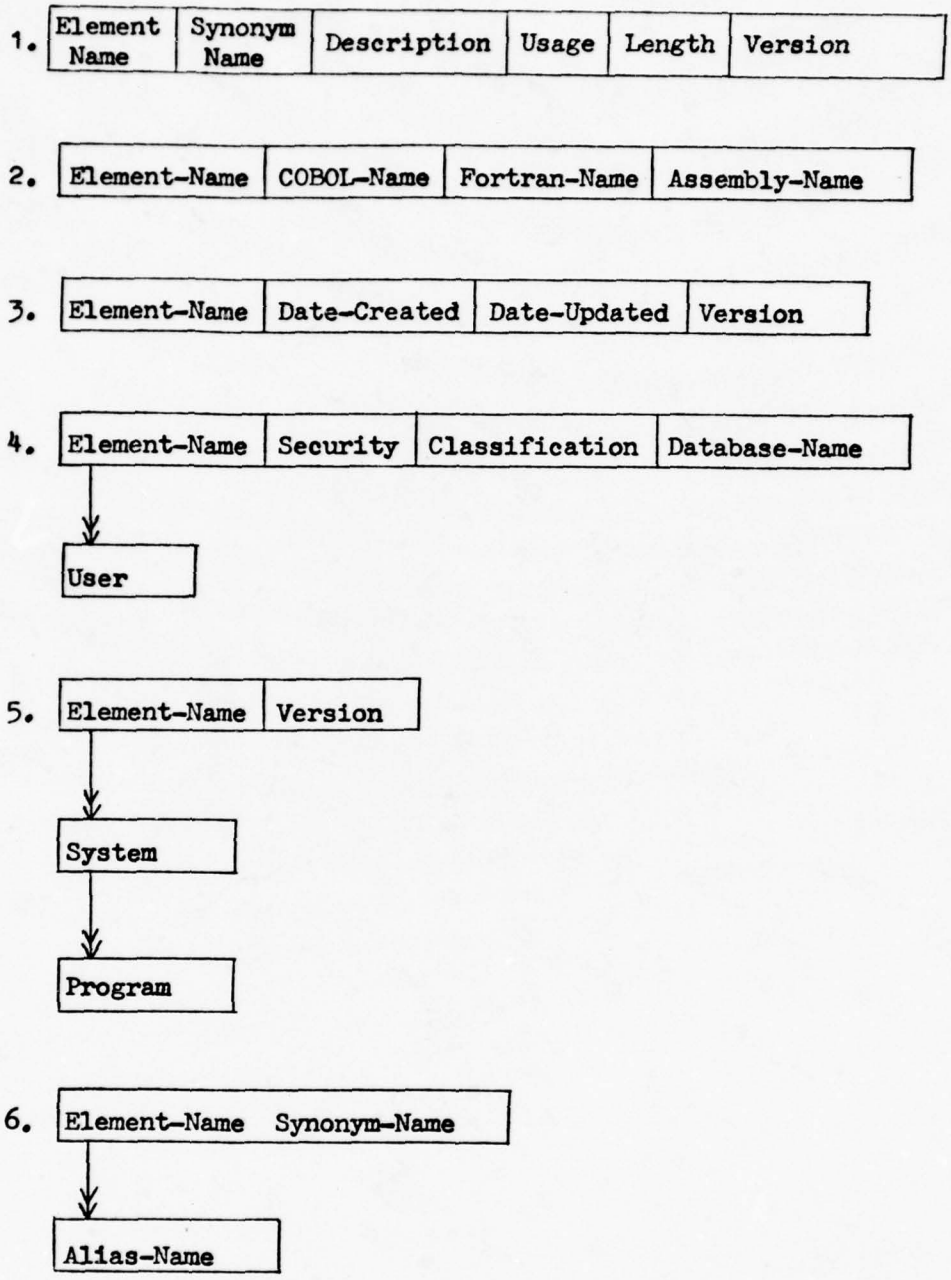
20. Alsberg, P.A. and Day, J.D. "A Principle for Resilient Sharing of Distributed Resources", Report from the Center for Advanced Computation, University of Illinios, Urbana, 1976.

21. Rothnie, and Goodman, op. cit., p. 22.

22. Berstein, P.A. and others, "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases(The General Case), Technical Report No. CCA-77-09, Cambridge, Massachusetts, December 15, 1977.

APPENDIX A

USER VIEWS



Views of Elements

1. | Aggregate-Name | Synonym-Name | Description | Version |

2. | Aggregate-Name | COBOL-Name | Fortran-Name | Assembly-Name |

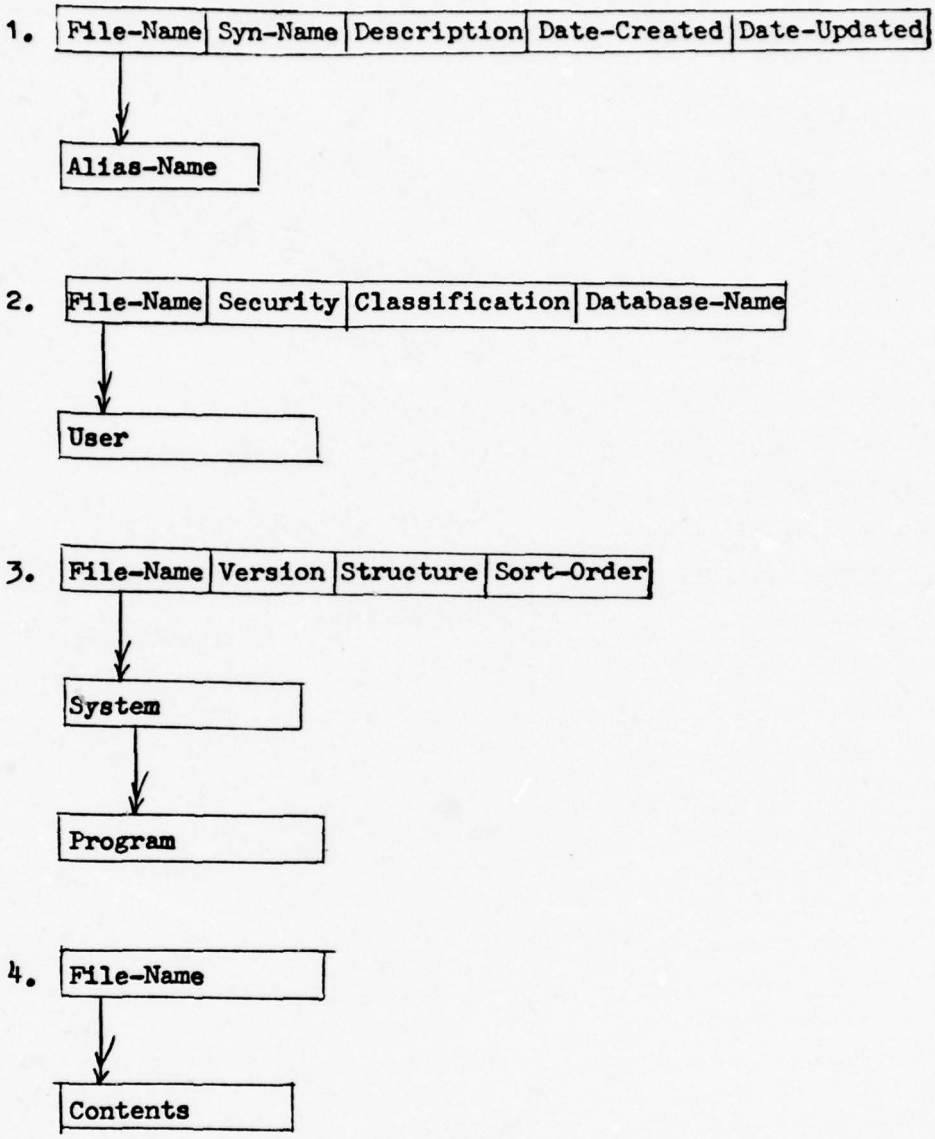
3. | Aggregate-Name | Date-Created | Date-Updated | Version |

4. | Aggregate-Name | Synonym-Name | Contents | Position |
   ↓
   Alias-Name

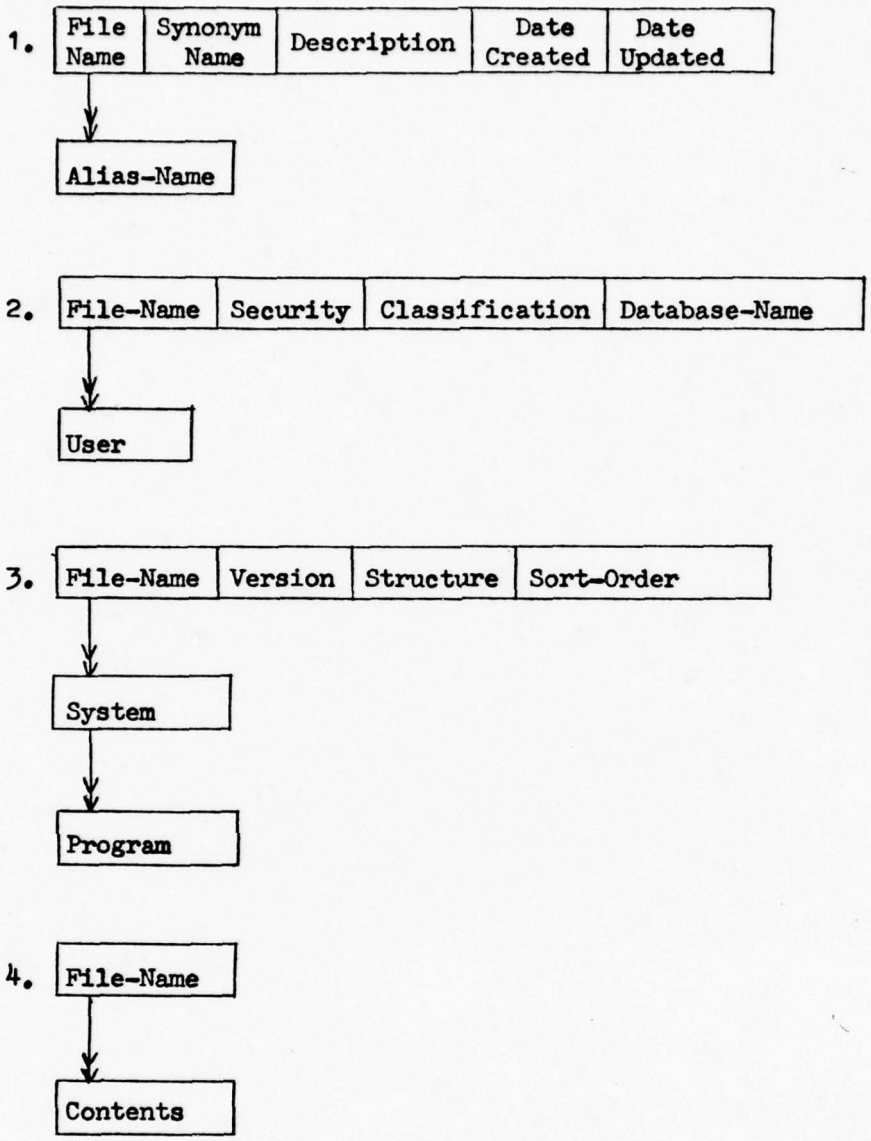
5. | Aggregate-Name | Security | Classification | Database-Name |
   ↓
   User

6. | Aggregate-Name | Version |
   ↓
   System
   ↓
   Program

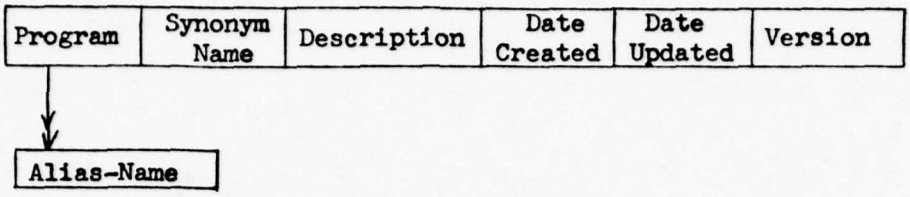
Views of Aggregates

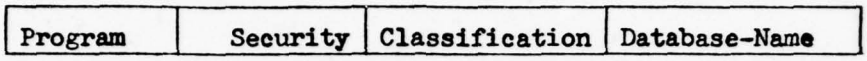


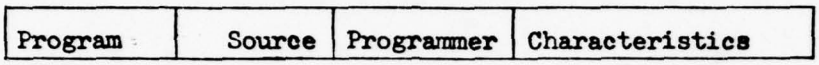
Views of Files

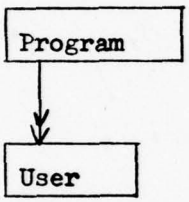


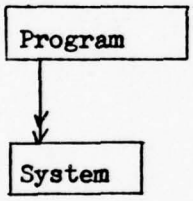
Views of Files

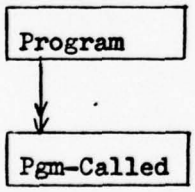
1. 

2. 

3. 

4. 

5. 

6. 

Views of Programs

1. | System | Synonym Name | Description | Security | Classification |

2. | System | Database-Name |

↓

| Program |

Views of Systems

1. | Database-Name | Synonym-Name | DBA | DBA-Details |

2. | Database-Name |

↓

| System |

↓

| Program |

Views of Databases

1. | User | Synonym-Name | Description |

2. | User |

↓

| System |

↓

| Program |

Views of Users

APPENDIX B

IMAGE SCHEMA

```
BEGIN DATA BASE STORE:

PASSWORDS:
    1 DBA;      << DATA BASE ADMINISTRATOR HAS READ>>
                << AND WRITE PERMISSION ON ALL ITEMS>>


ITEMS:
        ACCESS,                 X10  (1/1);<<COMMENTS>>
        ALIAS-NAMES,            X32  (1/1);<<COMMENTS>>
        ASSEMBLY-NAME,          X32  (1/1);
        CHARACTERISTICS,        X40  (1/1);
        CLASSIFICATION,         X10  (1/1);
        COBOL-NAME,             X32  (1/1);
        CONTENTS,               X32  (1/1);
        DATE-CREATED,            X6  (1/1);
        DATE-UPDATED,            X6  (1/1);<<MMDDYY>>
        DBA,                    X24  (1/1);
        DBA-DETAILS,            X40  (1/1);
        DESCRIPTION,            X72  (1/1);
        FORTRAN-NAME,           X6   (1/1);
        LENGTH,                 J3   (1/1);
        PGM-CALLED,             X32  (1/1);
        AGGREGATE-NAME,         X32  (1/1);
        DATABASE-NAME,          X32  (1/1);
        ELEMENT-NAME,           X32  (1/1);
        FILE-NAME,              X32  (1/1);
        PRIMARY-KEY,            X32  (1/1);
        PROGRAM,                X32  (1/1);
        RECORD-NAME,            X32  (1/1);
        SYSTEM,                 X32  (1/1);
        USER,                   X32  (1/1);
        POSITION,               J2   (1/1);
        PROGRAMMER,             X32  (1/1);
        SECURITY,               X12  (1/1);
        SORT-ORDER,             X8   (1/1);
        SOURCE,                 X8   (1/1);
        STRUCTURE,              X12  (1/1);
        SYNONYM-NAME,           X32  (1/1);
        USAGE,                  X6   (1/1);
        VERSION,                X6   (1/1);<<MMDDYY>>
```

```
SETS:


NAME:  MSTR-SYNONYM,AUTOMATIC(1/1);
ENTRY: SYNONYM-NAME(8),  <<NUMBER OF DETAILS SETS>>
CAPACITY: 1000;


NAME:  MSTR-DATABASE,MANUAL(1/1);
ENTRY: DATABASE-NAME(10);
CAPACITY: 10;


NAME:  MSTR-ELEMENT,MANUAL(1/1);
ENTRY: ELEMENT-NAME(4);
CAPACITY: 100000;


NAME:  MSTR-AGGREGATE,MANUAL(1/1);
ENTRY: AGGREGATE-NAME(5);
CAPACITY: 10000;


NAME:  MSTR-RECORD,MANUAL(1/1);
ENTRY: RECORD-NAME(5);
CAPACITY: 10000;


NAME: MSTR-FILE,MANUAL(1/1);
ENTRY: FILE-NAME(5);
CAPACITY: 10000;


NAME:  MSTR-PROGRAM,MANUAL(1/1);
ENTRY: PROGRAM(9);
CAPACITY: 10000;


NAME:  MSTR-SOURCE,AUTOMATIC(1/1);
ENTRY: SOURCE(1);
CAPACITY: 1000;


NAME:  MSTR-PROGRAMMER,AUTOMATIC(1/1);
ENTRY: PROGRAMMER(1); <<SECONDARY INDICE>>
CAPACITY: 1000;
```

```
NAME:  MSTR-SYSTEM,MANUAL(1/1);
ENTRY: SYSTEM(8);
CAPACITY: 100;


NAME:  MSTR-USER,MANUAL(1/1);
ENTRY: USER(7);
CAPACITY: 1000;


NAME:  MSTR-ALIAS,AUTOMATIC(1/1);
ENTRY: ALIAS-NAME(5);
CAPACITY: 200000;


NAME:  ELEMENT-DETAIL,DETAIL(1/1);
ENTRY: ELEMENT-NAME(MSTR-ELEMENT),
       SYNONYM-NAME(MSTR-SYNONYM),
       DESCRIPTION,
       USAGE,
       LENGTH,
       VERSION,
       COBOL-NAME,
       FORTRAN-NAME,
       ASSEMBLER-NAME,
       DATE-CREATED,
       DATE-UPDATED,
       SECURITY,
       CLASSIFICATION,
       DATABASE(MSTR-DATABASE);
CAPACITY: 100000;


NAME:  AGGREGATE-DETAIL,DETAIL(1/1);
ENTRY: AGGREGATE-NAME(MSTR-AGGREGATE),
       SYNONYM-NAME(MSTR-SYNONYM),
       DESCRIPTION,
       VERSION,
       COBOL-NAME,
       FORTRAN-NAME,
       ASSEMBLER-NAME,
       DATE-CREATED,
       DATE-UPDATED,
       SECURITY,
       CLASSIFICATION,
       DATABASE-NAME(MSTR-DATABASE);
CAPACITY: 10000;
```

```
NAME:  RECORD-DETAIL,DETAIL(1/1);
ENTRY: RECORD-NAME(MSTR-RECORD),
       SYNONYM-NAME(MSTR-SYNONYM),
       DESCRIPTION,
       VERSION,
       ACCESS,
       PRIMARY-KEY,
       COBOL-NAME,
       FORTRAN-NAME,
       ASSEMBLER-NAME,
       DATE-CREATED,
       DATE-UPDATED,
       SECURITY,
       CLASSIFICATION,
       DATABASE-NAME(MSTR-DATABASE);
CAPACITY: 10000;


NAME:  FILE-DETAIL,DETAIL(1/1);
ENTRY: FILE-NAME(MSTR-FILE),
       SYNONYM-NAME(MSTR-SYNONYM),
       DESCRIPTION,
       DATE-CREATED,
       DATE-UPDATED,
       VERSION,
       STRUCTURE,
       SORT-ORDER,
       SECURITY,
       CLASSIFICATION,
       DATABASE-NAME(MSTR-DATABASE);
CAPACITY: 10000;


NAME:  PROGRAM-DETAIL,DETAIL(1/1);
ENTRY: PROGRAM(MSTR-PROGRAM),
       SYNONYM-NAME(MSTR-SYNONYM),
       DESCRIPTION,
       DATE-CREATED,
       DATE-UPDATED,
       VERSION,
       SECURITY,
       CLASSIFICATION,
       DATABASE-NAME(MSTR-DATABASE),
       SOURCE(MSTR-SOURCE),
       PROGRAMMER(MSTR-PROGRAMMER),
       CHARACTERISTICS;
CAPACITY: 10000;
```

```
NAME:  SYSTEM-DETAIL,DETAIL(1/1);
ENTRY: SYSTEM(MSTR-SYSTEM),
       DATABASE-NAME(MSTR-DATABASE),
       SYNONYM-NAME(MSTR-SYNONYM),
       DESCRIPTION,
       SECURITY,
       CLASSIFICATION,
CAPACITY: 100;


NAME:  DATABASE-DETAIL,DETAIL,(1/1);
ENTRY: DATABASE-NAME(MSTR-DATABASE),
       SYNONYM-NAME(MSTR-SYNONYM),
       DBA,
       DBA-DETAILS;
CAPACITY: 10;


NAME:  USER-DETAIL,DETAIL(1/1);
ENTRY: USER-NAME(MSTR-USER),
       SYNONYM-NAME(MSTR-SYNONYM);
CAPACITY: 1000;


NAAE:  ELEMENT-USER,DETAIL(1/1);
ENTRY: ELEMENT-NAME(MSTR-ELEMENT),
       USER(MSTR-USER);
CAPACITY: 300000;


NAME:  ELEMENT-SYSTEM,DETAIL,(1/1);
ENTRY: ELEMENT-NAME(MSTR-ELEMENT),
       SYSTEM(MSTR-SYSTEM),
       PROGRAM(MSTR-PROGRAM);
CAPACITY: 300000;


NAME:  ELEMENT-ALIAS,DETAIL(1/1);
ENTRY: ELEMENT-NAME(MSTR-ELEMENT),
       ALIAS-NAME(MSTR-ALIAS);
CAPACITY: 300000;


NAME:  AGGREGATE-USER,DETAIL(1/1);
ENTRY: AGGREGATE-NAME(MSTR-AGGREGATE),
       USER(MSTR-USER);
CAPACITY: 30000;
```

```
NAME:  AGGREGATE-CONTENTS,DETAIL(1/1);
ENTRY: AGGREGATE-NAME(MSTR-AGGREGATE),
       CONTENTS,
       POSITION;
CAPACITY: 50000;


NAME:  AGGREGATE-SYSTEM,DETAIL(1/1);
ENTRY: AGGREGATE-NAME(MSTR-AGGREGATE),
       SYSTEM(MSTR-SYSTEM),
       PROGRAM(MSTR-PROGRAM);
CAPACITY: 50000;


NAME:  AGGREGATE-ALIAS,DETAIL(1/1);
ENTRY: AGGREGATE-NAME(MSTR-AGGREGATE),
       ALIAS-NAME(MSTR-ALIAS);
CAPACITY: 30000;


NAME:  RECORD-CONTENTS,DETAIL(1/1);
ENTRY: RECORD-NAME(MSTR-RECORD),
       CONTENTS,
       POSITION;
CAPACITY: 40000;


NAME:  RECORD-USER,DETAIL(1/1);
ENTRY: RECORD-NAME(MSTR-DECORD),
       USER-NAME(MSTR-USER);
CAPACITY: 20000;


NAME:  RECORD-SYSTEM,DETAIL(1/1);
ENTRY: RECORD-NAME(MSTR-RECORD),
       SYSTEM-NAME(MSTR-SYSTEM),
       PROGRAM(MSTR-PROGRAM);
CAPACITY: 20000;


NAME:  RECORD-ALIAS,DETAIL(1/1);
ENTRY: RECORD-NAME(MSTR-RECORD),
       ALIAS-NAME(MSTR-ALIAS);
CAPACITY: 30000;


NAME:  FILE-CONTENTS,DETAIL(1/1);
ENTRY: FILE(MSTR-FILE),
       CONTENTS;
CAPACITY: 30000;
```

```
NAME:  FILE-USER,DETAIL(1/1);
ENTRY: FILE(MSTR-FILE),
       USER(MSTR-USER);
CAPACITY: 20000;


NAME:  FILE-SYSTEM,DETAIL(1/1);
ENTRY: FILE(MSTR-FILE),
       SYSTEM(MSTR-SYSTEM),
       PROGRAM(MSTR-PROGRAM);
CAPACITY: 30000;


NAME:  FILE-ALIAS,DETAIL(1/1);
ENTRY: FILE(MSTR-FILE),
       ALIAS-NAME(MSTR-ALIAS);
CAPACITY: 20000;


NAME:  PROGRAM-ALIAS,DETAIL(1/1);
ENTRY: PROGRAM(MSTR-PROGRAM),
       ALIAS-NAME(MSTR-ALIAS);
CAPACITY: 20000;


NAME:  PROGRAM-SYSTEM,DETAIL(1/1);
ENTRY: PROGRAM(MSTR-PROGRAM),
       SYSTEM(MSTR-SYSTEM);
CAPACITY: 30000;


NAME:  PROGRAM-USER,DETAIL(1/1);
ENTRY: PROGRAM(MSTR-PROGRAM),
       USER(MSTR-USER);
CAPACITY: 40000;


NAME:  PROGRAM-PROGRAM,DETAIL(1/1);
ENTRY: PROGRAM(MSTR-PROGRAM),
       PGMS-CALLED;
CAPACITY: 60000;


NAME:  DATABASE-SYSTEM,DETAIL(1/1);
ENTRY: DATABASE-NAME(MSTR-DATABASE),
       SYSTEM(MSTR-SYSTEM);
CAPACITY: 40;
```

```
NAME:  USER-SYSTEM,DETAIL(1/1);
ENTRY: USER(MSTR-USER),
       SYSTEM(MSTR-SYSTEM);
CAPACITY: 10000;


END.
```

APPENDIX C

SIMULATION PROGRAM LISTINGS

```
                                    JOB         C,ONE GLOBAL DATA DICTIONARY
                                    SIMULATE
                            INUSE   START,MAT,00
                                    QUEUE       #A
                                    SEIZE       #A
                                    DEPART      #A
                                    ADVANCE     #0
                                    RELEASE     #A
                                    ENDATA,0
                            LAPUN   FUNCTION    RN2,D8
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 1
                                    ASSIGN      1,1
                                    TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 2
                                    ASSIGN      1,2
                                    TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 3
                                    ASSIGN      1,3
                                    TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 4
                                    ASSIGN      1,4
                                    TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 5
                                    ASSIGN      1,5
                                    TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 6
                                    ASSIGN      1,6
                            DAE     INUSE       P1,<120,70>
                                    QUEUE       P1
                                    SEIZE       P1
                                    DEPART      P1
                                    ADVANCE     120,70
                                    RELEASE     P1
                                    TRANSFER    ,END            GET DICTIONARY
                                    GENERATE    5000,FNSEAPUN   SITE NUMBER 7
                                    ASSIGN      1,7
                                    INUSE       P1,<10,75>
                                    QUEUE       P1
                                    SEIZE       P1
                                    DEPART      P1
                                    ADVANCE     10,7
                                    RELEASE     P1
                            END     QUEUE       0
                                    SEIZE       0
                                    DEPART      0
                                    TRANSFER    .2,RETRVE,UPDATE
                            RETRVE  ADVANCE     175,25          150-200 CLOCK UNITS TO RETRIEVE
                                    TRANSFER    ,OUT            RELEASE DICTIONARY
                            UPDATE  TRANSFER    .5,INSERT,MODIFY
                            INSERT  ADVANCE     525,75          450-600 CLOCK UNITS TO INSERT/DELETE
                                    TRANSFER    ,OUT            RELEASE DICTIONARY
                            MODIFY  ADVANCE     225,75          150-300 CLOCK UNITS TO MODIFY
                            OUT     RELEASE     0
                                    QUEUE       P1
                                    SEIZE       P1
```

```
            56      44                      DEPART      P1
            57      45                      TRANSFER    .15,LONG,SHORT
            58      46          LONG        ADVANCE     120,70
            59      47                      TRANSFER    ,FIN
            60      48          SHORT       ADVANCE     10,7
            61      49           FIN        RELEASE     P1
            62      50                      TABULATE    1
            63                  1           TABLE       M1,200,25,50
            64      51                      TERMINATE   1
            65                              START       1000
            66                              END
```

```
                                    JOB       1,7 GLOBAL DATA DICTIONARIES
                                    SIMULATE
                          INUSE     STARTMACRO
                                    QUEUE     #A
                                    SEIZE     #A
                                    DEPART    #A
                                    ADVANCE   #B
                                    RELEASE   #A
                                    ENDMACRO
                          EXPON     FUNCTION  RN5,NE                 RANDOM NUMBER 5 WITH EXPON DISTN
1                                   GENERATE  5000,FNSEXPON           SITE NUMBER 1
2                                   ASSIGN    1,1
3                                   ASSIGN    2,11
4                                   TRANSFER  ,ONE                   DETERMINE TRANSMISSION DELAY
5                                   GENERATE  5000,FNSEXPON           SITE NUMBER 2
6                                   ASSIGN    1,2
7                                   ASSIGN    2,12
8                                   TRANSFER  ,ONE                   DETERMINE TRANSMISSION DELAY
9                                   GENERATE  5000,FNSEXPON           SITE NUMBER 3
10                                  ASSIGN    1,3
11                                  ASSIGN    2,13
12                                  TRANSFER  ,ONE                   DETERMINE TRANSMISSION DELAY
13                                  GENERATE  5000,FNSEXPON           SITE NUMBER 4
14                                  ASSIGN    1,4
15                                  ASSIGN    2,14
16                                  TRANSFER  ,ONE                   DETERMINE TRANSMISSION DELAY
17                                  GENERATE  5000,FNSEXPON           SITE NUMBER 5
18                                  ASSIGN    1,5
19                                  ASSIGN    2,15
20                                  TRANSFER  ,ONE                   DETERMINE TRANSMISSION DELAY
21                                  GENERATE  5000,FNSEXPON           SITE NUMBER 6
22                                  ASSIGN    1,6
23                                  ASSIGN    2,16
24                                  TRANSFER  ,ONE                   DETERMINE TRANSMISSION DELAY
25                                  GENERATE  5000,FNSEXPON           SITE NUMBER 7
26                                  ASSIGN    1,7
27                                  ASSIGN    2,17
          ONE       INUSE     P1,<10,7>       SEND REQUEST
28M                 QUEUE     P1
29M                 SEIZE     P1
30M                 DEPART    P1
31M                 ADVANCE   10,7
32M                 RELEASE   P1
33                  TRANSFER  .2,RETRVE,UPDATE
          RETRVE    INUSE     P2,<175,25>     150-200 CLOCK UNITS TO RETRIEVE
34M                 QUEUE     P2
35M                 SEIZE     P2
36M                 DEPART    P2
37M                 ADVANCE   175,25
38M                 RELEASE   P2
39                  INUSE     P1,<10,7>       RESPONSE
40M                 QUEUE     P1
41M                 SEIZE     P1
42M                 DEPART    P1
43M                 ADVANCE   10,7
```

```
/ GLOBAL DATA DICTIONARIES
SIMUL2  GPS     20-JUN-78 23:46

                                RELEASE   P1
                                TABULATE  P1
                                TRANSFER  ,FIN
                UPDATE          ASSIGN    3,0             INITIALIZE FOR ASSIGNING QUEUES
                                TRANSFER  .3,INSERT,MODIFY
                INSERT          ASSIGN    3,1
                                TRANSFER  ,CONT
                MODIFY          ASSIGN    3,2
                CONT            SPLIT     0,NET,3
                                QUEUE     P2
                                SEIZE     P2              DATA DICTIONARY
                                DEPART    P2
                                TEST E    P3,1,MOD        TRANSFER TO MOD IF FALSE
                                ADVANCE   525,75          450-600 CLOCK UNITS TO INSERT/DELETE
                                TRANSFER  ,PAST
                MOD             ADVANCE   225,75          150-300 CLOCK UNITS TO MODIFY
                PAST            RELEASE   P2
                                TRANSFER  ,OUT
                NET             TEST E    P1,P3,BYE       TRANSFER TO BYE IF FALSE
                                ASSIGN    3,1             DONT SEND TO PARENT SITE
                BYE             BOOST     P3,<120,70>     REMOTE TRANSMISSION TIME
                                QUEUE     P3
                                SEIZE     P3
                                DEPART    P3
                                ADVANCE   120,70
                                RELEASE   P3
                                ASSIGN    3,10            GET DATA DICTIONARY
                                QUEUE     P3
                                SEIZE     P3
                                DEPART    P3
                                TEST E    P3,1,MOD1       TRANSFER TO MOD1 IF FALSE
                                ADVANCE   525,75          450-600 CLOCK UNITS TO INSERT/DELETE
                                TRANSFER  ,DELAY
                MOD1            ADVANCE   225,75          150-300 CLOCK UNITS TO MODIFY
                DELAY           RELEASE   P3
                                ASSIGN    3=,10
                                BOOST     P3,<120,70>     REMOTE TRANSMISSION TIME
                                QUEUE     P3
                                SEIZE     P3
                                DEPART    P3
                                ADVANCE   120,70
                                RELEASE   P3
                OUT             ASSEMBLE  1
                                BOOST     P1,<10,7>       RESPONSE
                                QUEUE     P1
                                SEIZE     P1
                                DEPART    P1
                                ADVANCE   10,7
                                RELEASE   P1
                                TABULATE  P1
                FIN             TERMINATE 1
                1               TABLE     M1,100,100,25
                2               TABLE     M1,100,100,25
                3               TABLE     M1,100,100,25
                4               TABLE     M1,100,100,25
```

? GLOBAL DATA DICTIONARIES
SIMUL2 GPS 20-JUN-78 23:46

```
111                 5       TABLE   M1,100,100,25
112                 5       TABLE   M1,100,100,25
113                 7       TABLE   M1,100,100,25
114                         START   1000
115                         END
```

```
                1                       JOB         C,7 LOCAL DATA DICTIONARIES
                2                       SIMULATE
                3           INUSE       STARTMACRO
                4                       QUEUE       #A
                5                       SEIZE       #A
                6                       DEPART      #A
                7                       ADVANCE     #B
                8                       RELEASE     #A
                9                       ENDMACRO
               10           EXPON       FUNCTION    RN5,BE          RANDOM NUMBER 5 WITH EXPON DISTR
               11           SITE        FUNCTION    RN3,D7
               12          .14,1/.28,2/.42,3/.56,4/.7,5/.84,6/1.0,7/
               13       1                  GENERATE    5000,FNSEXPON   SITE NUMBER 1
               14       2                  ASSIGN      1,1
               15       3                  ASSIGN      2,11
               16       4                  TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               17       5                  GENERATE    5000,FNSEXPON   SITE NUMBER 2
               18       6                  ASSIGN      1,2
               19       7                  ASSIGN      2,12
               20       8                  TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               21       9                  GENERATE    5000,FNSEXPON   SITE NUMBER 3
               22      10                  ASSIGN      1,3
               23      11                  ASSIGN      2,13
               24      12                  TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               25      13                  GENERATE    5000,FNSEXPON   SITE NUMBER 4
               26      14                  ASSIGN      1,4
               27      15                  ASSIGN      2,14
               28      16                  TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               29      17                  GENERATE    5000,FNSEXPON   SITE NUMBER 5
               30      18                  ASSIGN      1,5
               31      19                  ASSIGN      2,15
               32      20                  TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               33      21                  GENERATE    5000,FNSEXPON   SITE NUMBER 6
               34      22                  ASSIGN      1,6
               35      23                  ASSIGN      2,16
               36      24                  TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               37      25                  GENERATE    5000,FNSEXPON   SITE NUMBER 7
               38      26                  ASSIGN      1,7
               39      27                  ASSIGN      2,17
               40                  ONE     INUSE       P1,<10,7>       SEND REQUEST
               41M     28                  QUEUE       P1
               42M     29                  SEIZE       P1
               43M     30                  DEPART      P1
               44M     31                  ADVANCE     10,7
               45M     32                  RELEASE     P1
               46      33                  TRANSFER    .2,LOCAL,REMOTE
               47      34          LOCAL   QUEUE       P2
               48      35                  SEIZE       P2
               49      36                  DEPART      P2              DICTIONARY LOOK-UP
               50      37                  TRANSFER    .2,RETRVE,UPDATE
               51      38          RETRVE  ADVANCE     175,25          150-200 CLOCK UNITS TO RETRIEVE
               52      39                  TRANSFER    ,RELEND
               53      40          UPDATE  TRANSFER    .3,INSERT,MODIFY
               54      41          INSERT  ADVANCE     525,75          450-600 CLOCK UNITS TO INSERT/DELETE
               55      42                  TRANSFER    ,RELEND
```

```
                56      43              MODIFY  ADVANCE  225,75          150-300 CLOCK UNITS TO MODIFY
                57      44              RELEMO  RELEASE  P2
                58                      INUSE    P1,<10,7>              RESPONSE
                59M     45                       QUEUE    P1
                60M     46                       SEIZE    P1
                61M     47                       DEPART   P1
                62M     48                       ADVANCE  10,7
                63M     49                       RELEASE  P1
                64      50                       TABULATE P1
                65      51                       TRANSFER ,OUT
                66      52              REMOTE  ASSIGN   3,0              NO UPDATES FOR REMOTE SITES
                67                      INUSE    P2,<120,10>            LOOK UP IN LDD
                68M     53                       QUEUE    P2
                69M     54                       SEIZE    P2
                70M     55                       DEPART   P2
                71M     56                       ADVANCE  120,10
                72M     57                       RELEASE  P2
                73      58              RETRY   ASSIGN   4,FN$SITE        SITES CANNOT BE THE SAME
                74      59                      TEST NE  P1,P4,RETRT     UPDATE ONLY ONE SITE
                75      60                      SPLIT    6,NET,3         BROADCAST MSG IN NETWORK
                76      61                      TRANSFER ,COLLECT
                77      62              NET     TEST E   P1,P3,BYE       TRANSFER TO BYE IF FALSE
                78      63                      ASSIGN   3,1
                79                      BYE     INUSE    P3,<120,70>     REMOTE TRANSMISSION TIME
                80M     64                       QUEUE    P3
                81M     65                       SEIZE    P3
                82M     66                       DEPART   P3
                83M     67                       ADVANCE  120,70
                84M     68                       RELEASE  P3
                85      69                      TEST E   P3,P4,ACK       IF P3 EQ P4 THIS IS THE SITE
                86      70                      ASSIGN   3+,K10          GET REMOTE SITE DICTIONARY
                87                              INUSE    P3,<175,25>     150-200 CLOCK UNITS TO RETRIEVE
                88M     71                       QUEUE    P3
                89M     72                       SEIZE    P3
                90M     73                       DEPART   P3
                91M     74                       ADVANCE  175,25
                92M     75                       RELEASE  P3
                93      76                      ASSIGN   3-,K10
                94                      ACK     INUSE    P3,<120,70>     REMOTE TRANSMISSION TIME
                95M     77                       QUEUE    P3
                96M     78                       SEIZE    P3
                97M     79                       DEPART   P3
                98M     80                       ADVANCE  120,70
                99M     81                       RELEASE  P3
               100      82              COLLECT ASSEMBLE 7
               101                              INUSE    P1,<10,7>       RESPONSE
               102M     83                       QUEUE    P1
               103M     84                       SEIZE    P1
               104M     85                       DEPART   P1
               105M     86                       ADVANCE  10,7
               106M     87                       RELEASE  P1
               107      88                       TABULATE P1
               108      89              OUT     TERMINATE 1
               109                      1       TABLE    M1,100,100,25
               110                      2       TABLE    M1,100,100,25
```

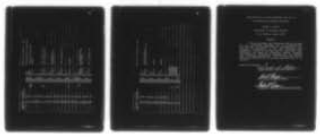
```
        111                 3       TABLE   M1,100,100,25
        112                 4       TABLE   M1,100,100,25
        113                 5       TABLE   M1,100,100,25
        114                 6       TABLE   M1,100,100,25
        115                 7       TABLE   M1,100,100,25
        116                         START   1000
        117                         END
```

```
                1                       JOB         C,7 GOD/LOD DATA DICTIONARIES
                2                       SIMULATE
                3           INUSE       STANDMACRO
                4                       QUEUE       #A
                5                       SEIZE       #A
                6                       DEPART      #A
                7                       ADVANCE     #B
                8                       RELEASE     #A
                9                       ENDMACRO
               10           EXPON       FUNCTION    RN5,BE          RANDOM NUMBER 5 WITH EXPON DISTR
               11           SITE        FUNCTION    RN4,D7
               12          .14,1/.28,2/.42,3/.56,4/.7,5/.84,6/1.0,7/
               13      1                GENERATE    5000,FNSEXPON   SITE NUMBER 1
               14      2                ASSIGN      1,1
               15      3                ASSIGN      2,11
               16      4                TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               17      5                GENERATE    5000,FNSEXPON   SITE NUMBER 2
               18      6                ASSIGN      1,2
               19      7                ASSIGN      2,12
               20      8                TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               21      9                GENERATE    5000,FNSEXPON   SITE NUMBER 3
               22     10                ASSIGN      1,3
               23     11                ASSIGN      2,13
               24     12                TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               25     13                GENERATE    5000,FNSEXPON   SITE NUMBER 4
               26     14                ASSIGN      1,4
               27     15                ASSIGN      2,14
               28     16                TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               29     17                GENERATE    5000,FNSEXPON   SITE NUMBER 5
               30     18                ASSIGN      1,5
               31     19                ASSIGN      2,15
               32     20                TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               33     21                GENERATE    5000,FNSEXPON   SITE NUMBER 6
               34     22                ASSIGN      1,6
               35     23                ASSIGN      2,16
               36     24                TRANSFER    ,ONE            DETERMINE TRANSMISSION DELAY
               37     25                GENERATE    5000,FNSEXPON   SITE NUMBER 7
               38     26                ASSIGN      1,7
               39     27                ASSIGN      2,17
               40           ONE         INUSE       P1,<10,7>       SEND REQUEST
              41M     28                QUEUE       P1
              42M     29                SEIZE       P1
              43M     30                DEPART      P1
              44M     31                ADVANCE     10,7
              45M     32                RELEASE     P1
               46     33                TRANSFER    .2,LOCAL,REMOTE
               47     34           LOCAL  QUEUE     P2
               48     35                SEIZE       P2              DATA DICTIONARY
               49     36                DEPART      P2
               50     37                TRANSFER    .2,RETRVE,UPDATE
               51     38           RETRVE ADVANCE   175,25          150-200 CLOCK UNITS TO RETRIEVE
               52     39                TRANSFER    ,RELEND
               53     40           UPDATE TRANSFER  .3,INSERT,MODIFY
               54     41           INSERT ASSIGN    3,0
               55     42                SPLIT       6,NET,3         UPDATE ALL GDD S
```

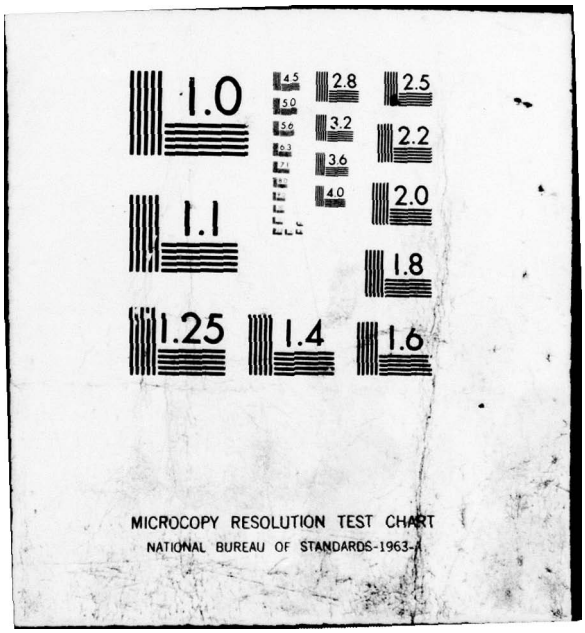
AD-A062 716 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO F/G 9/2
DEVELOPMENT OF A DATA DICTIONARY: FOR USE IN A DISTRIBUTED INTE--ETC(U)
AUG 78 G K POWELL

UNCLASSIFIED AFIT-CI-79-84 NL

2 OF 2
AD
A062716



END
DATE
FILMED
3-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
                56      43                  ADVANCE   525,75          450-600 CLOCK UNITS TO INSERT/DELETE
                57      44                  RELEASE   P2
                58      45                  TRANSFER  ,COLLECT
                59      46          MODIFY  ADVANCE   225,75          150-300 CLOCK UNITS TO MODIFY
                60      47          RELEND  RELEASE   P2
                61                          INUSE     P1,<10,7>       RESPONSE
                62M     48                  QUEUE     P1
                63M     49                  SEIZE     P1
                64M     50                  DEPART    P1
                65M     51                  ADVANCE   10,7
                66M     52                  RELEASE   P1
                67      53                  TABULATE  P1
                68      54                  TRANSFER  ,OUT
                69      55          NE1     TEST E    P1,P3,HYE       SITES MAY NOT BE THE SAME
                70      56                  ASSIGN    3,1
                71                  HYE     INUSE     P3,<120,70>     REMOTE TRANSMISSION TIME
                72M     57                  QUEUE     P3
                73M     58                  SEIZE     P3
                74M     59                  DEPART    P3
                75M     60                  ADVANCE   120,70
                76M     61                  RELEASE   P3
                77      62                  ASSIGN    3+,K10          DATA DICTIONARY
                78                          INUSE     P3,<110,10>     ENTER/DELETE FROM GDD
                79M     63                  QUEUE     P3
                80M     64                  SEIZE     P3
                81M     65                  DEPART    P3
                82M     66                  ADVANCE   110,10
                83M     67                  RELEASE   P3
                84      68                  ASSIGN    3-,K10
                85                          INUSE     P3,<120,70>     REMOTE TRANSMISSION TIME
                86M     69                  QUEUE     P3
                87M     70                  SEIZE     P3
                88M     71                  DEPART    P3
                89M     72                  ADVANCE   120,70
                90M     73                  RELEASE   P3
                91      74          COLLECT ASSEMBLE  7
                92                          INUSE     P1,<10,7>       RESPONSE
                93M     75                  QUEUE     P1
                94M     76                  SEIZE     P1
                95M     77                  DEPART    P1
                96M     78                  ADVANCE   10,7
                97M     79                  RELEASE   P1
                98      80                  TABULATE  P1
                99      81                  TRANSFER  ,OUT
               100      82          REMOTE  ASSIGN    4,FNSSITE       GET RANDOM REMOTE SITE
               101      83                  TEST NE   P1,P4,REMOTE    SITES CANNOT BE THE SAME
               102                          INUSE     P2,<100,10>     LOOK UP IN GDD
               103M     84                  QUEUE     P2
               104M     85                  SEIZE     P2
               105M     86                  DEPART    P2
               106M     87                  ADVANCE   100,10
               107M     88                  RELEASE   P2
               108                          INUSE     P4,<120,70>     REMOTE TRANSMISSION TIME
               109M     89                  QUEUE     P4
               110M     90                  SEIZE     P4
```

```
            111M    91                  DEPART    P4
            112M    92                  ADVANCE   120,70
            113M    93                  RELEASE   P4
            114     94                  ASSIGN    4+,K10          REMOTE DATA DICTIONARY
            115                         INUSE     P4,<175,25>     150-200 CLOCK UNITS TO RETRIEVE
            116M    95                  QUEUE     P4
            117M    96                  SEIZE     P4
            118M    97                  DEPART    P4
            119M    98                  ADVANCE   175,25
            120M    99                  RELEASE   P4
            121     100                 ASSIGN    4-,K10
            122                         INUSE     P4,<120,70>     RESPONSE
            123M    101                 QUEUE     P4
            124M    102                 SEIZE     P4
            125M    103                 DEPART    P4
            126M    104                 ADVANCE   120,70
            127M    105                 RELEASE   P4
            128                         INUSE     P1,<10,7>       RESPONSE
            129M    106                 QUEUE     P1
            130M    107                 SEIZE     P1
            131M    108                 DEPART    P1
            132M    109                 ADVANCE   10,7
            133M    110                 RELEASE   P1
            134     111                 TABULATE  P1
            135                 1       TABLE     M1,100,100,25
            136                 2       TABLE     M1,100,100,25
            137                 3       TABLE     M1,100,100,25
            138                 4       TABLE     M1,100,100,25
            139                 5       TABLE     M1,100,100,25
            140                 6       TABLE     M1,100,100,25
            141                 7       TABLE     M1,100,100,25
            142     112         OUT     TERMINATE 1
            143                         START     1000
            144                         END
```

89

DEVELOPMENT OF A DATA DICTIONARY: FOR USE IN A DISTRIBUTED INTEGRATED DATABASE

Gordon K. Powell

Department of Computer Science

M. S. Degree, August 1978

ABSTRACT

A data dictionary was developed that can be used on an integrated database that is geographically distributed. The content, logical representation, and physical representation of the data dictionary were presented. A study of the best way to distribute this dictionary was made by simulating several proposed distributions. Thousands of messages were run against each simulated distribution. The best distribution was found after determining the average response time of a message.

COMMITTEE APPROVAL:

Gordon E. Stokes

Gordon Stokes, Committee Chairman

Bill Hays

Bill Hays, Committee Member

Theodore A. Norman

Theodore A. Norman, Department Chairman