Insights into Experiential Pedagogy, Volume 6, 1979

THE DESIGN OF A DATABASE SYSTEM TO SUPPORT BUSINESS SIMULATION AND EXPERIENTIAL LEARNING

F. Paul Fuhs, Virginia Commonwealth University

ABSTRACT

This paper extends the concept of variability in models and cases beyond the use of random number generation to include the appropriate substitution of facts and questions at designated nodes within models and cases. The database concept of program and data independence is applied to the model construction process to support the model builder in controlling the different associational paths which are generated in a model or case, when one set of facts is substituted for another. We present the design of a database system, FAQSCAM (Facts and Questions for Simulations, Cases, and Models) at the School of Business of Virginia Commonwealth University. This system supports the data and associational requirements of both computerized and non-computerized cases and models. We describe how a model builder can express desired associational paths within a model, how these descriptions are incorporated into the database, and how these paths can later be modified. We demonstrate how a single logical generalized database structure and a single physical database will simultaneously support many cases and models, each having its own facts and questions and its own logical structure of as8ociational paths through the case or model.

INTRODUCTION

Simulation models and cases require considerable time in the development phase of their life cycle. This time is justified only if the models and cases can be used repetitively. Unfortunately, the continued use of many models and cases is limited not only by their content, but also by their exposure to the participants, the population for whom they were developed. If the participants are students, variability of facts and questions within models and cases is highly desirable. Variability of facts and questions in a model or case allows the case or model to be used over a number of years, allows a participant to interact mote than once with the model or case, and prevents the detection of a single set of 'acceptable" answers.

This paper addresses the problem of controlling variability in simulation models, games, and cases where random number generation does not provide sufficient variability. Since the methodology presented is applicable to simulation models, games, cases and to some types of experiential learning, the term "model' will be used to imply all of these forms of learning. The notion of variability in models is extended in this paper to include the ability of the model to pre- sent any one of a set of facts or questions at each of a number of points in the dialog between the model and its participants. This is referred to as text substitution, the appropriate substitution of logically related facts and questions. We point Out that variability creates associational problems between blocks of text for the model builder in the design phase and at those times when the model is used. A methodology by which the model designer can structure a model to incorporate variability and a means of expressing this variability for computer input to a database is presented. Text substitution can be accomplished without tearing apart the fabric of a model by applying the concept of program and data independence to the design phase of model building. The database constructs of hashing and linked lists can control the presentation of associated blocks of text to participants, when they are interacting with either a computerized or non-computerized model. Lastly, we show how at Virginia Commonwealth University the different logical Structures of many different models can simultaneously be supported within one generalized logical and physical database structure.

Information Flows in Models

One can view a model as a sequence of information flows between the model (or model builder) and its participants. At any given time in this sequence the participant is either the communication source or sink. The model builder acts as communication source by sending information to the participant in two forms; facts and questions. The participant assumes the role of communication source by responding to questions, by inputting questions, or by actively changing the direction of the dialog.

There are at least three types of information that can be communicated to a participant through a model. The first is Internal information; facts and questions pertinent to the real world beyond the model or to a view of the real world as presented by the model. Internal information is usually related to the primary objectives of a model, whether the objectives are to acquire a skill, to learn more about a certain aspect of reality, or to enhance problem solving ability. Interpretive information is the second type of information found in models. Interpretive facts and questions pertain to information about the role of the participant in the model, how the dialog is to proceed, and the mechanics of how the model itself functions to mirror external reality. The debriefing process is in this interpretive information category For it is important to ascertain at various points in the presentation of the model whether the participant understands the objectives, assumptions, and variables of the model, as well as the underlying principles and theories. Reflexive information is the third type of information found in models. This category presents the participant with feedback on his performance as he interacts with a model and presents guidance on how the participant can improve his role or behavior with respect to the model. In addition to these three types of information there are times when it is valid to deliberately communicate noise in a model (irrelevant or illogical information), especially if one of the objectives of the model is to teach the participant how to distinguish noise from information.

TEXTUAL VARIABILITY

The flow of these types of information and noise between a model and its participants is considered to be in the form of text. Text can be a single value, a word, a phrase, a sentence, or one or more contiguous paragraphs. A block of text represents some arbitrary semantic unit to the model builder.

Every model has its own requirement for the different types of information and for different amounts of

textual variability, depending on the model's objectives and content. Some models are relatively self-con-tamed and can generate sufficient variability through their own algorithmic processes, their use of random number generation, or the conditional sequencing of their events and processes. Other models are enhanced by the proper inclusion of substitutive facts and questions, drawn from a database constructed and maintained separately, but accessed by the model itself.

Problems of Logical Inconsistency

When a model builder desires to include textual variability in a model, the danger arises of creating logical inconsistencies between the facts sequentially presented, or between the facts presented and a later question. This is especially true, when the selection of facts and questions is automated. Textual substitution comes in two forms; equivalency and mutual exclusion. Two texts are equivalent, if they are either synonymous or if, while they convey different information, they do not create inconsistencies. Two texts are mutually exclusive, if only one of these can be p resented to a participant at one exposure to the model, without creating logical inconsistencies. How can a model designer structure a model to avoid logical inconsistencies? How can a model be designed so that its related facts and questions will be logically associated?

Link Paths and Levels

Linked lists are appropriate for maintaining logical consistency in a model which incorporates textual variability. A linked list is a group of ordered data items or elements [1, p.57]. A link path is a path of associated elements in a network, which network contains interrelated and mutually exclusive elements. These definitions can easily be extended to include ordered sequences of textual material, that is, facts and questions. Informational link paths can be made to exist between related facts and questions.

Each block of text, expressed as a set of facts or a single question can be represented as a node in a network of nodes. Each node can belong to one or more link paths. A block of text is defined as constant or variable depending on the link paths which flow through it. If all link paths flow through a particular node, that block of text is considered constant for the model. Every participant receives this block of text within a sequence of information presented. The mutually exclusive type of textual variability can be represented as occurring when a parent node has more than one link path leading from it. Such twin nodes of a common parent represent mutually exclusive information. Textual variability of the equivalent type can be represented at a node by sub-nodes within that node. Modeling equivalent variability as a set of sub-nodes preserves the capability of direct substitution of any one of the sub-nodal texts within a node. The representation of nodes along link paths preserves logical consistency between not only facts, but also between facts and questions.

Recognizing that a model has its own algorithmic and branching structure, nevertheless, from the viewpoint of the flow of information between a model and its participants, a model may be structurally depicted as a linear sequence of information.[1] Since information is frequently presented in a sequential manner, e.g. books, movies, and lectures, the informational nodes of a model can be organized on a more general level. The nodes of a model can be clustered into a set of linearly arranged levels. The model designer arbitrarily defines the number of levels. However, this number frequently is determined by either the functions within the model, changes in the source-sink relationship, or the link path length. The logical representation of the information flow of a model is, therefore, a network of nodes embedded within the framework of a sequence of informational levels.

INFORMATIONAL NODE STRUCTURES

Figure 1 is an example of the use of levels and nodes to represent textual variability within a model. This figure shows the first seven levels of a model, its nodes of facts and questions, and its link path structure. Each node is labeled by a level number followed by an arbitrary, but unique node number for that level. A node numbered Out of sequence like node 3.6 represents a node that might have been added after the original design of the model was completed. The model designer is responsible for the design of such a structure. The designer determines the number of levels and nodes and the content of the material at each node. Each node contains one or more sub-nodes. Each sub-node represents equivalent text; either a set of facts or a single question. For clarity the sub-nodes in figure 1 are not depicted, but can be visualized as a third dimension projecting upward from the surface of the paper. In figure 1 the dashed line from node 3.2 represents a link between the sub-nodes of node 3.2. Other nodes may have similar links, although they are not illustrated.

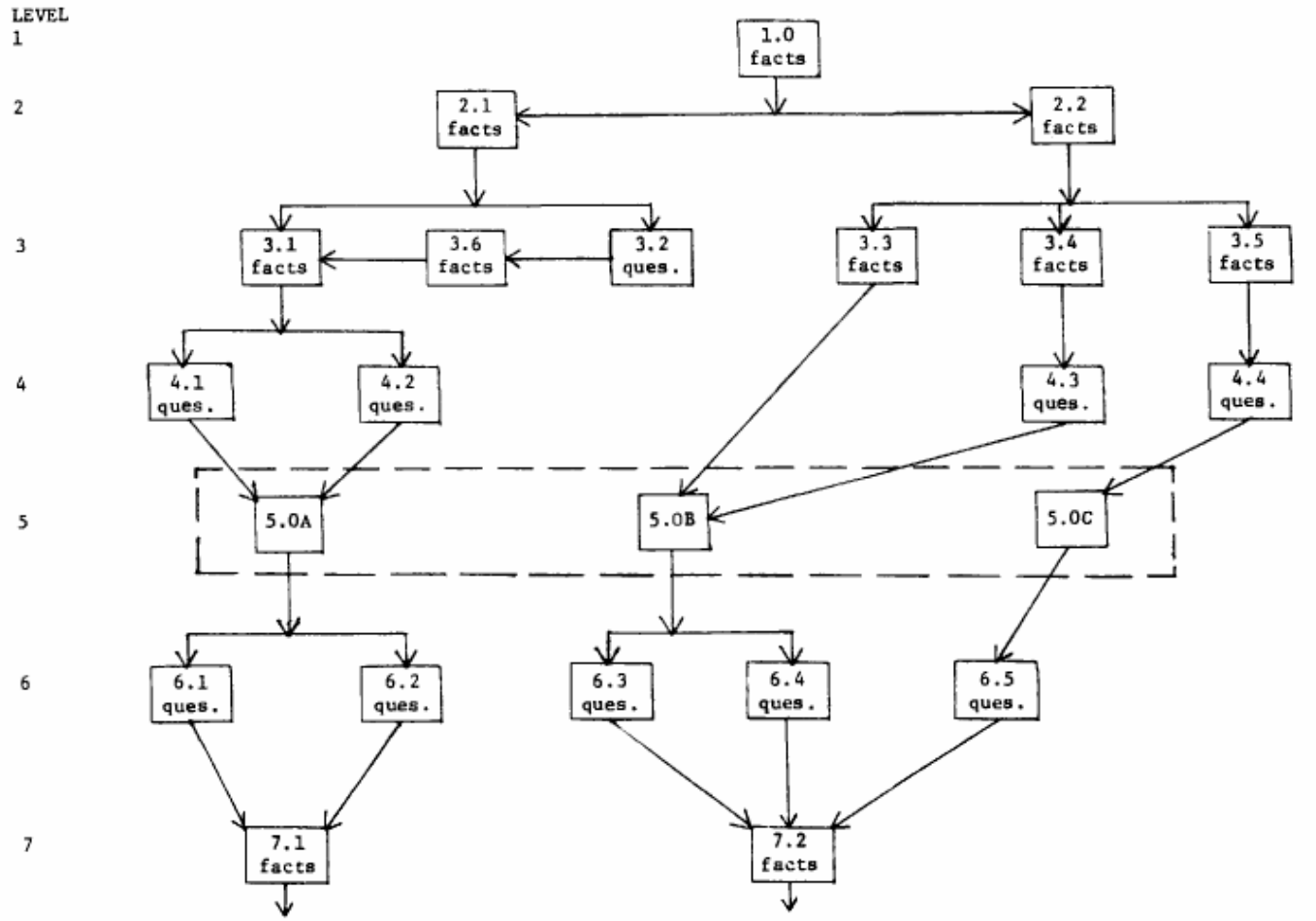
It is important to note that, while the node structure is hierarchical, it is not a tree. The structure is a network, since there are nodes which have more than one parent. For example, in figure 1 node 5.OB is related to two parent nodes, nodes 3.3 and 4.3. Also two or more nodes at the same level can share a common parent. Such nodes as 3.3, 3.4, and 3.5 are called twin nodes. It is the network aspect of the node structure which allows a model to simultaneously contain both constant and variable text. Constant text is represented at a level by one physical node having one sub-node, such that all nodes of previous levels have their link paths flow through this node. Variability is represented by not only multiple sub- nodes within a node, but also by twin nodes.

Information is presented to the participant beginning at level L At each level only one node of twin nodes is selected and within that node only one sub-node is presented to the participant. For example, in figure 1, after a set of facts at node 2.2 is presented, there are three mutually exclusive twin nodes; nodes 3.3, 3.4, and 3.5, any one of which may be selected. Every level need not be used as seen in figure 1, where there is a direct link between nodes 3.3 and 5.OB.

Questions also can be considered equivalent or mutually exclusive. If they are considered equivalent text, the questions are represented as sub-nodes. However, when the model designer desires to introduce two or more different kinds of questions, these are expressed as individual, mutually exclusive nodes. For example, the model designer may wish to divide questions into categories based on their degree of difficulty. In Figure 1 nodes 4.1 and 4.2 are sets of such questions on different link paths. Each question node may have up to two links extending from it, as shown by the dashed lines extending from node

[1] An exception to this is the branching structure in dialogs of CAI, where the information presented is dependent upon the participant's responses

FIGURE 1
EXAMPLE OF THE STRUCTURE OF AN INFORMATION FLOW



4.1. One link is for sub-nodal (equivalent) questions. The other link is for text containing the answer(s) to each question.

Figure 1 also demonstrates how the model designer can build supplementary link paths into a model to aid the slower learner. The link path from nodes 2.1 to 3.2 to 3.6 to 3.1 is a supplementary path, longer and perhaps more detailed than the more direct path 2.1 to 3.1.

Physical versus Logical Nodes

While it is true that all node structures and link paths are logical structures, we introduce the concepts of a physical and a logical node to handle a problem created by the fact that the node structure is a network and not a tree. The problem arises when link paths join at a coon node and that common node is not the end of these link paths. When one exits from a common node, different link paths may be needed to preserve logical consistency. For example, in figure 1 the dashed line around level 5 encloses constant text. Therefore, all link paths contain this same set of facts When level 5 is viewed in terms of textual content, there is one physical node, node 5.0. But, questions 6.1 and 6.2 are based on the facts given in node 3.1, while questions 6.3 and 6.4 depend upon facts from either 3.3 or 3.4. Thus the physical node 5.0 must have different link paths extending below it. To preserve such logical consistency the physical node 5.0 is divided into three logical nodes; 5.OA, 5.OB, and 5.0c.

Figure 1 shows how an informational node structure can simultaneously represent and maintain associational variability, associational necessity, and the prevention of associational inconsistency. Nodes 3.3, 3.4, and 3.5 demonstrate associational variability. The linking of nodes to constant text and the linking of answers to specific questions manifest associational necessity. The prevention of associational inconsistencies is maintained by non-intersecting link paths. For example, in figure 1 the question at node 6.4 can only be asked of a participant, if the facts at node 2.2 are first given.

INPUT OF NODE REPRESENTATIONS

After the model designer determines appropriate facts,

questions, and answers and finishes laying out the node structure of the model, the next step is to prepare this material for computer input to a Database Management System. There are two types of input. One is a description of the informational node structure. The other is the textual material of facts, questions, and answers.

Nodal Input

The logical associations between the nodes are expressed as inverted lists. Figure 2 illustrates how the node structure of

FIGURE 2
NODAL INPUT

```
& 1.0    2.1  2.2
& 2.1    3.1  3.2
& 2.2    3.3  3.4  3.5
& 3.1    4.1  4.2
& 3.2F   3.6
& 3.3    5.0B
& 3.4    4.3
& 3.5R   4.4
& 3.6    3.1
& 4.1    5.0A
& 4.2    5.0A
& 4.3    5.0B
& 4.4    5.0C
& 5.0A   6.1  6.2
& 5.0B   6.3  6.4
& 5.0C   6.5
& 6.1    7.1
& 6.2    7.1
& 6.3    7.2
& 6.4    7.2
& 6.5    7.2
```

figure 1 is represented for computer input. A delimiter (&) prefaces each node. The number immediately after the delimiter is a node number. All numbers after the first node number, until the next delimiter, are nodes linked to that node. There is no theoretical limit to either the total number of nodes or the number of nodes associated with a given node. The model designer may forte or restrict the selection of certain link paths. In figure 2 the F after node 3.2 represents a forced link path. The model builder may also wish to temporarily prevent certain facts or questions from being presented to the participants, until a later time, such as an examination period. The "R after node 3.5 in figure 2 shows such a restricted path. The node representation in figure 2 becomes input to a computer program called "the Builder." This program constructs the actual database linkages and is explained in the section, Models in a Database Environment.

Input of Facts, Questions, and Answers

Text in the form of facts, questions, and answers is then associated with nodes and entered as input to the Builder program. This program also writes the text into the database and constructs all necessary associations in light of the node structure inputted. Figure 3 is an example of text input based on part of figure 1. The numbers in parentheses at the right of each entry in figure 3 are for illustration only. Each entry in figure 3 contains a beginning delimiter (&&) a node number, a specification of whether the entry is a fact (F), a question (Q), or an answer (A), and an ending delimiter (&), followed by one or more lines of appropriate text. Each entry can have an unlimited number of lines of text. Sub-nodes within a node are designated for a fact or question by

FIGURE 3
TEXT INPUT

```
&& 1.0-F & lines of text        (1)
&& 2.1-F & lines of text        (2)
&& 2.1-F & lines of text        (3)
&& 2.2-F & lines of text        (4)
&& 3.1-F & lines of text        (5)
&& 3.2-Q & lines of text        (6)
&& 3.2-A & lines of text        (7)
&& 3.2-Q & lines of text        (8)
&& 3.2-A & lines of text        (9)
```

specifying the same node number for two or more entries. The second and third entries in figure 3 are sub-nodes within node 2.1. Answers are placed immediately after the question to which they refer. A question may have more than one answer. Entries 6 and 8 are sub-nodes within node 3.2.

The nodal input, the text input, and the Builder program are but part of the larger database system. A rationale for the database system and an explanation of how it functions are presented in the following sections.

PROGRAM AND DATA LNDEPENDENCE

Before the advent of Database Management Systems *(DBMS),* traditional file handling systems posed serious problems for application systems designers, when they faced the task of making structural changes to data. Usually a change, whether of data type or data association, necessitated changing not only the data, but also the programs which manipulated the data. The model builder has had this same problem when creating and maintaining computerized models. Frequently, models go through a number of evolutionary refinements before the creator is satisfied that the learning objectives will be met. When text and text substitution are to be incorporated into a model, the problem of changing and maintaining data associations is even greater. When changes must be made to either the data or its associations, the model builder does not wish to tear apart the algorithmic (program) part of a computerized model. The model builder wishes program and data independence; the ability to change data and its associations without having to modify the programs which utilize the data.

Database Management Systems provide this independence by acting as intermediaries between the physical database and the application programs. The binding process in DBMS is no longer between application programs and data associations, but between application programs and the DBMS on the one hand, and between the DBMS and the data associations on the other. In a DBMS associations between data are logically defined to the DBMS and physically controlled by the DBMS.

Program and data independence, which is provided by a DBMS, allows the model builder to define the logical associations between blocks of text and then to hand control of these associations over to the DBMS. The model builder is then free to concentrate on the program or algorithmic part of the model and not be burdened with having to control the data and its associations,

which support the model. In a DBMS environment the model builder can modify the program or the data independently of each other.

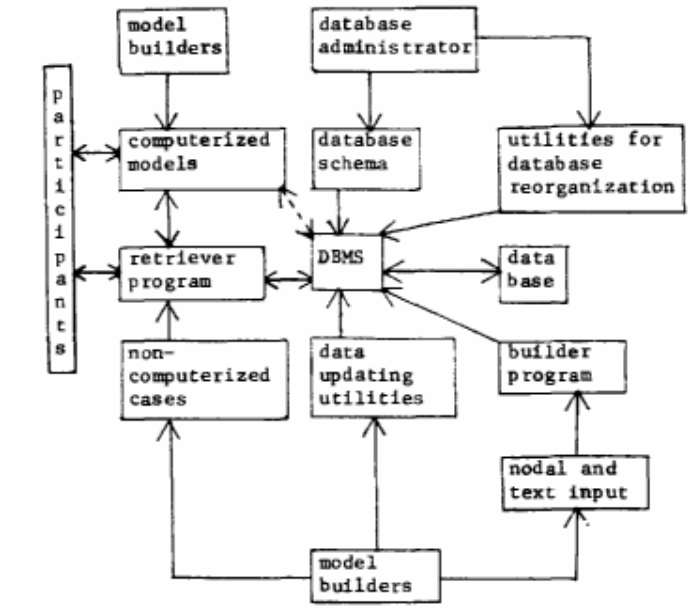
Models, cases, games, and other experiential learning techniques, which Incorporate textual variability, can also be viewed as having two components; an active, algorithmic, program component and a more passive data component. The relative mixture of these components differs, depending upon the proportion of algorithmic processes and text in a model. Cases and simulation models exemplify this difference. Once the textual associations have been built into a case and control handed over to a DBMS, the program part of the case is minimal. Cases, each containing their own substitutive text, can be presented to their participants under the control of a small driver program. Such a program, interacting with the DBMS, merely follows associational link paths through the database from the beginning to the end of a case, selecting allowable blocks of text. Simulation models, however, frequently contain numerous algorithmic processes, with continual transfer of control between data retrieval and the execution of algorithmic code.

MODELS IN A DATABASE ENVIRONMENT

Figure 4 illustrates the FAQSCAM system. This figure shows the relationship between models, the database, and the DBMS. This figure also shows the relationship between the model builders, the participants, and the Database Administrator. The FAQSCAM system is implemented on a Hewlett Packard 3000 Series II computer. The database is supported by HP's IMAGE DBMS software. System development was in four phases.

In phase I the Database Administrator (DBA), a faculty member, created one generalized database schema.

FIGURE 4
OVERVIEW OF THE DATABASE SYSTEM



A schema is a definition of the data items (fields), the data entries (records), data sets (files), and the logical associations between these as well as the data set capacities and access restrictions to the database. Using a vendor supplied utility program the Database Administrator then created an empty database.

In phase LI the DBA created the Builder program mentioned previously. The Builder program superimposes the unique associational structure of each model upon the one generalized database structure. The Builder program then writes text into the database.

In phase III the DBA created a database Retriever program which simplifies the model builders' interface with the DBMS. The Retriever program is composed of two modules. The first module handles data retrieval for cases and is designed to produce cases which are used as offline non-computerized cases. The algorithmic part of these cases is completely subsumed under the driver program. Under the control of a few parameters to this driver module, cases can be automatically generated as handouts for class. Such parameters include case number, password identification, number of output copies, and a designation of whether each copy for a class is to be textually similar.

The second module Interacts with computerized models, which have their own algorithms. The interaction between the models and the Retriever program involves processing control and the direction of the flow of facts, questions, and answers. The ordered transitions between a model's algorithmic processing and the retrieval of text is controlled by the model itself. The program component of a model has initial control and can abdicate degrees of control during its processing. Each model interfaces with the Retriever program through ordinary CALL statements, while the Retriever program interfaces with the DBMS through database CALL statements. Processing control can be temporarily abdicated by the program of a model for one or more nodes or levels. The extent of this abdication is given in the CALL statements. Then the Retriever program will follow link paths built into the database, access information from the database, and interact directly with the participant until control is returned to the model. A model's program retains processing control by generating calls to the Retriever program to place information from the database into working storage areas of the model's program. An alternative, but less frequently used, method of control, as shown by the dashed arrow in figure 4, is a direct database call to the DBMS. The Retrieval program, therefore, provides not only flexibility in terms of processing control, but also allows an easier interface for the model builder with the DBMS. This interface is analogous to the functioning of conversational database languages.

Information can be presented to the participants either from the model or in a specified format from the Retrieval program. The origin of the information is transparent to the participants. A model may direct the flow of information to itself or by default allow the retrieval of facts, questions, and answers to be pre8ented to the participants. A fact may flow to a participant or to the program of a model as input to a parameter of the model. Questions flow directly from the database to the Retrieval program and then to the participants. A participant's response to a question moves directly to the Retrieval program for comparison with the correct answer in the database. The correct answer and a resultant comparison are given to the participant and optionally to the model. Future extensions of the database system will incorporate an automatic grading module.

Updating and Reorganizing the Database

The model builder or the D8A can modify the database as shown in figure 4 through various utility programs. Changes to the database are inevitable, but occur in different frequencies depending upon the type of change. The addition, deletion, and editing of sub-nodal facts, questions, and answers are the most frequent types of database updating. The addition and deletion of entire nodes is next in frequency of occurrence. Less frequent is an associational change between two or more nodes. A Hewlett Packard conversational system called QUERY allows easy access to the database for perusal and most updating. The least frequent change is to the schema itself, which supports all the models. Such a reorganizational change is made by the DBA, using other vendor supplied utilities. First, the database is unloaded to tape, then structural changes are made to the schema, then the database is reloaded back to disk. The unloading and reloading process takes less than 15 minutes.

Data Sets of the Database

There are four data sets in the database. Figure 5 illustrates these data sets, showing the names of the most important data items, the link structures within and between the data sets, and an example of some of the data entries which would exist, based on the nodal and text input of figures 2 and 3. The data sets are of two types; master, and detail data sets. Data set 1 is a master data set. Data sets 2, 3, and 4 are detail data sets. The entries in the master data set are stored and accessed through hashing. Therefore, in figure 5 the physical order of the data entries in data set 1 is for illustration only. Each master entry also contains a number of chain heads. Each chain head contains the address of the first member of a set in a detailed data set. Other set members are linked within the detail data set. An entry in a master data set may be linked to entries in one or more detail data sets. In figure 5 by convention a dot symbolizes a pointer, while a zero designates an end of chain. For clarity of illustration the physical position of the pointers or links in figure 5, relative to the other data items in each data set has been altered.

Data set 1 contains an entry for every sub-node of a model. This data set has four data items and three pointers. The data items include MODEL number, NODE number, SUB--NODE number, and PATH TYPE. The latter is a designation of whether a link path is forced, restricted, or neither. Sub-nodes within data set 1 are associated with their respective text in data set 4 via the TEXT link in data set 1. All facts, questions, and answers are in data set 4. The first logical entry in data set 1 is the dummy node 0.0. This is the beginning of every model, since level 1 of any model may have more than one textual starting point. The hierarchical (HIER) link in data set 1 points to individual sets of twin nodes in data set 2. Pointers in data set 2 join together all the twin nodes of a common parent node in data set 1. For example, node 1.0 in data set 1 is linked to node 2.1 in data set 2, which in turn is linked to node 2.2 in data set 2. In data set 1 the SUB-NODAL link associates a node in data set 1 with its sub-nodes in data set 3. For example, node 2.1 in data set 1 starts a chain of two sub-nodes in data set 3. Thus, data sets 1 and 2 reflect the hierarchical and mutually exclusive associational structure of a model. Data sets 1 and 3 control logical equivalency.

In data set 1 rather than link every sub-node within a node to the same entries in data set 2, a dummy
sub-node 0 is created for each node and entered into data set 1 by the Builder program. In data set 1 sub- node 0 represents the entire node, while sub-nodes greater than zero represent actual sub-nodes. Since text is associated with each real sub-node, each nonzero sub-node in data set 1 has associated with it textual information in data set 4. Since data entries with a sub-node of 0 in data set 1 represent an entire node, these entries hold the chain heads for all associated twin nodes in data set 2 and all associated sub-nodes in data set 3.

These dummy entries are also used to indicate whether one of the twin nodes of a parent contains a forced link path. In data set 1 of figure 5 the data entry node 2.1 sub-node 0 specifies that there is a forced path in one of the twins of this parent node. The Retriever program can examine each twin node of 2.1 in data set 2 to find the forced path. Figure 5 shows that of the twins 3.1 and 3.2 in data set 2, the path selected must be from node 2.1 to node 3.2.

The data items of data set 2 include the MODEL number, the Predecessor (PRED) node and a Predecessor sub-node (PRED SUB-NODE) from data set 1, the NODE number, the TEXT TYPE and the PATH TYPE. The PATH TYPE designates one of the twin nodes as a forced or restricted node. TEXT TYPE is either Fact (F), Question (Q), or Undefined (U). If TEXT TYPE is fact or question, data set 1 contains a link to the actual text in data set 4. When more than one sub-node is associated with a node in data set 2, TEXT TYPE is tagged as Undefined. Data set 3 contains all the sub-nodes associated with each node and contains their individual text types. In data set 3 a path type of forced or restricted may be specified for individual sub-nodes.

Data set 4 contains the data items; MODEL number, NODE number,. SUB-NODE number, the text LINE NO., and lines of TEXT. This data set contains all facts, questions, and answers for all models. Text is allocated in blocks of 72 characters. Blocks of text are linked together in data set 4 to form variable sized facts, questions, and answers.

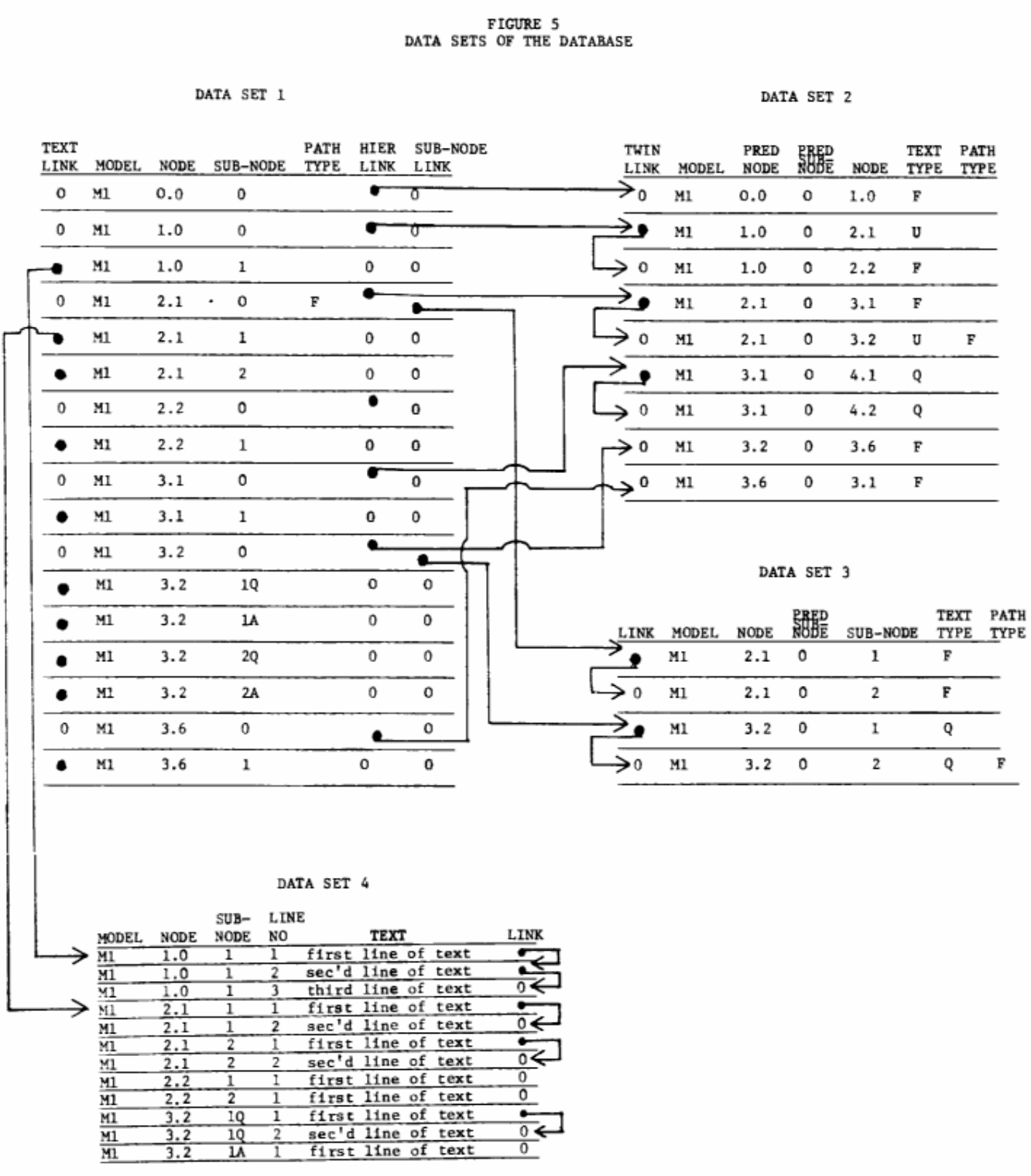
Generating a Database for a Single Model

When a model's textual associational structure is to be entered into the database, the Builder program first inputs a nodal representation as in figure 2 and builds data entries into data sets 1 and 2. For each node in the node representation a dummy node of O and one node with a sub-nodal designation of 1 are entered into data set 1. Each entry of the inverted list associated with each node is translated into a data entry and placed into data set 2. Each parent node in data set 1 is linked via the HIER link to one of the child nodes in data set 2. Other children of a parent are linked together within data set 2. Forced and restricted nodes are tagged; the parent in data set 1, the twins in data set 2.

When a node has more than one parent on the nodal diagram, this is represented in the database by one node entry in data set 2 for each parent. In figure 5 there are two nodes labeled 3.1 in data set 2. The first entry 3.1 associates the twin nodes 3.1 and 3.2. The second entry 3.1 has an association with the parent node 3.6 in data set 1.

Nodes like 5.0, which must be represented as logical and physical nodes, are entered into the database as follows. An entry is created in data set 1 for the physical node 5.0, sub-node 1. This entry will later contain an associational chain from the TEXT link to text in data set 4. Each logical node, 5.0A, 5.OB, and 5.0C, becomes an entry in data set 1 with a sub-

FIGURE 5
DATA SETS OF THE DATABASE



DATA SET 1

TEXT LINK	MODEL	NODE	SUB-NODE	PATH TYPE	HIER LINK	SUB-NODE LINK
0	M1	0.0	0			0
0	M1	1.0	0			0
	M1	1.0	1		0	0
0	M1	2.1	0	F		
	M1	2.1	1		0	0
	M1	2.1	2		0	0
0	M1	2.2	0			0
	M1	2.2	1		0	0
0	M1	3.1	0			0
	M1	3.1	1		0	0
0	M1	3.2	0			
	M1	3.2	1Q		0	0
	M1	3.2	1A		0	0
	M1	3.2	2Q		0	0
	M1	3.2	2A		0	0
0	M1	3.6	0			0
	M1	3.6	1		0	0

DATA SET 2

TWIN LINK	MODEL	PRED NODE	PRED SUB-NODE	NODE	TEXT TYPE	PATH TYPE
0	M1	0.0	0	1.0	F	
	M1	1.0	0	2.1	U	
0	M1	1.0	0	2.2	F	
	M1	2.1	0	3.1	F	
0	M1	2.1	0	3.2	U	F
	M1	3.1	0	4.1	Q	
0	M1	3.1	0	4.2	Q	
0	M1	3.2	0	3.6	F	
0	M1	3.6	0	3.1	F	

DATA SET 3

LINK	MODEL	NODE	PRED SUB-NODE	SUB-NODE	TEXT TYPE	PATH TYPE
	M1	2.1	0	1	F	
0	M1	2.1	0	2	F	
	M1	3.2	0	1	Q	
0	M1	3.2	0	2	Q	F

DATA SET 4

MODEL	NODE	SUB-NODE	LINE NO	TEXT	LINK
M1	1.0	1	1	first line of text	
M1	1.0	1	2	sec'd line of text	0
M1	1.0	1	3	third line of text	0
M1	2.1	1	1	first line of text	
M1	2.1	1	2	sec'd line of text	0
M1	2.1	2	1	first line of text	
M1	2.1	2	2	sec'd line of text	0
M1	2.2	1	1	first line of text	0
M1	2.2	2	1	first line of text	0
M1	3.2	1Q	1	first line of text	
M1	3.2	1Q	2	sec'd line of text	0
M1	3.2	1A	1	first line of text	0

node of 0. Each *of* these entries are then associated with their children in data set 2. Each logical node is also entered into data set 2 and each is associated with their respective parent nodes.

The Builder program next processes the text input (see figure 3), builds associations between data sets (1 and 3) and (1 and 4) and defines the text type for each entry in data set 2. Facts, questions, and answers are entered into data set 4. Each line of text is associated with a sub-node in data set 1 and linked together with other lines of text as a unit in data set 4. When equivalent text is first encountered, e.g. the second sub-node of node 2.1, an additional sub-node is created in data set 1 and two sub-nodal entries are created in data Set 3. The latter two sub-nodes are linked together, after having been linked to node 2.1 sub-node 0 in data set 1. The new entry in data set 1 is linked to its text in data set 4. Answers are associated with questions by sharing a common sub-nodal prefix in data set 1. In figure 5 sub-nodes 1Q and 1A of node 3.2 illustrate this association. Multiple answers to a question can be entered within the same block of text in data set 4 or in separate blocks at the discretion of the model builder.

Text Retrieval

An example of text retrieval based on the entries in figure 5 will demonstrate how the data sets of the database work together to assume logical consistency. In this example assume blocks of text are to be retrieved from the database and presented to a participant for the first three levels of the model. Note that there is a forced path through node 3.2 sub-node 2. The Retriever program begins by hashing into data set 1 on a key composed of the combined data items; MODEL number, NODE number, and SUB-NODE number. Ml-0.0-0 is the first key. After locating this entry its HIER link points to node 1.0 in data set 2. Since the twin link in data set 2 for this entry shows end of chain, node 1.0 has no twins. And since the SUB- NODE link in data set 1 has an end of chain for node 1.0 sub-node 0, the Retriever program then hashes back irto data set 1 under key Ml-1.0-l and follows the TEXT link from data set 1 to data set 4, where each line of text associated with that sub-node is retrieved and presented to the participant. When blocks of text were entered into data set 4, they were linked in ascending order by LINE NO.

The nodes on level 2 are then found by hashing into data set 1 using the key M1-1.0-0 to obtain the links to nodes 2.1 and 2.2 in data set 2. Since neither node is forced nor restricted, one of the nodes is chosen at random. Assume node 2.1 is selected. The TEXT TYPE value of Undefined in data set 2 specifies that there is more than one sub-node at node 2.1. The key M1-2.1-0 is used to hash into data set 1 to locate the two sub-nodes of node 2.1 in data set 3. Again, neither sub-node is forced nor restricted, so one of these nodes is chosen at random. If sub-node 2 were selected, the key Ml-2.l-2 is used to hash into data set 1 to retrieve the associated text in data set 4.

then the key M1-2.1-0 is used to enter data set 1 to find the children of node 2.1. The PATH TYPE of “F” in data set 1 for entry M1-2.1-0 cells us that one of the twin nodes of the parent node 2.1 has a forced link path. In data set 2 all twin nodes of node 2.1 will be accessed until the forced node is located. When the entry for node 3.2 is located in data set 2, the key M1-3.2-O is used to hash into data set 1 to find the links to the two sub-nodes of 3.2 in data set 3. Since the forced path is through sub-node 2 of node 3.2, this sub-node is selected. This entry *is* a question. Therefore, to locate the text associated with this question the key Ml-3.2-2Q is used to hash into data sex 1 to find the TEXT link to data set 4. The answer to this question is easily retrieved by changing the key to M1-3.2-2A and hashing back into data set 1 to pick up the link to data set 4. This process is continued until control is handed back to a model’s program or until the end of the model is reached.

Extending the Database to Multiple Models

The power of the FAQSCAN system is in its capability to simultaneously support multiple cases and models, each having its own nodal structure. This one physical database can contain entries from more than one case or model. Each model is logically separated from the others, although entries from different models may be physically adjacent to each other in the database. The part of each key containing the model number differentiates the models. Once hashing occurs into data set 1 for a given model, entrance into the other data sets is solely by way of associational pointers. Since the Retriever program usually controls accesses to the database, human accessing errors such as slipping from one model into another are avoided.

Database Security

Database security is maintained in a number of ways. Passwords must be given to the Retriever program to enable the accessing of entries from the database. Attempting to dump the physical contents of the database without authorization is prevented by both the operating system and the DBMS. Also the physical randomness of the models’ entries might discourage such attempts. As an added security measure encryption of the text in data set 4 is a planned extension to the FAQSCAN system.

REFERENCE

[1] Kroenke, David, Database Processing (Chicago: SRA, 1977).