Discussion Paper No. 391

A Data Base for Operations Research Models

by

Edward A. Stohr
and
Mohan R. Tanniru*

Northwestern University
Graduate School of Management
Evanston, Illinois 60201

July, 1979

*Graduate School of Business, University of Wisconsin, Madison, Wisconsin 53715

A DATA BASE FOR OPERATIONS RESEARCH MODELS

Edward A. Stohr
Graduate School of Management
Northwestern University
Evanston, Illinois 60201

and

Mohan R. Tanniru
Graduate School of Business
University of Wisconsin
Madison, Wisconsin 53715

ABSTRACT

This paper develops the design of a data base system to support operations research models in the context of an integrated planning system involving a number of different users and computer programs. The requirements for such a system are described, a 'network' data base schema is developed and the schema and command language are illustrated through a specific example.

1. INTRODUCTION

Computerized planning systems and models have been used increasingly in recent years [11]. However there are relatively few examples of truly integrated systems involving multiple corporate departments and several layers of decision-making from the most detailed planning at the production level to more aggregate financial summaries and analyses for budgeting and strategic decision-making. Much research has been concerned with the complexity of such systems and the ways in which computers can provide information and analytic support to the decision-makers. The general requirements for a decision support system (DSS) have been outlined in [1], [2] and [9]. In this paper we describe how data base management system techniques can be used to keep track of the complex information flows between different decision-making units and computer programs. The objectives are: (1) to increase the integrity of the planning process by assuring that the correct inputs are used by the various computer programs and by providing an 'audit trail' and (2) to aid the users by facilitating data retrieval and processing and by providing an environment in which it is easy to test and keep track of the effects of different model assumptions and data values during sensitivity analyses.

The advantages of Data Base Management Systems (DBMS) in assuring data integrity and security, reducing data redundancy, maintaining multiple relationships between data items and providing the basis for powerful retrieval languages are well-documented [10]. To date applications have been primarily in an information retrieval or data processing environment. The 'data base approach' is to design the logical structure (schema) for the data to allow for the estimated data requirements of different users and programs. For modeling and planning systems this might appear to be a difficult task since the data relationships to be maintained differ from model to model and in the context of any one model are often ascertained only through extensive experimentation aimed at finding the most important variables affecting the objective of interest. What is needed here is a retrieval language and data base system which is capable of dynamically transforming logical data structures to the needs of a particular model at a particular time. By logical structures we mean the data base 'schema' (system administrators view) and more particularly the class of 'subschemas' (programmer's views) which can be derived from the schema. A mapping language for this purpose has been outlined in [4].

In this paper we concentrate on a different aspect of the use of DBMS technology to support decision-making—namely its use to help manage the data flows and models during sensitivity analyses in both the simulation and production use of a planning system. The computer programs themselves are represented by record occurrences in the data base which can be retrieved and executed by high level commands. We describe a schema based on the CODASYL Data Base Task Group (DBTG) 'network' approach [5] which is not problem-specific and can be used in a wide variety of applications. Related work appears in [3]. The present paper is an extension and illustration of previous work on this subject [13].

2. REQUIREMENTS FOR A COMPUTERIZED DSS

Some of the problems which must be faced in the construction of an integrated planning system are:

(1) The necessity to transform data into the specific format required by a particular software package or computer program.

(2) Difficulties in data naming conventions when different programs and packages are to be coordinated.

(3) The fact that the planning system might require alternative mappings of the same data inputs into (different values of) the same output variables. Common examples occur when the planning system is to be capable of testing outcomes under different depreciation or inventory management methods.

(4) The necessity to coordinate a large number of programs which interact with each other through their inputs and outputs and must be executed in certain sequences to ensure valid results.

(5) A complex data integrity problem must be solved in that data redundancy is a desired norm rather than something to be avoided. The planning system should be capable of generating and maintaining alternative values of output variables at each of several stages and levels of the planning process. It should be possible and simple to trace, display and, if necessary, adjust the entire history of program runs, assumptions and data values which underly any output. This is useful, for example, in iterative budgeting processes involving negotiation between corporate divisions and headquarters.

(6) Because many users may be involved in the planning process adequate security measures must be provided. Many users will have restricted rights to access and modify data and to initiate program runs.

(7) Because planning systems must be capable of evolution and extension they should be built in a modular manner and easy to modify and maintain.

(8) Adequate programmer and user documentation should be available.

(9) Since many users will be involved with differing levels of programming expertise the command language must be easily understandable and forgiving of mistakes. Users should feel 'comfortable' with the system in the sense that they know where they are at all stages and can have confidence that the data supplied them is up-to-date and correct. In this connection the planning system should facilitate the communication process so that a given user can easily determine the relevant actions of other users—a form of electronic mail.

In the following section we outline the design of a system which attempts to meet many of these requirements.

3. BACKGROUND AND TERMINOLOGY

The data base design will be illustrated using the network approach [5] primarily because such systems are more readily available commercially. An equivalent system design could readily be implemented in a hierarchical [8] or relational [6] DBMS.

In the network approach the schema or logical description of the data involves data items, record types and set types. A data item is the smallest unit of data which has a meaning. Subsets of data items can be associated with each other by specifying that they belong to the same record type. Record types are in turn related to each other via the 'set' construct which constitutes a one-to-many relationship between an 'owner' record type and a 'member' record type. These constructs are sufficient to represent data relationships of arbitrary complexity (see the graphical representation of the proposed data base schema in Figure 2). At any time the stored database (or 'realization') of the schema will consist of an arbitrary number of 'instances' of each record-type corresponding to the values assumed by the data items (see Figures 6 and 8). After the schema has been designed and the data loaded, retrieval according to the DBTG specification involves the use of a data manipulation language (DML). The DML gives the user the ability to traverse the data base network in a record-by-record manner. Since this is a fairly 'low level' language, unsuitable for non-expert users, the proposed command language will contain many 'pre-programmed' retrieval programs.

The planning system is being implemented in the APL programming language using the EDBS Data Base Management System, [7]. The schema is described using a 'Data Definition Language' (DDL). A partial definition of the proposed schema using the EDBS DDL is illustrated in Figure 1.

In the implementation advantage has been taken of some of the features available in APL which are not duplicated in other languages:

(1) The language possess an 'Execute' operator which allows the construction of commands as character strings which can subsequently be executed. Using this feature character matrices consisting of sequences of commands to open and read files and execute programs ('functions' in APL) can be stored in the database and retrieved and executed as required.

(2) Functions can be converted to character matrices, stored in the data base system in this form and subsequently retrieved. A 'group' of such functions can be stored and retrieved in a similar manner. In the following 'Function Group' will be equivalent to a program (with its 'main' and sub-programs) in another programming language.

(3) Data arrays of any dimension can be stored as 'components' of a random access file. Furthermore it is a simple matter to encode character and numeric arrays into a character string which can be stored as a single file component.

The above capabilities of APL have been used to simplify the design and will simplify the exposition which follows. However the logical design could be implemented in other programming languages with only minor changes.

Before describing the schema and the processing commands which operate on the database it is necessary to define some of the terms we will be using. A '<u>Planning System</u>' consists of a number of <u>Models</u>. Each Model in turn consists of a number of <u>Processes</u>. A Process involves the use of a <u>Function Group</u> in the context of a model. A Function Group (= program) executes a sequence of operations on the data and represents a computational step. A Function Group may consist of one or more cooperating APL functions. A Process is associated with only one Function Group however the same Function Group may be used in a number of different Processes in the same or different models. The specification for a Process includes a definition of the environment in which its associated Function Group is to operate including the naming of its inputs and outputs (with renaming of variables where necessary)—see Figure 4. Thus a Function Group might execute a linear programming algorithm and might be used as a Process in both a production planning Model and a cash management Model.

To maintain data integrity and to ensure that all inputs which affect the results of a Process are recorded all data is stored on direct access files unless erased by the user. This includes data entered in response to program prompts if that data affects the values of output variables. It is necessary to distinguish between different kinds of data inputs to Processes. '<u>Exogenous Input</u>' to a Process is data previously output by another process. Since all data entry functions (e.g. loading of data from a magnetic tape) are performed by Processes almost all data is exogenous. The most important exception to this occurs with processes accepting on-line input from users. Such data is referred to as '<u>Own Input.</u>'

A number of algorithms (including corporate planning simulation packages) operate by interpreting a logical specification of the users problem. Often data values (parameters) are imbedded in this logical statement however it is better practice for many reasons to make the logical statement as independent as possible of the data by expressing parameters as variables. To emphasize this and to increase the self-documenting capabilities of the system this form of 'input' to algorithms is treated separately and is referred to as 'Problem Statement.'

During the execution phase of a Model the data inputs and outputs of each Process are recorded in the data base. The initial set of data for a Process will be referred to collectively as a Base Case. An adjustment of the values of a Base Case for sensitivity testing or other purposes is a Case. To save data storage a Case is represented only by the APL instructions which transform values of the Base Case data. A Run is the record of a computation performed for a specific Case of a Process. There is a 1:1 correspondence between a Run and its associated output data.

4. A DATA BASE SCHEMA FOR DECISION SUPPORT

The database design includes four APL files. The Data Dictionary File contains definitions of: (1) the Models, (2) the Processes including their input and output variables and Problem Statement definitions and (3) the Report Definitions (formats) for output variables. The Data Base Directory maintains the logical relationships between Models, Processes and the Base-cases, Cases and Runs which constitute the computational history of each Process. The term 'Directory' is used because the data base records primarily contain 'pointers' to the Data Dictionary and Model Data Files in which the data values themselves are stored. The Data File for a Model contains the actual input and output data values and character matrices describing the nature and purpose of each Basecase, Case and Run. The Function Library contains the code for all functions used by the Planning System; either individual functions or complete Function Groups may be accessed. All functions must be registered in the Function Library before they can be used by a Process. This process of registration also enforces programmer documentation. The Function Library is part of another 'system development' system and will not be further described here.

Figure 1 below shows part of the schema definition for the Data Base Directory in the DDL of EDBS while Figure 2 gives a graphical representation of the schema.

```
DATA BASE PLANSYS NETWORK
RECORD MODEL
DATA-ITEM MNAME STRING(10):KEY
DATA-ITEM FNAME STRING(6)
DATA-ITEM M-DESCR-PTR NUMERIC(3,0)
END
RECORD BASECASE
   .    .   .
   .    .   .
   .    .   .
SET M-P
OWNER IS MODEL
MEMBER IS PROCESS
END
SET P-B
   .    .   .
   .    .   .
   .    .   .
```

Figure 1
Part of Schema Definition

Underlined data item names in Figure 2 are record 'keys' consisting of user-given names for the record instances. The suffix 'PTR' indicates that the data item is a 'pointer' (value of a component number in another file). Data item names ending with 'DESCR-PTR' are pointers to user-given character matrices describing the associated record instance. The meanings of the other data items are:

MODEL.FNAME - name of model Data File

PROCESS.FUNCGRP - name of APL workspace.name of Function Group

PROCESS.EXEC-PTR - pointer to character matrix representing the sequence of APL commands required to execute the Function Group.

CASE.C-EXEC.PTR - pointer to character matrix giving the APL commands representing the changes to the Basecase associated with this particular Case.

CASE.PS-NAME - the PSNAME of the Problem Statement associated with this Case.

RUN.DATA-PTR - pointer to the data output from the Run.

PROB-STATEMENT.PS-PTR - pointer to the character matrix containing the Problem Statement.

PROB-STATEMENT.L-NL – a binary variable indicating whether or not the Planning System is to automatically read the Problem Statement into the user's workspace. If not this will be done under program control.

INPUT.L-NL – binary variable indicating whether or not the Planning System is to automatically load the input data into the user workspace. If not then the data will be read under program control.

INPUT.O-EX – binary variable indicating that the input variables are 'Own' or 'Exogenous'.

INPUT.I-VAR-LIST – name of model.name of Process: list of the input variable names followed by their dimensions in parentheses (see Figure 6).

OUTPUT.O-VAR-LIST – list of output variables and dimensions.

REPORT-DEF.R-DEF-PTR – pointer to the character string encoding of the report format (headings, labels, etc.).

The pointers in the records 1 through 6 in Figure 2 point to the Data Dictionary File. These records are stored during the Model Definition Phase. Records 7 through 10 record the history of processing and their pointers point to the Model Data File. These records are stored during the Model Running Phase.

The Planning System is designed to facilitate automated running of all or some of the Processes within a Model at the Option of the user. The PROCESS records within the M-P set are stored in their 'natural' precedence order (the first record in the set is the first process to be run, etc.) The Planning System has the information to ensure this (see Section 5). The records in the P-B and B-C sets are maintained by the Planning System in LIFO order under the assumption that the latest Basecase and Case are the most likely to be used. If this is not the case the user can indicate the particular Basecases and Cases to be used prior to running the Model (or any valid subsequence of Processes).

The various Cases relate the input and output data generated during sensitivity analyses. The C-EXEC-PTR for the last CASE record in the B-C set (which is loaded at the same time as its associated BASECASE record) actually points to a character matrix of APL instructions which loads the entire set of data for the BASECASE. The C-EXEC-PTR's for other CASE records point to

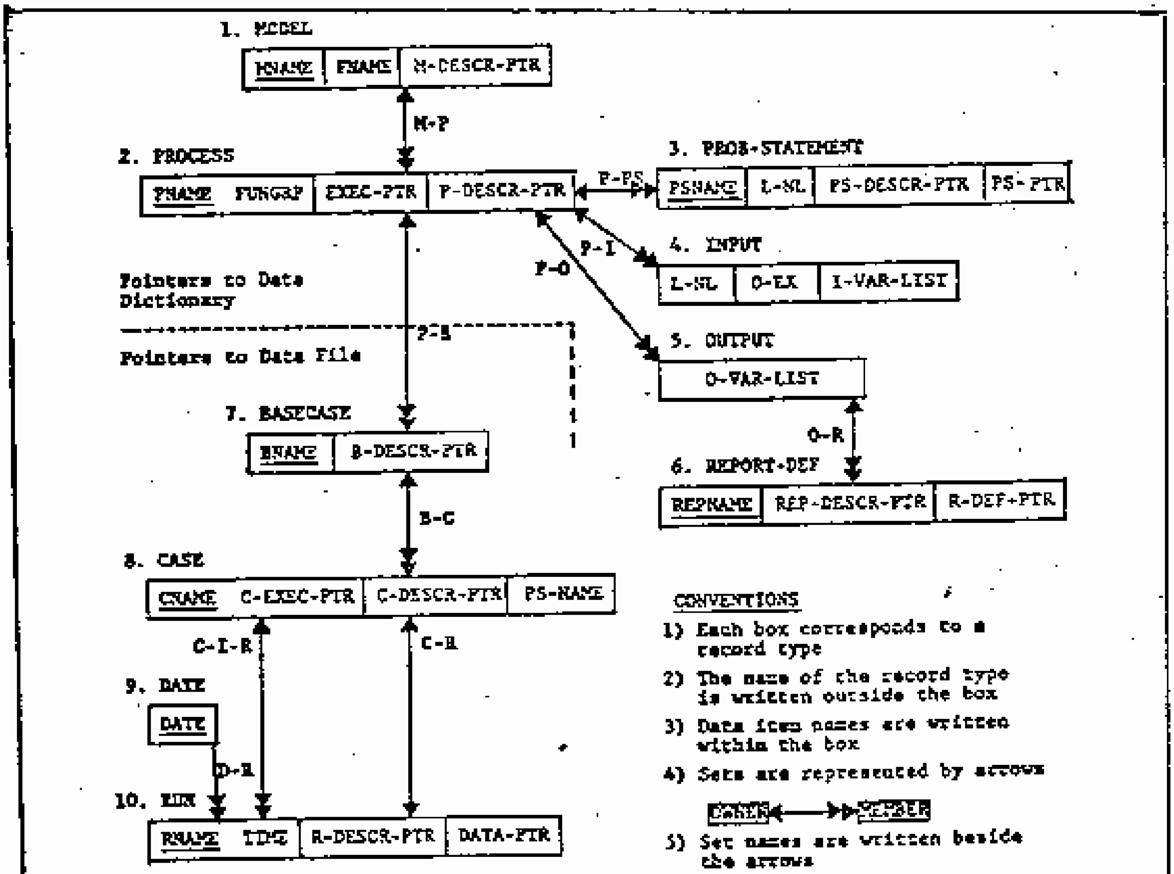


Figure 2
Schema for Planning Data Base Directory

character matrices of the APL instructions which convert the original Basecase data values to those for the Case. Thus the last record in the B-C set is normally accessed first and successive 'Prior' records are obtained until the desired Case is reached. The PS-NAME field in the Case record cross-references the PROB-STATEMENT record associated with the version of the Problem-Statement (if any) used by the Case. The C-I-R set associates a Case of a Process with the Exogeneous Inputs used (these inputs are by definition the outputs of Runs of other Processes). The DATE record type and D-R set were introduced to facilitate retrieval in response to queries involving dates. For example, one might wish to display yesterday's RUN records or purge last month's processing history.

One or more Problem Statements are associated with a Process through the P-PS set. Note also that the APL instructions pointed to by C-EXEC-PTR may contain editing instructions to allow experimentation with various changes to a Problem Statement. The P-I set contains one or more INPUT records. The I-VAR-LIST for Exogeneous INPUT data is a character string prefaced by the Model name (if from a different model) and Process name of the Process which generates the data. Array variable names are followed by the array dimensions in parentheses to allow data validation (see Figure 6).

The P-O set will often contain only one member record. However, the set of output variables may be partitioned for reporting purposes. The O-R set associates one or more alternative Report Definitions with the variable list of the owner record of the set occurrence. A generalized report generator system accepts a report definition input interactively by the user and encodes it as a character string. This is loaded into the Data Dictionary during the 'registration' process (see below) and is pointed to by the R-DEF-PTR data item. Standardized input or output variable data displays can be obtained automatically at any time without the necessity for storing a report definition.

5. THE MODEL DEFINITION PHASE

During this phase the data dictionary is loaded with data representing definitions of all components of the Model. At the same time instances of data records of types 1 through 5 are loaded into the Data Base Directory and any user-defined functions and Function Groups are loaded into the Function Library. The loading of this data is accomplished by 'registering' the Model, its Processes, Function Groups, Report Definitions and (optionally) Global Variables using an interactive 'REGISTER' function as outlined below. A DOCUMENT function uses

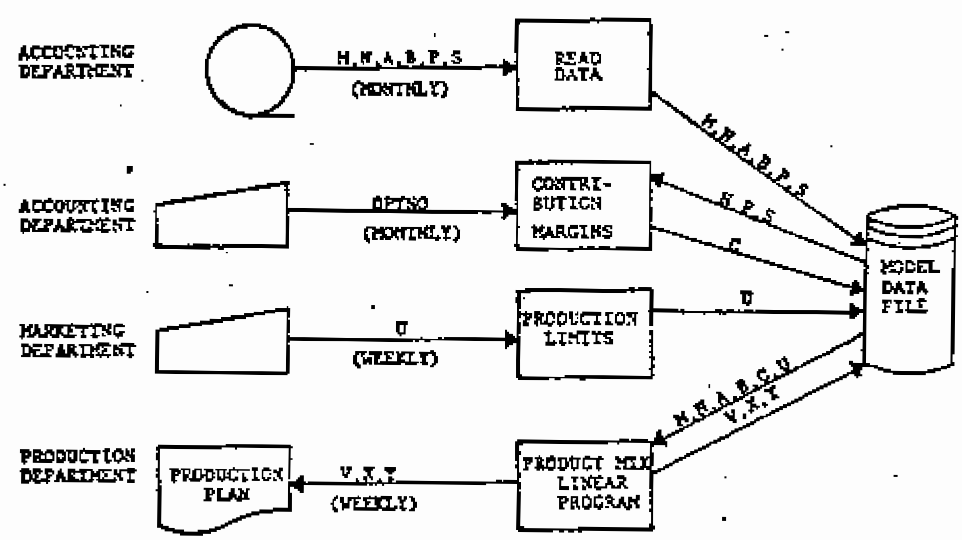


Figure 3
Data Flow for Product Mix Planning Model

```
STEP
  1       )LOAD PLANSYS
  2       REGISTER 'MODEL'
      NAME=PRODUCTMIX
      FILE=PMIXFILE
      MODEL DESCRIPTION:
      THIS MODEL IS RUN WEEKLY TO PRODUCE A PRO-
      DUCTION PLAN FOR THE FOLLOWING WEEK.  THE
      BASIS OF THE MODEL IS A LINEAR PROGRAM
      (SEE PRODMIXLP DEFINITION).  PROCESSES,
      RESPONSIBILITIES AND DATA FLOWS ARE AS
      FOLLOWS:
      1)  PROCESS READDATA:
      ACCOUNTING, MONTHLY CYCLE.  PROCESS READS
      VALUES OF VARIABLES M,N,A,B,P,S  FROM A
      MAGNETIC TAPE AND STORES THEM IN MODEL
      DATA FILE.
      2)  PROCESS CMARGINS:
      ACCOUNTING, MONTHLY CYCLE. PROCESS COM-
      PUTES CONTRIBUTION MARGINS, C, UNDER SEV-
      ERAL OVERHEAD ALLOCATION OPTIONS.  IN-
      PUTS:  N,P,S.  OUTPUTS:  C
      3)  PROCESS PLIMITS:
      MARKETING, WEEKLY CYCLE.  PROCESS INVOLVES
      INTERACTIVE INPUT OF MAXIMUM DESIRABLE
      PRODUCTION QUANTITIES, U, FOR EACH PRO-
      DUCT BASED ON WEEKLY SALES FORECASTS.  IN-
      PUTS:  U.  OUTPUTS:  U
      4)  PROCESS PRODMIXLP:
      PRODUCTION, WEEKLY CYCLE.  PROCESS RUNS A
      LINEAR PROGRAM TO DETERMINE PRODUCTION
      PLAN.  INPUTS:  M,N,A,B,C,U.  OUTPUTS:  V,
      X,Y
      DEFINITIONS OF MODEL VARIABLES:
      M=NUMBER OF RESOURCE CONSTRAINTS
      N=NUMBER OF PRODUCTS
      A=(M×N) MATRIX OF PRODUCTIVITY
      COEFFICIENTS
      B=M-VECTOR OF RESOURCE CONSTRAINT LIMITS
      P=N-VECTOR OF PRICES OF PRODUCTS
      S=N-VECTOR OF DIRECT COSTS
      C=N-VECTOR OF CONTRIBUTION MARGINS
      U=N-VECTOR OF MAXIMUM DESIRED PRODUCTION
      QUANTITIES
      V=$VALUE OF OPTIMAL PRODUCTION PLAN
      X=N-VECTOR OF OPTIMAL PRODUCTION QUANTI-
      TIES FOR PRODUCTS
      Y=M-VECTOR OF SHADOW PRICES
  3       REGISTER 'PROCESS'
      NAME=PRODMIXLP
      FUNCTION GROUP=MATHPR.LPGENGRP
      EXECUTION COMMAND SEQUENCE:
      RUNLP

      PROCESS DESCRIPTION:
      THE OPTIMAL PRODUCTION MIX FOR THE FOLLOW-
      ING WEEK IS FOUND BY A LINEAR PROGRAM.
      THE LPGENGRP IN THE MATHPR WORKSPACE CON-
      TAINS A TABLEAU GENERATOR AND WILL EXECUTE
      THE LINEAR PROGRAM.  THE DEFINITION OF THE
      PRODUCTION MIX PROBLEM IS STORED IN THE
      CHARACTER MATRIX, PRODMIXDEF, IN THE MODEL
      DATA FILE.

      INPUT VARIABLE LISTS:
      LOAD/EXOG/READDATA:M,N,A(M×N),B(M)
      LOAD/EXOG/CMARGINS:C(N)
      LOAD/EXOG/PLIMITS:U(N)≡L(N)

      PROBLEM STATEMENT NAMES=PRODMIXDEF
      PROBLEM STATEMENT DESCRIPTION:
      STANDARD LINEAR PROGRAMMING FORMULATION

      OUTPUT VARIABLE LISTS:
      V,X(N),Y(M)

      REPORT NAME=PRODPLAN
      REPORT DESCRIPTION:
     WEEKLY PRODUCTION PLAN
```

Figure 4
Definition of Model and Processes
(computer prompts are underlined;
user commands are indented)

this data to automatically provide user- and programmer-level documentation of the model.

The Model Definition Phase will be illustrated by a simple application of linear programming to a production planning problem. The data flows involved are shown in Figure 3. The model itself is explained during the Registration Process which is illustrated in Figure 4. In step 1 the Planning System is loaded into the user's workspace. In step 2 the Model is 'registered'. Step 3 shows the registration of the Process, PRODMIXLP, which runs the linear program. The LPGENGRP Function Group referred to accepts a Problem Statement in the form of a character matrix PRODMIXDEF which defines the problem in 'sigma notation' format (see Figure 5) [12].

```
*LINEAR PROGRAM FOR PRODUCTION MIX PROBLEM
*
VAR=X(J),J IN 1 THRU N
*
MAXIMIZE
S C[J]X(J)
J IN 1 THRU N
*SUBJECT TO:
FOR I IN 1 THRU M
S A[I;J]X(J)≤B[I]
J IN 1 THRU N
*
FOR J IN 1 THRU N
X(J)≤L[J]
*END
```

Figure 5
Problem Statement for Process PRODMIXLP
(The symbol 'S' substitutes
for 'Σ' in algebraic notation)

Note that the Model uses a variable, U, for the desired upper limits for production whereas the linear programming uses the variable L. This difference in naming conventions is resolved by the 'equivalencing' phrase U(N)≡L(N) in the Input Variable List for the PRODMIXLP Process (Figure 4). A realization of the Data Base Directory schema at the end of the Model Definition Phase is shown in

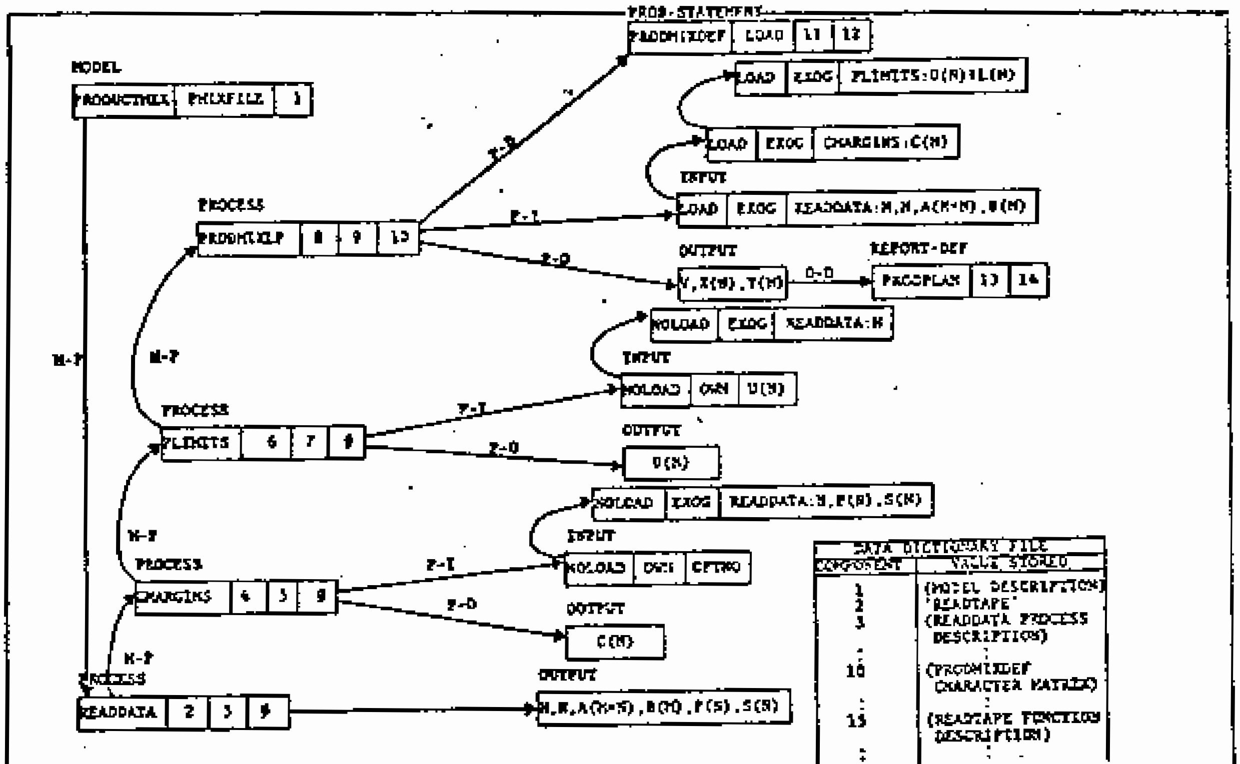


Figure 6
Realization of Part of Database Directory After Model Definition Phase
(The symbol ∅ indicates a null value)

Figure 6. The four processes shown in the Figure are called READDATA, CMARGINS, PLIMITS and PRODMIXLP and have been stored in the M-P set in a feasible execution sequence.

6. THE MODEL RUNNING PHASE

During this phase the various Processes within a Model are executed and the complete history of all the computations is recorded in record types 7 through 10 of the schema.

STEP

1.)LOAD PLANSYS
2. READYMODEL 'PRODUCTMIX'
3. LOADPROCESS 'READDATA'

TO EXECUTE PROCESS TYPE: READTAPE

4. HOW 'READTAPE'

(the previously 'registered' explanation of readtape is accessed in the data-dictionary and printed)

5. READTAPE

(this function reads the data from the magnetic tape and displays relevant summary statistics or the raw data itself)

6. STORERUN 'JUNEDATA'

DESCRIPTION=JUNE UPDATE FROM MAGNETIC TAPE.
(RUN JUNEDATA STORED 06/28/79) 10:03 A.M.)

7. ERASEPROCESS 'READDATA'
8. LOADPROCESS 'CMARGIN'

TO EXECUTE PROCESS TYPE: COMPC

STEP

9. COMPC

FUNCTION USES DATA FROM PROCESS READDATA-
LATEST OUTPUT IS:
RUN=JUNEDATA,06/28/79 10:03 A.M.
DESCRIPTION=JUNE UPDATE FROM MAGNETIC TAPE
OK? Y
DESIRED OPTION NUMBER=3
STORE BASECASE? Y
NAME=JUNEMARGINDATA
DESCRIPTION=DATA FROM NORMAL UPDATE OF PLANNING DATABASE
(BASECASE JUNEMARGINDATA STORED)
(CASE JUNEMARGINDATA STORED)
(the COMPC function now computes and displays the contribution margins according to option 3)
STORE RUN? Y
NAME=JUNEMARGINS
DESCRIPTION=JUNE MARGINS COMPUTED ACCORDING TO OPTION NO. 3
(RUN JUNEMARGINS STORED 06/28/79 10:08 A.M.)

10. ERASEPROCESS 'CMARGIN'
11. CLOSEDATABASE

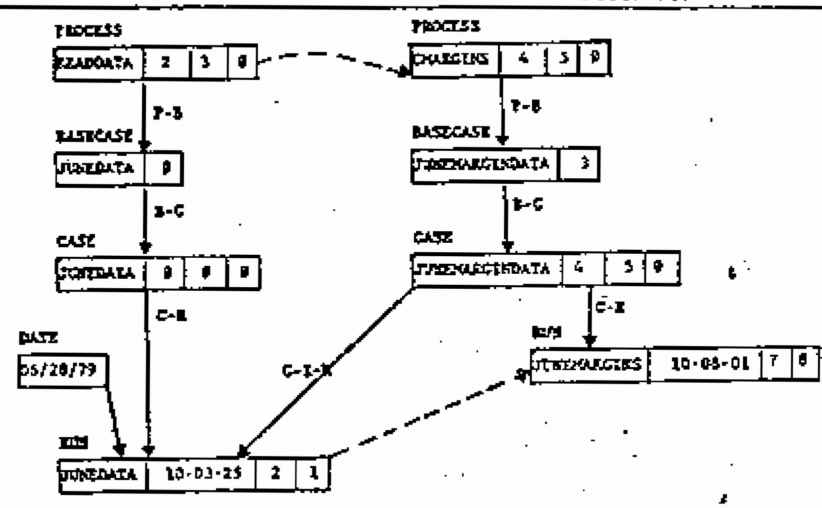
Figure 7
Running Phase of Model

(User Commands are Indented, Computer Prompts are Underlined)

Because of space limitations only a few commands will be illustrated. Continuing the previous example, Figure 7 illustrates the computer interaction of the Accounting Department running processes READDATA and CHARGINS which use the functions READTAPE and COMPC respectively. The Planning System is run in 'manual' mode, i.e. each process is loaded separately into the user's workspace and the associated function is executed by the user. Both functions use Planning System Commands internally to read the data and store the results for the run.

Figure 8 shows the instances of record types 7 through 10 after this interaction. The following is an explanation of the steps in Figure 7:

1.)LOAD PLANSYS loads 'core' planning system functions and 'opens' the Function Library file.
2. READYMODEL opens the Database Directory, the Data Dictionary and the PMIXFILE Data File for the PRODUCTMIX model.
3. LOADPROCESS accesses the Process record for READDATA. It uses the FUNCRP field to load the correct Function Group into the workspace and the EXEC-PTR field to access and print the execution instructions for the process which were stored in the Data Dictionary. Since the P-I set is empty LOADPROCESS halts.
4. The HOW function can be used at any time to retrieve and print previously registered definitions and explanations of the Model, Processes, Function Groups and Global Variables.
5. The READTAPE user function issues commands to the computer operator to load the relevant magnetic tape and reads the data into the next available 'slot' in the Data File using the 'format' given in the O-VAR-LIST field of the OUTPUT record corresponding to the current PROCESS record. The Component Number in which the data has been placed is stored in a global variable for use in any subsequent STORE command.
6. The STORERUN prompt asks if a RUN record is to be stored in the Data Base Directory. The RUN. DATA-PTR field gets the value of the Component Number from the Global Variable of the previous step. The user's description is stored as the next available component in the Data File and the associated Component Number in the R-DESCR-PTR field of the RUN Record.



DATA FILE

COMPONENT	VALUE STORED
1	VALUES FOR H,N,A,B,P,S
2	(DESCRIPTION OF READDATA RUN)
3	(CHARGINS BASECASE DESCRIPTION)
4	'TREAD FNO.3'
5	VALUES FOR N,P,S,OPTNO
6	(CHARGINS CASE DESCRIPTION)
7	(CHARGINS RUN DESCRIPTION)
8	VALUE OF C

Figure 8
Portions of the 'Data Base Directory' and 'Data File' affected by the June 25th Execution of the READDATA and CHARGIN Processes

(The symbol Ø indicates a null value)

7. The ERASEPROCESS command clears the functions and global variables related to the READDATA Process from the workspace.

8. The CHARGINS Function Group (COMPC) is loaded (see step 3 above). In this case there is an Exogenous Input. However since the L-NL field equals 'NOLOAD' the LOADPROCESS function does not itself read the data into the workspace as global variables—this will be done during the execution of the function COMPC in step 9.

9. The user-defined function COMPC is executed. Prior to reading the input data the function checks that it is being run under the 'Process' CHARGINS and that the required exogenous data were derived by the Process READDATA. Since the user has not previously specified the exogenous Cases to be used the latest RUN record in the C-R set for READDATA is accessed and the user is asked if the data for the Run with NAME←JUNEDATA should be used. In this case the user responds in the affirmative. If the user had said 'no' other RUN records in the current C-R set would have been accessed in turn until the desired version of the data was obtained. The function then asks for its Own Input (OPINO) which is input from the terminal. Having located and read the input data COMPC then asks the user if a BASECASE record is to be stored and the user again answers 'yes'. Since this is a new Basecase a CASE record is automatically generated and the P-B and B-C sets are connected for Process CHARGINS.

The APL instruction in Component 4 of the Data File in Figure 8 is generated and stored automatically by the system. When executed this instruction will retrieve the Base Case data from Component 5. At this stage the system records the source of the exogenous data used by this run of the CHARGINS process. This is done by connecting (in the C-I-R set) the RUN record of the READDATA Process to the CASE record just established for the CHARGINS Process. After performing the computations the function displays the results and stores the corresponding RUN record and output data as shown.

10. The function groups and global variables for the CHARGINS Process are erased from the workspace.

11. The CLOSEDATA BASE command closes all Planning System files for the model.

Continuing the example the interactions of the Marketing Group and the Production Department in running the Processes PLIMITS and PRODMIXLP will be similar to the above and are not shown here. Note however that these users can access the BASECASE, CASE, and RUN records of previous users prior to running their Processes to ensure that the previous planning steps have been properly executed and to read any messages left by previous users. (One way of doing this is to issue an AUDIT command which will list all such records stored in the data base subsequent to a given date).

To illustrate the use of the CASE record type in the data base schema suppose that the Marketing Department revises its forecasts after the production plan is run because of a new promotion on product 3. The value of U[3], the third component of the limits vector, must be revised upwards by 1200 units. The interaction is as follows:

```
    )LOAD PLANSYS
    LOADPROCESS 'PLIMITS'
    STORECASE
NAME←PROD3PROMOTION
GIVE EXECUTION INSTRUCTIONS:
P[3] ← P[3] + 1200
GIVE DESCRIPTION:
REVISION TO PLAN BECAUSE OF PROMOTION ON PRODUCT 3

(CASE PROD3PROMOTION STORED 06/29/79 4:05 P.M.)
    CLOSEDATABASE
```

The PRODMIXLP must then be rerun. Note that the PROD3PROMOTION Case will be automatically selected. The Planning System will first retrieve the Base Case and load the data into the workspace. The APL assignment command shown above will then be executed.

REFERENCES

1. S. Alter, 'A Taxonomy of Decision Support Systems,' Sloan Management Review, Vol. 19, No. 1, Fall, 1977.

2. R.W. Blanning, 'The Functions of a Decision Support System,' to appear in Information and Management.

3. R.H. Bonczek, C.W. Holsapple and A.B. Whinston, 'A Theoretical Design for a Decision Support System,' Proceedings IEEE COMPSAC Conference, Chicago, 1977.

4. R.H. Bonczek and A.B. Whinston, 'A Generalized Mapping Language for Network Data Structures,' International Journal of Information Systems, Vol. 2, 1977.

5. Codasyl Committee, 'Data Base Task Group Report,' Association for Computing Machinery, April, 1971.

6. E.F. Codd, A Relational Model for Large Shared Data Banks, Communications of the ACM, Vol. 13, No. 6, June, 1970.

7. 'Educational Data Base System,' Department of Computer Services, The University of Calgary, User's Guide, September, 1977.

8. Information Management System/360, Version 2, Application Programming Reference Model SH20-0912, IBM, White Plains, NY, 1974.

9. P.G.W. Keen and M.S. Scott Morton, 'Decision Support Systems: An Organizational Perspective,' Addison Wesley, Boston, 1978.

10. James Martin, 'Computer Data Base Organization,' Prentice-Hall, Englewood Cliffs, N.J., 2nd edition, 1977.

11. T.H. Naylor and H. Schauland, 'A Survey of Users of Corporate Planning Models,' Management Science, Vol. 22, No. 9, May 1976.

12. E.A. Stohr, 'A Mathematical Programming Generator System in APL,' Working Paper No. 348, Center for Math Studies in Economics and Management Science, Northwestern University, Evanston, IL, March 1979.

13. M.R. Tanniru, 'A Decision Support System for Planning,' Ph.D. Dissertation, Northwestern University, October 1978.