20r

DESIGN AND IMPLEMENTATION OF PROBLEM ENVIRONMENTS AND SOFTWARE SUPPORT TOOLS FOR A MANAGEMENT INFORMATION SYSTEMS COURSE

by

ROBERT A. BIRCHARD

B.A., Emporia State University, 1977

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree of

MASTER of SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

approved by:

Major Professor

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I wish to express my gratitude to Tom Graham and Kenneth Conroe of the KSU Computing Center. Tom Graham for his help with IDMS release 5.7 and the RUNPROG EXEC. Kenneth Conroe for his help with EMPIRE and CMS.

I also am grateful to my wife, Loretta, for her encouragement, her help in proof reading, and her loving care for me and our young son while I was completing this report.

Chapter I

INTRODUCTION

Due to relatively recent advances in automated data processing and computing our society is now in a time called the information age. The information age is a time of dynmanic change. It is a time in wich new ideas and new technologies are a driving force.

This driving force has brought about many changes in our society in the last decade alone. A much wider segment of our society can now use a microcomputer as a tool in their daily work. This includes management at all levels in business, industry, and government.

Computers generate a lot of information. In recent years experts in the use of information systems (IS) have been confronted with this ever increasing onslaught of information. The IS must be selective in dealing with this ocean of information. As we develop new and more sophisticated ways of generating information IS must become more sophisticated in selection of useful information. This has resulted in the increased use and development of new information systems.

1.1 THE PROBLEM

One area of growth in information systems today is that of Management Information Systems (MIS). The use of MIS is on the increase in all areas of our economy. This includes all levels of management in business, industry, and government. The application of MIS will continue to grow and be a part of management practices for some time.

In 1980 the Kansas State University Computer Science Department first offered a degree in IS. The IS curriculum has provided students with a satisfactory foundation in business computing practices through several undergraduate courses. The CS dept. has recently decided to offer an undergraduate level course in MIS.

In 1982 Neil Strunk completed his masters project with a report titled, SUPPORT TOOLS FOR AN UNDERGRADUATE LEVEL MIS COURSE, (STRUNK:82). The purpose of his project was to develop a plan to give business students an adequate preparation in MIS. The Computer Science department IS curriculum can also benefit from the results obtained in Neil's report.

The report expanded on several areas important to the introduction of a new MIS course. The Student Behavioral Learning Objectives (SBLO) and proposed course topics were outlined, potential textbooks were reviewed, and a discussion of ideal hardware and software was presented. Available hardware and software at the University and a survey of software available in the market place are described. Fi-

nally, a framework of problem environments is built upon a foundation of the behavioral objectives.

Strunk's work left several unsolved problems. One problem that remains can be subdivided into three parts. First, which specific problem environments are to be used. Secondly, what are the software requirements for implementing these problem environments. Finally, the implementation of the software support tools. In preparing to solve the problem an examination must be made of what has already been done in regard to the proposed SBLO and the proposed problem environments.

1.2 STUDENT BEHAVIORAL OBJECTIVES DESCRIBED

The SBLO are activities in which students are expected to demonstrate their performance as they progress through a course. There are several bases for the SBLO. First, they are a statement of what a student is to learn in the course; and secondly, they are a standard by which the progress of the student can be measured.

The SBLO for a MIS course have been derived from characteristics of the course audience and course objectives for that audience. The audience concerned is the IS majors. In specifying SBLO for this audience it is necessary to make some assumptions. First of all, it is assumed that these IS majors will have had the experience resulting from completion of at least the freshman and sophomore level of their

curriculum. Another assumption is that they will have had data base experience. With this level of experience it is assumed that these IS students will benefit most from a functional, hands-on approach.

There are several objectives for this course that need to be considered. First, the students are to learn what a MIS consists of, secondly, they are to learn about design of a MIS, thirdly, they are to grow in their knowledge of how business functions, and finally, they are to learn how computer systems are merged to form a MIS.

1.3 SBLO DETAILED

In (STRUNK:82), four general areas in which MIS students should demonstrate spec behavioral responses were identified in the report by Neil Strunk(STRUNK:82). These general areas are the following:

1. Decision Support Systems (DSS)
2. Data Base Management Systems (DBMS)
3. Office Automation (OA)
4. Transaction Processing Systems (TPS)

Later in this report an examination will show specifically how each of these are will be covered in the MIS course for IS majors. Now the work that has already been done in terms of detailing the SBLO for each area will be considered.

Decision Support Systems (DSS) are interactive computer systems. They are designed as an aid to managers in solving

problems that involve judgement (i.e., unstructured and semi-structured problems). It is important to emphasize that it is the manager who has the authority and the base of knowledge necessary to make a decision. The purpose of the DSS is to supply the manager with information which will aid in making a decision.

With the growing mountain of information today a DSS can be a valuable aid to a manager in putting information to work and in selection of useful information. The following SBLO in the area of DSS will provide the IS major with a good understanding of the role of a DSS within a MIS:

- use an available application program to answer a structured problem,
- use one or more available DSS tools and combine the incomplete pieces to make a decision about a predefined problem,
- design a semi-structured problem environment within a predefined scope, (STRUNK:82).

Objectives defined in the area of Data Base Management Systems will make use of one data base and one data base management system. The data base will already be available and the SBLO are:

- use a query language to retrieve a record
- use a query language to retrieve information from two types of records
- add a record to the data base

- delete a record from a data base
- change a field value in a record in the data base.

In the area of office automation objectives defined for IS majors are as follows:

- send and receive mail
- retrieve a file from an automatic filing system
- put a week's schedule on a calendar system
- schedule a meeting with a specified number of people using a calendar system
- set up and participate in an electronic conference
- make a remote presentation

Transaction processors are in wide use in many organizations. Often their use can be found at a customer business interface or at the operational level. Because of their wide usage it is considered important that IS students acquire the ability to use a transaction processor. Now the problem environments to satisfy these SBLO will be considered.

1.4 PROBLEM ENVIRONMENTS DESCRIBED

See chapters 3, 4, and 5 for details of specific software requirements for the support tools of the MIS course. First though the problem environments (scenarios) as expanded in chapter IV of Neil Strunk's report (STRUNK:82), will be described.

MIS problem environments will be subdivided into the following four areas :

- Decision Support Systems
- Data Base Management Systems
- Office Automation
- Transaction Processing Systems

Problem scenarios that incorporate the SBLO described in section 1.3 will be described in general for each of these four areas.

1.4.1 DECISION SUPPORT SYSTEMS

Problem 1: A semi-structured problem. The student is required to use one or more DSS available applications to generate output. A decision is required by the student based on the output.

a) The student is presented with a semi-structured problem,
b) the necessary decision support tools are described with respect to their possible support in the necessary decision,
c) the student is required to choose the appropriate tools to use,
d) the applications are run by the student
e) the student makes a decision based on output from the tools,

f) evaluation will be based on the choice of tools and the resulting decision.

Problem 2: the student will design a semi-structured problem.

a) The student is supplied with decision support tools,
b) documentation on use of the tools is available to the students,
c) the student is to design a semi-structured problem whose solution requires the aid of several DSS tools,
d) the student must demonstrate solution of the problem,
e) evaluation will be based on the way in which DSS tools are incorporated into a specialized 'system' to solve the problem.

Problem 3: This is an unstructured problem.

a) The student is given a simple unstructured problem. The solution will require some retrieval of information to aid in the decision,

b) information needed to make the decision is retrieved from the data base,

c) the student must report his decision as an indication of completion.

1.4.2 DATA BASE MANAGEMENT SYSTEMS

The students will be provided with a simple, predefined data base for this series of problems. The objective is to expose the students to a low level of data base manipulation. It is desirable to limit the problem environments to one data base management system, using the most English-like query language available.

Problem 1: This is a simple retrieval. The purpose is to introduce students to the DBMS.

a) Students are given a simple subschema description of the data base to be used,

b) a record to be retrieved is specified,

c) the record is retrieved by a student through interactive query,

d) proof of completion is the query result.

Problem 2: This scenario builds on the previous one.

a) Students are instructed to retrieve two different record types from the data base,
b) proof of completion is the query result.

Problem 3: This problem uses queries that modify the data base. Specifically add, delete, and change a record.

a) The student is given a specific record name,
b) the student must locate the record through interactive queries,
c) the record is deleted,
d) the name of another record and a change to be made in it are given to the student,
e) the record must be located and changed,
f) the student is instructed to insert a record provided,
g) the student must locate the place to insert the record and add it to the data base,
h) completion of the delete, change, and add will terminate this exercise.

Problem 4: This scenario is to enable more experienced students to manipulate the data base subschema structure.

a) The student is given a subschema to the data base,

b) documentation of the data manipulation language (DML) appropriate to the DBMS is provided,

c) students are provided with specification for a specific structure change,

d) the student must use the DML to make the change,

e) successful changed subschema use will end the exercise.

1.4.3 OFFICE AUTOMATION

Problem 1: This exercise is intended to give the student experience on an intra/inter office mail system.

a) i) The student is given a text message,

ii) the student is given the name of a person to whom the message is to be sent,

b) a file is created containing the message, point 2 the message is sent to the specified person,

c)

receiving of the message will constitute successful completion of this task,

d) the student must read-in a message sent to them,

e) a printed copy of this message will constitute successful completion.

Problem 2: This problem is intended to provide the student with experience using an automatic file system.

a) The student is given the name of a file to retrieve,
b) the student must use an automatic file system to retrieve the specific file,
c) a printed copy of the retrieved file will complete this exercise.

Problem 3: This problem is intended to give the student experience using an electronic calendar system.

a) The student is instructed to enter a schedule of his week into the calendar system,
b) the student is given the name of a fellow person on the system, point 2 the student is instructed to schedule a meeting with this person based on that person's calendar schedule,
c) successful completion of the scheduled meeting will end the exercise.

Problem 4: The ability to setup and participate in a remote electronic conference is the experience provided by this problem.

a) The student is given a group of names of other students and a time to set up a conference with them,

b) at the specified time, the people should be called to begin the conference,

c) successful contact with the designated people and a short conversation will indicate completion.

1.4.4 TRANSACTION PROCESSING SYSTEMS

It is important to enforce understanding of transaction processing systems (TPS) by using one. They handle the majority of the work load of an information system (STRUNK 82). Problem 1: This problem will allow the student to demonstrate the ability to use a TPS.

1.

a) The student is given documentation for a specific TPS,

b) a listing of input data is given to the student,

c) an operational report is specified,

d) the student enters the data into the TPS,

e) the printed report will indicate successful completion of this exercise.

These scenarios are general in nature in order to allow flexibility in choice of software support. In chapter 2 specific problem environments will be described which will

satisfy the SBLO chosen and will be implemented on the software support systems specified.

The problem scenarios to be described do not necessarily shed any light on the limitations of MIS systems currently available. In order to gain a full understanding of a MIS it is important for IS students to experience its limitations. A later chapter will include problem environments which make clear the limitations of a particular MIS system.

1.5 THE SOLUTION

Sections 1.3 and 1.4 have shown some of the background necessary to develop a solution to the problem of specifying and implementing specific problem environments for a MIS course for IS majors. This background material first defined a SBLO, then a description of previous work which outlined the SBLO followed, and finally a description of general problem environments was presented.

The solution is divided into three parts. First of all the specific hardware and software environment must be described. Secondly, specific SBLO and problem environments must be detailed. Finally, actual implementation of the problem environments must be carried out and documented.

Chapter II will describe the software-hardware environment, SBLO chosen, and specific problem environments. Chapter III will describe the design and implementation of the data base to be used for the DBMS problem environments.

Chapter IV will describe the implementation of the user interface to be used in the DBMS problem environments. Chapter V will describe implementation of the DSS problem environments. Chapter VI will be conclusions and future work. An appendix will contain documents pertinent to implementation of each of the three areas.

Chapter II

THE PROBLEM ENVIRONMENTS

2.1 INTRODUCTION

Translation of the SBLO into actual implementation of problem environments requires careful evaluation of many alternatives. Factors that require evaluation can be grouped into several areas. These include constraints imposed by the approach, hardware and software available, and the SBLO.

The course is geared to IS majors. The problem environments must be implemented in an interactive environment in order to give the student contemporary experience. In addition, for the student to benefit most, it is necessary for him to have a hands-on experience. These two factors alone impose some constraints on hardware and software.

2.2 HARDWARE/SOFTWARE SUPPORT TOOLS

It is not considered economically feasible to buy or lease new hardware or software for the MIS course at this time. Therefore, the decision has been made to use hardware and software already available. There is a possibility of writ-

ing application programs or utility programs to implement the problem environments.

There are currently two machines considered as candidates for use in this course. The two machine environments are:

1. National 6/30, 4 megabytes of storage, running OS/MVT. Release 21.8A and HASP Remote Job Entry under VM/SP release 1. (University Computing Center)
2. Interdata 8/32, 1 megabyte of storage, running under UNIX. (Computer Science Department)

The National system is the main computer system used for academic and research applications. It can easily support the additional load imposed by the 30 studenst per semester expected. The hardware itself has little effect on resolving the issue of which hardware and software to use. The Interdata 8/32 supports many software tools under the UNIX operating system. Among these are tools that would be useful for implementing problem environments in the area of office automation.

A very convenient intra/inter office mail system and a calendar appointment system are available. In the mail system the user is notified at login if he has mail from other users. Mail is sent to other users by means of a simple command and is addressed to a login name.

Currently, the 6/30 has software tools available that would be useful for implementing problem environments in the areas of DBMS and DSS. For the DBMS section of the course,

available is the Integrated Data Management System (IDMS) through the Cullinane Corporation. The DSS problem environments can easily be implemented interactively using a software tool called EMPIRE produced by Applied Data Research.

The IDMS data base management system is a state-of-the-art data base management system for network modeled data bases. The IDMS DML commands are imbedded in either COBOL or PL/I. Until recently, IDMS has been run most frequently in the batch mode at KSU. Now, release 5.7 has been implemented here. It is hoped that this will enable more convenient use of IDMS interactively through CMS. Whatever the case, it will still be necessary to write a user 'front end' to enhance the effectiveness of IDMS in the CMS environment. More discussion of this will follow in a later chapter.

EMPIRE is the decision support system currently available. It is a specialized application tool designed to allow the non-technical user to describe business situations, assign values to known quantities, produce reports, and evaluate various alternative assumptions. It contains a modeling language that allows situations to be easily described. Reports are controlled by an extensive set of formatting options. Access to all of EMPIRE's features is accomplished by simple English-like commands in the CMS environment.

2.3 SBLO FOR THE MIS COURSE

The SBLO have a direct influence on the specific problem environments. Therefore, a closer look will be taken at the actual SBLO chosen for the MIS course. Again considered will be the four general areas in which IS students should display behavioral responses in an MIS course.

For the most part the SBLO as described were adopted. Therefore, to avoid unnecessary duplication of material only the changes to the SBLO will be discussed. A complete listing of the SBLO for this MIS course can be found in appendix A.

IDMS does not have available an interactive query language for users. The student can still get the feel of using a query language with limited capabilities. The DBMS SBLO will be changed to read use of a query language subset. In effect the origional SBLO are unchanged.

In the area of office automation, constraints will eliminate several SBLO previously described. The Interdata 8/32 is the machine which will be used for the OA problem enviro.It has no software tools that enable electronic conferencing. Therefore, the following two SBLO must be eliminated:

- set up and participate in an electronic conference
- make a remote presentation.

All other OA SBLO previously described will be adopted.

Currently there is no TPS available. Therefore the TPS SBLO will not be included in this course.

EMPIRE will enable satisfactory implementation of all SBLO described for the area of DSS. Therefore, all the DSS SBLO will be adopted for use in this course.

Several changes in the SBLO have been briefly described. In summary, some of the OA objectives must be dropped and all the TPS SBLO will be eliminated from the MIS course for the present. With this background, the specific problem environments to be used will be described.

2.4 SPECIFIC PROBLEM ENVIRONMENTS

The use of interrelated problem environments is desirable because this will give the student a better understanding of the way in which a MIS is used in a real business setting. For this reason, the problem environments within each area will be interrelated in such a way that they build one on another. Also, the problem environments will be interrelated as much as possible so that a problem environment in one area makes use of the result of the completion of a problem environment in another area. Next, will be described the way in which problem environments of different areas will be tied together.

In order to tie together the problem environments it seemed best to create a ficticious setting which could pro-

vide a rational for completion of the series of tasks. Satisfactory completion of the series of tasks would require use of software tools from each of the three areas covered by this MIS course. A job environment and a role for the student in this environment will be created.

The job environment will be described in outline form. This will include a brief profile of a fictious company and a functional outline of the role of the manager who is the direct supervisor of the student. The job environment is as follows:

> COMPANY NAME: Justin Allen Petroleum Corporation
> DESCRIPTION: Justin Allen has been in business for over 50 years. It is a major corporation employing more then 20,000 people. It is a large producer of oil, natural gas, and natural gas liquids. In addition, it refines, transports, and markets fuels and lubricants. It also is an important manufacturer and marketer of chemical products.
>
> Justin Allen has five major divisions. The division with which we are concerned is the Chemicals Division. This division is responsible for marketing chemicals and related products.
>
> SUPERVISOR:Ms. L. J. Prose
> DESCRIPTION:Ms. L. J. Prose is a member of the commercial development area of the Chemicals group. It is her responsibility to carry out economic evaluation of new products and marketing research.
>
> STUDENT ROLE: Entry level information systems specialists.
> DESCRIPTION: Assume that you hold an entry level position in the Justin Allen Petroleum Corporation. You are an imformation systems specialists directly responsible to Ms. L. J. Prose. Ms. Prose will oversee your work and direct your activities and you will report your results to her.

We now have sufficent background to describe the individual problem environments. Each problem environment description will include the problem scenario, data given to the student, and necessary supporting documents and files.

2.4.1 OFFICE AUTOMATION PROBLEM ENVIRONMENTS

Problem 1: The purpose of this exercise is to give you experience on an intra/inter office mail system. You will be provided with documentation for the Interdata 8/32 mail system. When you first login on the Interdata 8/32 there will be mail for you. In order to complete the assignment you must read the mail and complete the task it describes for this problem.

1. Data:
   a) No data is necessary for completion of this problem.
2. Support:
   a) Documentation for the Interdata 8/32 mail system.
   b) the following mail:

> From: MS L. J. Prose To: I. S. Majors Subject: new product (newlub id# 15195500).
>
> I have recently been informed Justin Allen will soon be marketing a superior quality lubricant recently developed in our labs. It is necessary for completion of the economic analysis that I confirm the tax rate for this type of product. You will be provided with documentation necessary to retrieve this information. **Note: In order to sat-

isfy problem 1 you need only acknowledge that you have read this message. To do this, mail the following message to _____: Message #1 received by (your name) on (date).

Problem 2: The purpose of this problem is to give you experience using an electronic calendar system. You will be provided with documentation for the Interdata 8/32 electronic calendar system. You must enter your own schedule for one week into the system. You will be given the name of another student with whom you are to schedule a meeting. In the course of the meeting you must get the employee number of the person with whom you met.

**Note: To complete problem 2 you must confirm your meeting. To do this mail the following message to _____: (your name) met with (other students name) employee id# _____ on (date) at (time).

1. Support:
2. Documentation on the electronic calendar system available on the Interdata 8/3
3. the name of another student to schedule a meeting with,
4. each student must be given a unique id number. This number will be called their employee id #.

2.4.2 DBMS PROBLEM ENVIRONMENTS

Problem 1: The purpose of this exercise is to introduce you to IDMS. You will be given a simple subschema description of the data base. You are to retrieve the MKT record in the PROD_MKT set. The MKTNAME for the market you need is MIDWEST.

To bring up IDMS at your terminal type 'RUNPROG DBQUERY' As soon as IDMS is up a menu of tasks will be displayed on the screen. Use the DBGET command with MIDWEST for RID, MARKET for RN, and NEWLUB for PID. This will automatically retrieve the specified record and store a copy of it in a file named 'DBOUT LISTING'. The record will also be displayed. To print a copy you must use 'OSPRINT DBOUT LISTING (FORM 2601'. This output must be turned in to receive credit for this exercise.

1. Data:
   a) No data is necessary.
2. Support:
   a) Subschema description,
   b) documentation on 'DBQUERY' command set,
   c) name of record to retrieve,
   d) simple data base,
   e) 'front end' written in PL/I.

Problem 2: This problem builds on the previous DBMS problem. You must bring up IDMS using the command 'RUNPROG DBQUERY'. A menu of tasks will be displayed when the system is ready. Use the DBGET command to retrieve two records. First retrieve the 1984 MKTEST record for NEWLUB in the MIDWEST. From this record you need information on market share (MKTSHR), unit cost (UCOST), and unit price (PRICE). The second record you must retrieve is the 78thru83 MKTHIST record for NEWLUB in the MIDWEST. From this record you need to get information on total units sold in the market place by quarter for the previous 5 years (YEAR1 thru YEAR5). Be sure to save a copy of this information because you will need it on DSS problem 2.

1. Data:
   a) No data is necessary.
2. Support:
   a) same as for DBMS problem 1.

Problem 3: The purpose of this series of exercises is to familiarize you with queries that modify the data base, specifically add, delete, and change a record occurance. After your recent evaluation of NEWLUB using EMPIRE you will have arrived at a decision about the best unit price for this product. Therefore, it is necessary to change some information in our data base. First you need to delete the MKT re-

cord in the PROD_MKT set with MKTNAME NORTHWEST. Then you must change the UCOST, and PRICE fields in the MKTEST record of the EST set. A new market has been added for our product so you must add an MKT record with MKTNAME SOUTHWEST. When you have finished these modifications print a copy of the contents of the data base using the DBREPORT command. Hand in your output to receive credit for this set of exercises.

1. Data:
   a) The student must use data derived from DSS problem 2 for changes made to UCOST and PRICE.
2. Support:
   a) same as for DBMS problems 1 and 2.

2.4.3 DSS PROBLEM ENVIRONMENTS

Problem 1: The purpose of this problem is to introduce you to EMPIRE and give you experience with a computer solved structured problem. No data input is required from you to complete this problem. You will be given documentation describing the decision support system EMPIRE. Invoke EMPIRE, execute finansum, PRINT from fsumrep. The result will be generation of a ten year financial and operating summary for the Justin Allen Petroleum Corporation. To save a copy of

this report you must use the command PRINT FROM FSUMREP TO FINREPORT. This will output a copy of the report to a file 'FINREPORT EMPOUT' which you will OSPRINT.

1. Data :

   a) No data required.

2. Support :

   a) EMPIRE model file (FINANSUM EMPMOD),

   b) EMPIRE report file (FSUMREP EMPREP),

   c) EMPIRE data file (FSUMDATA EMPDATA),

   d) EMPIRE control file (FSUMCON EMPCON),

   e) Documtation describing report generation using EMPIRE.

Problem 2: The purpose of this problem is to give you experience with a computer solved structured problem. You will be provided with documentation describing the decision support system EMPIRE. Your job is to analyze a quarterly forecast of potential earnings during the coming year for NEWLUB. The analysis will be further developed in following DSS problems. For this first problem you will need the data on tax rate, unit cost, unit price, and total units sold in the market place. In addition, you will need the data on estimated market share by quarter.

You will be given documentation on how to input your data. Also, you will be given documentation describing the prompts that will be generated by EMPIRE and the response you should make.

1. Data:
   a) The student must use data derived from completion of DBMS problems 1 and 2. This will be values for UCOST, PRICE, TUNITS, and MKTSHR.
2. Support:
   a) Document describing data input,
   b) document describing the EMPIRE session,
   c) EMPIRE model file (MYSAMPLE EMPMOD),
   d) EMPIRE report file (MYREP EMPREP),
   e) EMPIRE data file for TUNITS (HIST2 EMPDATA)

Problem 3: The purpose of this problem is to give you experience with a semi-structured problem. It will be necessary for you to use one or more DSS applications to solve this problem.

After completing DSS problem 2 and have examined your results, you would notice that the value for total earnings per share for the year was $2.031. This value falls short of the value set by corporate management for this product. The desired value is $2.50. You must examine some alternate estimates and strategies to determine whether or nor this goal can be met. In order to make this decision you must do one or more of the following:

- Impact analysis-- this can be used to provide a list of items that if changed will have an impact on the earnings per share.

- Target value analysis-- this enables one to determine if it is possible to improve earnings per share by altering only one other variable.
- Monte Carlo simulation--this enables one to consider a range of possible values for a set of items to determine the likelihood of achieving $2.50 per share.
- Inquiries--this enables one to display current values of specified variables.
- Temporary changes--this enables one to give a variable a temporary value to enable further evaluation. The original value is not deleted and the temporary value can be canceled at any time.

1. Data:
   a) No data input by the student is necessary.
2. Support:
   a) Documentation describing the EMPIRE commands needed to invoke the features described.
   b) EMPIRE model file.
   c) other files as for DSS problem 2.

2.5 SEQUENCING OF PROBLEM ENVIRONMENTS

The problem environments as they are set up have some restrictions on their ordering. Figure 1 shown below details these restrictions. A problem that appears below another one in a column follows after the one above it in time. In general, a problem that appears to the right of another problem on a row follows after it in time . Exceptions to this are that OA problem 2 may come at any time following the completion of OA problem 1. In addition, DBMS problem 4 may follow any time after completion of DBMS problem 3. Also DSS problem 1 may be assigned any time prior to DSS problem 2. Finally, DSS problem 4 may follow any time after completion of DSS problem 3. An arrow indicates that data from the problem at the base of the arrow is necessary for completion of the problem at the head of the arrow.

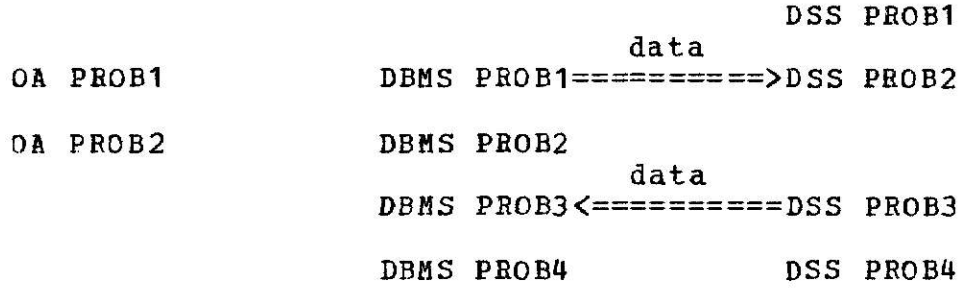

Figure 1: Sequencing of the problem environments

Chapter III

DBMS PROBLEM ENVIRONMENT IMPLEMENTATION: THE SCHEMA

In previous chapters, described were the SBLO, specific problem environments, and general hardware and software support requirements. In the following chapters the implementation of the specific problem environments for each of the three areas will be documented. The method chosen focuses on the software itself and documents the software development. The first set of problem environments dealt with are the DBMS problem environments.

There are three software units that need to be developed to support the DBMS problem environments. They are the schema, subschema, and user interface. The format of the user interface will depend on the schema and subschema, therefore we will begin with a description of schema and subschema development.

3.2 SCHEMA AND SUBSCHEMA REQUIREMENTS

The SBLO and the specific problem environments described for the area of DBMS imply some requirements for the software that will support them. Some of these requirements

concern general characteristics of the data base while others concern specific characteristics of structure and content.

The data base described by the DBMS SBLO is a 'simple' data base. A requirement implied by this is that the data base will have no more record types or set types than are necessary for satisifaction of the specific problem environments and SBLO. Another is that record types and set types must be conceptually clear and easy to understand, and distinct from one another.

Of course 'simple' does not mean simple-minded or overly contrived. Therefore, the data base must be represented by enough record types and set types to make it creditable as a data base. For this reason, record and set types, which will not necessarily be manipulated by a particular student during the course of the semester, will be included. Each of these record and set types would be useful in the support of specific problem environments developed at some later date in response to the evolution of the MIS course.

The second DSS problem environment makes use of imformation obtained through completion of the first two DBMS problem environments. In addition, the third DBMS problem environment makes use of information obtained through completion of the DSS problem environments 2 & 3. Therefore, the data base must contain information that will support and is ef-

fected by the outcome of the DSS problem environment completion.

Since the information in the data base must support the DSS problem environments what must be asked is : 'what type of information is needed to complete the DSS problem environments?'. The information that is needed is about a new product developed by the Chemical Division of Justin Allen Petroleum Corporation.

It seems best to make the purpose of the data base to be storing information about all products developed by the Chemical Division of the Justin Allen Corporation. One group of product records then would be specifically for newly developed products.

The SBLO impose some requirements on the structure of the data base. These requirements relate to the number of distinct record types and the way in which they will be manipulated by the students. To enable satisifaction of SBLO for the DBMS area the data base must contain at least two distinct record types. The subschema must contain a record occurance to be deleted, a record occurance to be changed, two record occurances to be retrieved, and must allow for the insertion of a new record occurance and a new record type.

Completion of DSS problem environments 2 & 3 will require the students to input four data entities. The student is to

input information concerning total units sold in the market place by quarter for a five year period, the estimated percentage share of the market by quarter for the coming year, an estimate of the cost of producing each unit, and a planned selling price for the unit. Information on each of these four entities must therefore be included in the data base.

There are some other data entities which are needed to implement the DSS problem environment software support tools that are not required in the data base for completion of any of the problems. If they are included in the data base, however, they would lend to its credibility as a data base. Therefore, as shown in figure 2 included will be these entities besides those that are needed by the students to complete the DSS problem environments.

- units sold
- sales revenue
- cost of sales
- gross profit
- overhead cost
- overhead expense rate
- profit before tax
- income tax rate
- profit after tax
- earnings per share
- number of shares

Figure 2: Additional entities to be included in the data base

3.1 SCHEMA AND SUBSCHEMA DESIGN

The design process for the schema and subschema is governed by the requirements and characteristics of IDMS. IDMS builds the data base as a set of record types and the relationships among record types. First to be examined are the record types included in the schema.

The purpose of the data base is to store information about products. Therefore, the first record type is named PRODUCT. Each occurance of a PRODUCT record will identify a product developed by the Chemicals Division of Justin Allen Petroleum Corporation.

Much of the data concerns marketing information. This implies the need for another record type that will be named MKT. Each occurance of the MKT record will identify a specific region which is a market, or a potential market for a product.

Some of the marketing information is historical and some is estimates on future performance in a market place. This suggests two more record types one named MKTHIST and another named MKTEST.

Each occurance of the MKTHIST record identifies a historical 5 year period for which marketing information about simliar products has been kept. Each occurance of the

MKTEST contains information on marketing estimates for a specific year for the product. Information in these two record types is concerned with one market place.

One of the DBMS SBLO requires the students to make a change in the subschema. In order to allow greater flexability in this regard, one more record type is included. This record type must fit easily into the purpose of the data base. This record type will be named DEV_HIST. Each occurance of the DEV_HIST record identifies documentation of the developmental history of the product. This could also include patent information if appropiate.

Record relationships are described as set types; it is necessary to define a record type as the owner record and a record type as the member record in each set type. Since the IDMS schema DDL enables one to define many types of relationships between record types consider the question, 'what are the relationships among the record types?'. These relationships can be resolved by studying the identity and role of each record type.

The PRODUCT record identifies a unique product. The MKT record identifies a unique market region for that product. Therefore, a set type is defined named PROD_MKT in which PRODUCT is the owner and MKT is the member record type. There are one or more occurances of MKT for each occurance of PRODUCT.

The MKTHIST record contains historical infomation about a specific market region. A set type named HIST will identify this relationship. The MKTEST record contains estimates for a specific market region. Therefore, a set type is defined named EST. For each of the MKT record there are one or more occurances of the MKTHIST record and the MKTEST record.

The DEVHIST record identifies information on the development history of a specific product. This suggests a set type named PROD_DEV in which PRODUCT is the owner record and DEV_HIST is the member record. There is only one occurance of the DEV_HIST record for each occurance of the PRODUCT record. Table 1 indicates the name of each record type and the data elements associated with it. Table 2 shows the relationships among records.

TABLE 1

Record Types and Their Attributes

RECORD TYPE	DATA ELEMENTS
PRODUCT	product id no
	product name
	product status
MKT	market id no
	market name
	overhead cost
	overhead expense rate
	tax rate
	number of shares
	unit cost
	unit price
MKTHIST	market id no
	product id no
	date of first quarter
	date of last quarter
	total units sold in the market place
MKTEST	date
	market id no
	product id no
	%market share by quarter
	units sold
	sales revenue
	cost of sales
	gross profit
	profit before tax
	profit after tax
	earnings per share
DEVHIST	product id no
	patent documents no
	development documents no

TABLE 2

Record Relationships

SET TYPE	OWNER	MEMBER
PROD_MKT	PRODUCT	MKT
HIST	MKT	MKTHIST
EST	MKT	MKTEST

3.2 STUDENT INTERACTION WITH THE DATA BASE

The information that students will need to complete DSS problem 2 is found in three different record types. The unit cost, unit price, and the tax rate are in the MKT record. The % market share is in the MKTEST record. The total units sold in the market place is in the MKTHIST record and these three records will be retrieved by the student to get the needed information. Since the SBLO only requires the student to retrieve two different record types this will satisfy the SBLO.

The result of DSS problem 3 will lead to a decision that will change the unit cost and unit price. Both these data items are in the MKT record. Therefore, to satisfy the SBLO for changing information in a record occurance, the students will enter a new value for unit cost and unit price based on their decision for DSS problem 3.

The MKT record identifies a market for a product. In the early stages of marketing a new product, new market regions may be added while other market regions may be dropped. This rational can be used to justify using an MKT record occurance as the one to delete and another MKT record occurance as the one to add.

The final SBLO for the DBMS area concerns a change in the subschema. The subschema as written for the MIS course will not include the DEV_HIST record. This is because the information in this record has no bearing on the solution of the DSS problems. By adding the DEV_HIST record to the subschema the students could satisfy this SBLO.

This completes the description of the schema and subschema development for the data base. Details on data elements and the actual schema and subschema DDL are found in appendix A. The data base as designed can be used by the student to satisfy all SBLO for the DBMS area. In addition, the information stored in the data base ties together the DBMS and DSS problem environments. Next described will be development of the user interface software.

Chapter IV

DBMS PROBLEM ENVIRONMENTS USER INTERFACE DEVELOPMENT

The IDMS was designed to be used in the batch mode. It does not provide a query language for the user. It will be necessary to write an application program that will support the DBMS problem environments. This chapter will describe the development of the user interface.

We will consider the requirements for the application program, discuss the design process, and consider how the design satisifies the SBLO and supports the problem environments for the DBMS area.

4.1 USER INTERFACE REQUIREMENTS

The course objectives include giving the students a hands-on experience in an interactive environment. Therefore, the application program must be interactive in nature. This implies the ability to receive input from the terminal and send output to the CRT. It also implies that the program

will prompt the user for input and must remain on-line until the student terminates the session.

Another part of the approach to the course objectives is the desire to give the students a good impression of a DBMS. This suggests a system which is user friendly and implies that the system will support English-like commands. It implies that it will provide explicit prompts, good documentation, and useful diagnostic aids. The IDMS system will provide the user with sufficent debugging aids.

The application program will act as an interface between the student and the data base. It must therefore include whatever code is necessary to enable the IDMS to use it as a run unit.

The SBLO mention several different manipulations of the data base. In order to enable the student to satisfy the SBLO the user interface must support the four data base manipulation commands as shown in figure 3

-retrieval of record occurances of more then one record type
-deletion of a record occurance
-insertion of a new record occurance
-change of one or more fields of a record occurance

Figure 3: Required data base manipulation commands

The SBLO also dictate that the student must complete a change in the subschema. This means that the application program which is the user interface must be adaptable to some changes in the subschema.

In order to provide the best service to the student the interface must be able to manipulate any record occurance in the data base. The data base structure has five record types and four set types (see figure 3.2). The interface must therefore be able to recognize and manipulate each of these. Also, in order to provide for the DBCHANGE command the interface must be able to recognize and manipulate all data elements in the data base.

Some other features would be useful to enhance the user friendliness of the interface. These include a command called DBMENU and a command called DBREPORT. The purpose of the DBMENU command is to enable the student to view the menu of tasks at any time that a prompt for a command is displayed. The purpose of the DBREPORT command is to print the contents of the data base in a convenient format.

To enable the student to end execution a command named LEAVE is provided. When it is executed the IDMS environment is exited and the on-line session is terminated.

In figure 4 requirements for the user interface are given.

-be interactive
-remain on-line until terminated by the user
-enable input and output through the CRT
-prompt the user for input
-support English-like commands
-be supported by helpful documentation of it's features
-be useable as an IDMS run unit
-support the following commands:
    --RETRIEVE
    --DELETE
    --INSERT
    --CHANGE
    --DBREPORT
    --DBMENU
    --LEAVE
-be adaptable to some changes in the subschema
-recognize and be able to manipulate all record types, set types, and data elements included in the schema DDL

Figure 4: User interface requirements

4.2 USER INTERFACE DESIGN

A structured design which divides the program into modules will be used. Each module will for the most part carry out only one function. The requirements specifications identify what functions the program needs to support. Each of these functions will be the basis for a module.

During the development of the requirements eight functions were identified. Eight commands to implement these functions are DBGET,DBDELETE, DBADD, DBCHANGE, DBREPORT, DBMENU, LEAVE.

Other modules may be identified by considering what functions will be carried out by the main program. The main program must build the data base, display the menu, accept a command, and process the command. This suggests two modules not yet mentioned, a module named DBUILD to build the data base and a module named PROC_CMD to process the commands.

4.2.1 MAIN PROGRAM

The high-level algorithm for the main program of the user interface is shown in table 3

TABLE 3

Main Program

```
1.0 Build the data base.
2.0 Display the menu.
3.0 Repeat until COMMAND=LEAVE
      3.1 Get a command
      3.2 Process the command
3.3 End repeat.
4.0 Release all areas.
5.0 End algorithm.
```

4.2.2 DBUILD

The first subroutine called by the main program is DBUILD. The purpose of this module is to initialize the data base with the appropiate data. the algothrim for DBUILD is shown in table 4

TABLE 4

DBUILD

```
1.0 Get a data card.
2.0 Do while not EOF
      2.1 Echo print the data card
      2.2 Call procedure INSERT
      2.3 Get a data card
2.5 End while.
3.0 End algorithm.
```

4.2.3 DBMENU

The next subroutine called by the main program is DBMENU. The purpose of DBMENU is to generate a menu of tasks. The menu will have an introduction, a listing of commands with the correct syntax, and an explanation of how to execute a command.

The purpose of DBMENU is to enable the student to view the menu of data base manipulation tasks whenever a prompt for a command is displayed. Following the successful execution of a command statement the menu will not be displayed unless specifically requested by the user; instead a prompt for a command statement will be displayed.

Following display of the menu the main program will prompt the user for a command. It gets the command, clears the menu from the screen, and then processes the command.

4.2.4 PROC_CMD

This command will be an English-like statement with syntax as shown in figure 5 . The PROC_CMD procedure first calls DBPARSE_CMD which assigns values to CTYP, RID, RN, PID, and MID the user is notified if one of these values is expected but not found in the command-statement. For all commands except DBMENU, DBREPORT, and LEAVE the next step is to call REC_FIND which obtains RN with RID or informs the user that the record was not found or is not a correct record type. Once the record has been obtained the appropriate subroutine is called to carry out the desired data base manipulation.

```
COMMAND_STMT::= <COMMAND_TYPE>|<COMMAND_TYPE>b<ARGSET>

COMMAND_TYPE::= DBGET | DBDELETE | DBADD | DBCHANGE
                DBREPORT | DBMENU | LEAVE

b::=one or more blanks

ARGSET::= <RECORD_ID>b<RECORD_TYPE> |
          <RECORD_ID>b<RECORD_TYPE>for<PRODNAME> |
<RECORD_ID>b<RECORD_TYPE>for<PRODNAME>inbthe<MKTNAME>

RECORD_TYPE::= PRODUCT | MKT | MKTHIST | MKTEST
               | DEVHIST

RECORD_ID::= PRODNAME | MKTNAME | FDATE | PERIOD
```

Figure 5: Syntax for the data base manipulation commands

4.2.5 DBGET

The purpose of DBGET is to output the data contained in a specific record occurence in the data base. Once REC_FIND has obtained the specific record occurance DBGET outputs a copy of it using PUT EDIT statements. Then it informs the user of the status of the command statement execution. If there is a syntax error the student will have opportunity to re-enter part or all of the command statement. If the execution is successful the user will be informed by a message displayed.

4.2.6 DBDELETE

The purpose of DBDELETE is to delete a specific occurance of a record type indicated by the student. It first recognizes the record type and uses it to erase the record. Then it prompts the student in the same way as DBGET.

4.2.7 DBADD

The purpose of DBADD is to add a new occurance of a record type along with the values for its data elements to the data base. It first prompts the student to input the data element values. Once all the data has been input it stores the

new record in the data base. After completion it prompts the student in the same way as described for DBGET.

4.2.8 DBCHANGE

The purpose of DBCHANGE is to change the value of one or more data elements in a record occurance that is already in the data base. It prompts the student to enter the name of the data element to be changed and it's new value. Once it has the new value it modifys the record. Following the change a check is made to see if there is another data element to change. If so it repeats the process until the user is finished.

4.2.9 DBREPORT

The purpose of DBREPORT is to print out the contents of the data base. This subroutine must be able to adapt to some change in the subschema. It will be written so that the output will not include the contents of the DEVHIST record. The format of the report will leave a place for this information so that no changes in the format will be necessary if DEVHIST is included.

4.2.10 LEAVE

The purpose of the LEAVE command statement is to terminate the on-line session. It enables the IDMS to take over control of all areas in use by the program and to make these areas available for other programs. It terminates execution of the user interface program. The student of course will still be in the CMS environment.

4.2.11 GIT

The command statement is input as a character string and a additional module is necessary to aid in parsing of the command statement. This module is named GIT and it's purpose is to get a substring from the command statement.

To parse the command statement GIT makes use of one or more blank characters that are used as a separator between each part of the command statement. GIT looks for the first blank space in the command statement. It gets the substring which precedes this blank and assigns it to a temporary string variable. It eliminates blanks preceding a word or separating words. If the string being parsed becomes less then 5 characters long then parsing is terminated and the user is informed if a command statement part was not found in the input string. The substring which follows the blank

is assigned to another temporary string variable that is parsed in later steps.

4.3 USER INTERFACE REVIEW

The user interface as described will enable the student to satisfy all the DBMS SBLO. The student can retrieve, insert, delete, or change a record occurence of any type in the data base. The user interface can easily be adapted to some changes in the subschema, thus allowing the student to satisfy this SBLO. The interface will be interactive and remain on-line until the session is terminated by the student. It will accept input from the student through the terminal and display output. It is user friendly and supports the use of English-like command statements.

Chapter V

SOFTWARE SUPPORT TOOLS FOR THE DSS PROBLEM ENVIRONMENTS

For the most part it was not necessary to design new software for the DSS problem environments. Two EMPIRE model programs and their report programs were chosen from EMPIRE reference material. It was necessary only to design an EMPIRE control file for problem environment one and problem environment three.

The EMPIRE control file is a relatively simple program that is the source of commands rather then the terminal. In this case it is used to imput data and execute the EMPIRE model program prior to printing a report. In this chapter the software support tools chosen for the DSS problem environments will be described.

5.1 PROBLEM ENVIRONMENT ONE

The model for problem environment one analyzes the consolidated ten year financial and operating summaries of Justin Allen Petroleum Corporation from 1970 through 1979. The algorithm for the model is shown in Table 5

The items that are used in the computations are shown in figure 6

TABLE 5

Financial Summary

```
1.0 FOR column = 1970 to 1979 DO
    1.1 compute total costs
    1.2 compute operating income
    1.3 compute income(loss) before income taxes
    1.4 compute net income(loss)
1.5 END 'for column 1970 to 1979 do '
2.0 FOR rows = netsales TO net income DO
    2.1 compute the 5 year subtotal for 1970-1974
    2.2 compute the 5 year subtotal for 1975-1979
    2.3 compute the 10 year total for 1970-1979
2.4 END 'for rows netsales to net income do'
3.0 END 'algorithm financial summary'
```

```
Net Sales
Employment Costs
Materials and Service
Depreciation
Taxes other then Employment and Income Taxes
Operating Income
Interest, Dividends and other Income(expense)
Interest and other Debt Charges
Estimated Close  Down Costs
Flood Expense
Income(loss) before Income Taxes
Income Taxes
Net Income (loss)
```

Figure 6: Data items for computation of the 10 year financial summary

The report command file is used to control the format and content of the report. For problem environment one it is

used to generate a ten year financial summary report for the Justin Allen Petroleum Corporation.

The control file used for problem environment one enables the student to generate the report without having to input the data or execute the model. It opens an external file, reads data for each item, and executes the model. Then the student must enter commands that cause the report to be generated.

5.2 PROBLEM ENVIRONMENT TWO

The EMPIRE model for problem environment two analyzes a quarterly forecast of potential earnings for a new product developed by the Chemical Division of Justin Allen Petroleum Corporation called NEWLUB. The algorithm for this model is shown in table 6

The items that are to be used in the potential earnings forecast computations are shown in figure 7

The report program for problem environment two generates a simple Pro Forma income statement which is the forecast of potentail earnings. It will be necessary to enter data and execute the model for this problem environment , therefore a control file is not necessary.

TABLE 6

potential earnings forecast

```
1.0 FOR columns march TO december DO

    1.1 compute units sold

    1.2 compute sales revenue

    1.3 compute cost of sales

    1.4 compute gross profit

    1.5.0 IF column does not equal march THEN

        1.5.1 IF sales revenue is greater than sales revenue
              from the previous quarter THEN

        1.5.2 compute overhead expenses by including the amount with
              the overhead expense increase rate of 5%

        1.5.3 otherwise the overhead expense rate for the current qu
              is the same as for the previous quarter

    1.5.4 END 'if column does not equal march'

1.7 END 'for columns march to december do'

2.0 compute total sales revenue for the year

3.0 compute income tax

4.0 compute profit after tax

5.0 compute earnings per share

6.0 END 'potential earnings forecast'
```

Units Sold
Sales Revenue
Cost of Sales
Gross Profit
Overhead Expense
Profit Before Tax
Income Tax
Profit After Tax
Earnings Per Share
Market Share
Unit Price
Unit Cost
Initial Cost of Overhead
Overhead Expense Increase Rate
Tax Rate
Number of Shares
Total Units Sold in the market

Figure 6: Items for computation of the potential earnings forecast

5.3 PROBLEM ENVIRONMENT THREE

The same EMPIRE model will be used for problem environment three as as was used for problem environment two. A report will not be generated during execution of this problem environment and it will not be necessary for the student to input data.

A control file will be used that will input data and execute the model just as the student did for problem environment two. Following execution of the model, the student must choose which EMPIRE tools to use in making a decision that will affect earnings per share of the new product.

5.4 DSS PROBLEM ENVIRONMENTS : OTHER CONSIDERATIONS

In order to complete the DSS problem environments it is necessary to make some changes in the virtual machine. During regular EMPIRE execution 700k storage is required and the loader tables must be set at 4. If one desires to use the interactive debug facilities of EMPIRE, then the loader tables must be set at 5. A further explanation of this is given in appendix B.

Chapter VI

CONCLUSIONS AND FUTURE WORK

The purpose of this report has been to design and implement specific problem environments and software support tools necessary to enable satisfaction of SBLO for an undergraduate level MIS course for IS majors. These problem environments and necessary software tools have been implemented.

The decision was made to use software systems currently available at Kansas State University. For that reason the Transaction Processing System problem environment was withdrawn until such time as software is available for it.

The Office Automation problem environments were implemented without any additional software support development. This was done using the electronic mail system and the calenda system available on the Interdata 8/32.

For the DSS problem environments software support tools were available but required some modifications to enable implementation. This included some changes in the CMS virtual machine configuration and two EMPIRE control files.

The DBMS problem environments required the most extensive development of software support tools. A data base was designed along with the software to support it and a 'front

end' or user interface was designed. These tools enabled satisfaction of the SBLO and tied together the DBMS and DSS problem environments.

6.1 FUTURE WORK

Altogether 14 program and data files are necessary to implement the DSS and DBMS problem environments in the CMS environment as shown in figure 8 In figure 8 the files are in packed and compressed form. Some of the files shown in figure 8 are generated by the EMPIRE and IDMS systems.

FILENAME	FILETYPE	LRECL	RECS	BLOCKS	PURPOSE
CHEMDEV	LISTING	800	11	11	schema compile
CHEMDEV	SCHMA	80	194	20	schema
DATABASE	FILE1	496	30	19	db initialize
DBIN	DECK	80	11	2	dmlp input
DBINIT	LISTING	800	1	1	db initialize
DBQUERY	DMLP	800	23	23	'front end'
DBQUERY	LISTING	800	90	90	'front end' compile
DBQUERY	TEXT	800	56	56	run 'front end'
DEVDMCL	DMCL	80	27	3	dmcl
DEVDMCL	LISTING	800	2	2	dmcl compile
DEVDMCL	TEXT	800	2	2	run 'front end'
DICTDB	DB	800	326	326	data dictionary
DLODDB	DB	800	12	12	'' ''
DMSGDB	DB	800	7	7	'' ''
LOAD	MAP	800	5	5	'front end' compile
NUPROD	LISTING	800	2	2	subschema cmpl
NUPROD	SUBSC	80	15	2	'' ''
NUPROD	TEXT	800	3	3	'fornt end' compile
RUNPROG	EXEC	800	1	1	run 'front end'
SUBSC	LISTING	800	1	1	subschema cmpl

Figure 8: Storage requirements for packed and compressed

The same files shown in unpacked and expanded form are shown in figure 9

FILENAME	FILETYPE	LRECL	RECS	BLOCKS
CHEMDEV	LISTING	133	231	39
CHEMDEV	SCHMA	80	194	20
DATABASE	FILE1	496	30	19
DBIN	DECK	80	11	2
DBINIT	LISTING	133	17	3
DBQUERY	DMLP	80	788	79
DBQUERY	LISTING	121	1966	299
DBQUERY	TEXT	80	726	73
DEVDMCL	DMCL	80	27	3
DEVDMCL	LISTING	133	34	6
DEVDMCL	TEXT	80	30	3
DICTDB	DB	3664	1000	4580
DLODDB	DB	2496	400	1248
DMSGDB	DB	2496	200	624
LOAD	MAP	100	160	20
NUPROD	LISTING	133	31	6
NUPROD	SUBSC	80	15	2
NUPROD	TEXT	80	44	5
RUNPROG	EXEC	800	1	1
SUBSC	LISTING	133	15	3

Figure 9: Storage requirements for expanded and unpacked

By comparing the two figures it can be seen that packing and compressing save about 6500 blocks of storage space. The overhead cost created by expanding and unpacking at logon and then packing and compressing at logoff is indicated in figure 10

```
CONNECT COST $.27 CPU COST $7.64
I/O COST $4.62 TOTAL COST $12.53
```

Figure 10: Cost of storage management

In addition the statistics for disk usage while the files are unpacked and expanded are shown in figure 11

CUU	M	CYL	TYPE	BLKSIZE	FILES	BLKS USED-(%)	BLKS LEFT	BLK TOTAL
191	A	14	3350	800	31	7241-91	739	7980
190	S	48	3350	2048	228	10444-91	1076	11520
19E	Y/S	40	3350	2048	496	8217-86	1383	9600

Figure 11: Disk statistics for expanded and unpacked

6.1.1 STORAGE MANAGEMENT

An important consideration for future work is to determine, 'what the best method is to store and make these files available?'. In addition it is important to consider what is the best way to maintain the CMS virtual machine of the student. In other words how will unnecessary files be erased and needed files be saved?

One possibility for storing these files is to put the programs into a procedure library, and the data files into a data set. A CMS EXEC procedure would be used to retrieve the files needed for each problem environment and would erase unneeded files at when the student logged off.

At logon a menu of course tasks would be displayed as depicted in figure 12 The student would enter the command for the problem environment chosen and the EXEC would retrieve all necessary files and invoke EMPIRE or IDMS as necessary.

MENU for CSxxx MIS course :

DBMS_PROBLEMS

DSS_PROBLEM_1

DSS_PROBLEM_2

DSS_PROBLEM_3

Figure 12: Menu of tasks for the MIS course

It would be important to consider whether it would be expensive to keep this EXEC on-line and whether or not it would be necessary to do so. If unneeded files are not erased after their use then the amount of storage required will be considerable.

6.1.2 OTHER CONSIDERATIONS

Another consideration for future work is finding a TPS. The TPS problem environments and software support tools would have to be designed and implemented as necessary.

In Neil Strunk's report (STRUNK : 82) he was concerned with developing a course that would give the business student an adequate knowledge of a MIS. He also suggested that his findings could be useful for the Computer Science department in developing a MIS course for it's curriculum. I believe that an area of future work is to consider how the problem environments and software support tools developed for the Computer Science department IS majors could be adapted to the needs of the business student.

Also of importance for future work is the trial use of these problem environments and software tools in an actual MIS course. This would enable one to evaluate whether or not they really satisfy the SBLO chosen for the MIS course.

Appendix A

SCHEMA AND SUBSCHEMA SOFTWARE SUPPORT TOOLS

Release 5.7 of IDMS was implemented at Kansas State University in June 1983. It is the purpose of this appendix to describe the EXECs used to compile and run the schema, subschema, DMCL, and DMLP programs needed to complete the DBMS problem environments.

A.0.3 HOW TO IMPLEMENT THE DATA BASE SUPPORT TOOLS

In order to implement a data base using IDMS in CMS there are several EXEC commands which have been developed and implemented by the Computing center at Cardwall hall. These commands are shown in figure 13

The special logon command is used only during compilation of the schema, subschema, DMCL, and data base initialization. The LINKIDMS enables one to access and use the EXEC's for compiling and running the data base and user 'front end'. The copyfile command makes a copy of an initialized data dictionary and puts it on the users A disk. The IDMSCHMA command compiles the schema which must be file type schma. It produces an output listing file called 'CHEMDEV

```
1.  LOGON VMxxx 1m noipl#def t3350 191 13#ipl vmstart
2.  LINKIDMS
3.  copyfile * db i = = a
4.  IDMSCHMA CHEMDEV
5.  IDMSDMCL DEVDMCL
6.  IDMSUBSC NUPROD
7.  IDMSDMLP DBQUERY
8.  RUNPROG DBQUERY
```

Figure 13: Exec Commands to Compile and Run the User Front End

LISTING'. This file is important only if there are errors in the schema DDL.

The IDMSDMCL compiles the DMCL which must be of file type DMCL. It also produces a listing file called "DEVDMCL LISTING'. This also is important only if there are errors in the DMCL.

The IDMSUBSC command is used to compile the subschema. It produces a listing file and a text file. The text file is important in later steps but the listing file is not. If there are compile time errors in the subschema then a command must be used to delete the subschema from the data dictionary. The command that is used to delete the subschema from the data dictionary is IDMSDELS. The subschema must be recompiled following corrections in the code for it.

The IDMSDMLP EXEC is used to compile the user 'front end' called DBQUERY. It produces a listing file that is important only in if compile time errors are present.

Throughout the preceeding steps the EXECs are entering information about the database into the data dictionary. After the application program has been successfully compiled the data dictionary is not needed for execution of the user 'front end'. The data dictionary should be packed and compressed and stored until such time as changes are to be made in the schema, subschema, DMCL, or application program. Text files that are created throughout the compilation process must be be maintained in a useable form during execution of the user 'front end'. Also the file called 'LOAD MAP A5' must be retained for the same use.

The RUNPROG EXEC initializes the users data base, enters the appropriate FILEDEFs and executes the user 'front end'. The input file is called 'DBIN DECK' and it contains the data necessary for program execution. The output from execution of DBQUERY is in a file called 'DBOUT LISTING'.

The code for the schema, subschema, and DMCL are in this appendix and the code for the user 'front end' is in appendix B. It was written in PL/I because it can be run within the CMS environment and IDMS has a PL/I preprocessor available.

TABLE 7

The Schema Used for the DBMS Problem Environments

```
SCHEMA DESCRIPTION.
SCHEMA NAME IS CHEMDEV.

AUTHOR.          ROBERT A. BIRCHARD
DATE.            06/24/83
INSTALLATION.    KANSAS STATE UNIVERSITY
                 COMPUTER SCIENCE DEPARTMENT

REMARKS.         THIS SCHEMA WAS DESIGNED TO SATISFY THE
                 STUDENT BEHAVIORAL LEARNING OBJECTIVES
                 OF A MANAGEMENT INFORMATION SYSTEMS
                 COURSE FOR INFORMATION SYSTEMS MAJORS
                 AT KANSAS STATE UNIVERSITY.

FILE DESCRIPTION.

          FILE NAME IS IDMS-FILE1
                    ASSIGN TO SYS010
                    DEVICE TYPE IS 3350.

          FILE NAME IS JOURNAL
                    ASSIGN TO SYS009
                    DEVICE TYPE IS 2400.

AREA DESCRIPTION.

     AREA NAME IS PRODUCT-REGION
               RANGE IS 1001 THRU 1010
               WITHIN IDMS-FILE1
               FROM 1 THRU 10.

     AREA NAME IS MARKET-REGION
               RANGE IS 1011 THRU 1020
               WITHIN IDMS-FILE1
               FROM 11 THRU 20.

     AREA NAME IS HIST-REGION
               RANGE IS 1021 THRU 1030
               WITHIN IDMS-FILE1
               FROM 21 THRU 30.

RECORD DESCRIPTION.

     RECORD NAME IS PRODUCT.
     RECORD ID IS 100.
     LOCATION MODE IS CALC USING PRODNAME
                          DUPLICATES ARE NOT ALLOWED.

     WITHIN PRODUCT-REGION AREA.
```

```
           03  PRODIDNO          PIC X(8).
               COMMENT 'PRODUCT ID#. THERE IS A UNIQUE ID#.
-                      'FOR EACH PRODUCT IN THE FILE..
           03  PRODNAME                PIC X(12).
           03  PRODSTATUS              PIC X(2).
               COMMENT 'THE PRODUCT STATUS. THIS IS USED TO
-                      'INDICATE THE MARKETING PHASE.
-                      'RESEARCH=01, TRIAL=02,
-                      'FULL PRODUCTION=03, CUSTOM ORDERS=0
-                      'DISCONTINUED=05'.

      RECORD NAME IS MARKET.
      RECORD ID IS 300.
      LOCATION MODE IS CALC USING MKTNAME
                              DUPLICATES ARE NOT ALLOWED.

      WITHIN MARKET-REGION AREA.

           03  MKTIDNO                 PIC X(8).
               COMMENT 'MARKET ID#. THERE IS A UNIQUE ID# F
-                      'EACH MARKETING REGION'.
           03  MKTNAME                 PIC X(12).
           03  OVHDCOST                PIC X(5).
               COMMENT 'OVERHEAD COST. THE ESTIMATED COST F
-                      'THE FIRST QUARTER OF THE FORECAST
-                      'PERIOD'.
           03  OVHDRATE                PIC X(3).
               COMMENT 'OVERHEAD INCREASE RATE. THE PERCENT
-                      'RATE AT WHICH OVERHEAD COSTS ARE
-                      'EXPECTED TO INCREASE DURING THE
-                      'FORECAST PERIOD'.
           03  TAXRATE                 PIC X(3).
           03  SHARES                  PIC X(5).
               COMMENT 'THE EFFECTIVE NUMBER OF COMMON STOC
-                      'SHARES TO BE USED FOR THE COMPUTATI
-                      'OF EARNINGS PER SHARE'.
           03  UCOST                   PIC X(5).
               COMMENT 'UNIT COST. THE ESTIMATED COST OF
-                      'PRODUCING EACH UNIT OF PRODUCT'.
           03  PRICE                   PIC X(5).
               COMMENT 'UNIT PRICE. THE PLANNED SELLING
-                      'PRICE PER UNIT OF PRODUCT'.

      RECORD NAME IS MKTHIST.
      RECORD ID IS 310.
      LOCATION MODE IS VIA HIST SET.
      WITHIN HIST-REGION.

           03  PERIOD                  PIC X(8).
             COMMENT 'PERIOD INDICATES THE TIME PERIOD
-                    'OF THE HISTORICAL SALES DATA FOR
-                    'PRODUCTS IN THE MARKET PLACE. EXAMPLE
-                    '78THRU83 INDICATES THE YEARS COVERED'
```

```
           03  YEAR1                      PIC X(23).
               COMMENT 'YEAR1 THRU YEAR5 EACH HOLD
-                      'HISTORICAL DATA BY QUARTER FOR ONE
-                      'YEAR WITHIN THE PERIOD'.
           03  YEAR2                      PIC X(23).
           03  YEAR3                      PIC X(23).
           03  YEAR4                      PIC X(23).
           03  YEAR5                      PIC X(23).

       RECORD NAME IS MKTEST.
       RECORD ID IS 320.
       LOCATION MODE IS VIA EST SET.
       WITHIN MARKET-REGION AREA.

           03  FDATE                      PIC X(4).
               COMMENT 'DATE OF THE FORECAST PERIOD'.
           03  UNITS                      PIC X(6).
               COMMENT 'UNITS SOLD. THE FORECAST NUMBER
-                        'OF UNITS OF PRODUCT SOLD'.
           03  SALES                      PIC X(8).
               COMMENT 'SALES REVENUE. UNITS*PRICE'.
           03  COS                        PIC X(7).
               COMMENT 'COST OF SALES. UNITS*UCOST'.
           03  GPROF                      PIC X(6).
               COMMENT 'GROSS PROFIT. SALES-COS'.
           03  OVHDEXPS                   PIC X(6).
               COMMENT 'OVERHEAD EXPENSE. OVERHEAD EXPENSES
-                      'FOR EACH QUARTER OF THE FORECAST
-                      'PERIOD'.
           03  PBT                        PIC X(6).
               COMMENT 'PROFIT BEFORE TAX. GPROF-OVHDEXPS'.
           03  TAX                        PIC X(6).
               COMMENT 'INCOME TAX EXPENSE. PBT*TAXRATE'.
           03  PAT                        PIC X(6).
               COMMENT 'PROFIT AFTER TAX. PBT-TAX'.
           03  EARN                       PIC X(4).

       RECORD NAME IS DEVHIST.
       RECORD ID IS 200.
       LOCATION MODE IS CALC USING PRODEVNAME
                             DUPLICATES ARE NOT ALLOWED.

       WITHIN HIST-REGION AREA.

           03  PRODEVNO                   PIC X(8).
               COMMENT 'PRODUCT DEVELOPMENT#. IT IS
-                      'EQUIVALENT TO PRODID#'.
           03  PRODEVNAME                 PIC X(12).
          COMMENT 'PRODEVNAME IS EQUIVALENT TO
-                 'PRODNAME. PRODUCT DEVELOPMENT NAME'.
           03  PATDOCNO                   PIC X(8).
          COMMENT 'PATENT DOCUMENT #. THIS IS A LIBRARY
-                 'CATALOG NUMBER FOR PATENT INFORMATION
-                 'CONCERNING THE PRODUCT'.
```

```
           03  DEVDOCNO                    PIC X(8).
         COMMENT 'DEVELOPMENT DOCUMENT NUMBER. THIS IS A
-                'LIBRARY CATALOG NUMBER FOR DEVELOPMENT
-                'HISTORY INFORMATION ABOUT THE PRODUCT'.

 SET DESCRIPTION.

      SET NAME IS PROD-MKT.
      ORDER IS NEXT.
      MODE IS CHAIN.
      OWNER IS PRODUCT
           NEXT DBKEY POSITION IS 1.

      MEMBER IS MARKET
           MANDATORY AUTOMATIC
           NEXT DBKEY POSITION IS 1.


      SET NAME IS HIST.
      ORDER IS NEXT.
      MODE IS CHAIN.
      OWNER IS MARKET
           NEXT DBKEY POSITION IS 2.

      MEMBER IS MKTHIST
           MANDATORY AUTOMATIC
           NEXT DBKEY POSITION IS 1.


      SET NAME IS EST.
      ORDER IS NEXT.
      MODE IS CHAIN.
      OWNER IS MARKET
           NEXT DBKEY POSITION IS 3.

      MEMBER IS MKTEST
           OPTIONAL AUTOMATIC
           NEXT DBKEY POSITION IS 1.
```

TABLE 8

The DMCL for the Product Data Base

```
DEVICE-MEDIA DESCRIPTION.

DEVICE-MEDIA NAME IS DEVDMCL
                  OF SCHEMA NAME CHEMDEV.

AUTHOR.          ROBERT A. BIRCHARD
DATE.            JUNE 1983
INSTALLATION.    KANSAS STATE UNIVERSITY
                 COMPUTER SCIENCE DEPARTMENT

REMARKS.  THE DMCL TABLES RESULTING FROM THESE SOURCE
          STATEMENTS WILL ALLOW RUN TIME ACCESS TO ALL
          FILES MAKING UP THE CHEMDEV DATA BASE.


BUFFER SECTION.

              BUFFER NAME IS DEVBUFFER
              PAGE CONTAINS 496 CHARACTERS
              BUFFER CONTAINS 8 PAGES.


AREA SECTION.

            COPY PRODUCT-REGION AREA.
            COPY MARKET-REGION AREA.
            COPY HIST-REGION AREA.
```

TABLE 9

The Subschema for the Product Data Base

```
ADD SUBSCHEMA NAME IS NUPROD
     OF SCHEMA NAME IS CHEMDEV
       DMCL NAME IS DEVDMCL.

ADD AREA PRODUCT-REGION.
ADD AREA MARKET-REGION.
ADD AREA HIST-REGION.
ADD RECORD PRODUCT.
ADD RECORD MARKET.
ADD RECORD MKTHIST.
ADD RECORD MKTEST.
ADD SET PROD-MKT.
ADD SET HIST.
ADD SET EST.
GENERATE.
```

Appendix B

THE USER FRONT END

The purpose of this appendix is to document the dmlp user 'front end'. It contains the PL/I code which has IDMS DML commands and statements embeded in it. Also it contains a sample session and the output file produced by the session.

PL/I was chosen because it can be run interactively in the CMS environment and a PL/I preprocessor is available at Kansas State University for IDMS. The 'front end' is maintained on-line by using a conditional loop in the main program which will not terminate until the command 'LEAVE' is entered by the user. The program sends output to the terminal by means of the PL/I DISPLAY statement. DISPLAY statement syntax alows an option which enables the user to send input to the program through the terminal. This is the REPLY option. The communication that the DISPLAY REPLY statement is very useful but has some limitations. The input and output arguments are character string type with no formatting options. The compiler will not allow a DISPLAY argument to exceed 72 characters in length and the argument value can not exceed 128 characters in length.

In order to save a copy of the session activity it is also necessary to provided an output file. This output file is 'DBJUT LISTING' and is sent to the users A disk for viewing after the session is terminated. If the user wishes to save a copy of the terminal session the command to use is 'CP SPOOL CONSOLE START'. At any time the user wishes the command 'CP SPOOL CONSOLE CLOSE STOP' is entered and whatever follows will not be recorded. When 'CP SPOOL CONSOLE CLOSE STOP' is entered the CRT will notify the user that a file has been saved on his virtual reader. The command 'READCARD FN FTYPE ' must be entered to keep the session record.

TABLE 10

Main Program and Declararions

```
  DBQUERY : PROCEDURE OPTIONS(MAIN);
/*DMLIST*/
/*SCHEMA_COMMENTS*/

  DECLARE
        1  CARDIN_REC              STATIC,
           2  TYP          CHAR (7),
           2  FILL1         CHAR (58),
           2  FTYP          CHAR(6),
           2  FILL2         CHAR (9),

        END_OF_AREA         CHAR (4) INIT ('0307'),
        END_OF_SET          CHAR (4) INIT ('0307'),
        INREC               CHAR (80) DEFINED CARDIN_REC,
        OK                  CHAR (4) INIT ('0000');
    DECLARE (TRUE) BIT(1) INIT ('1'B);
    DECLARE FOUND BIT(1) INIT('0'B) EXTERNAL;
    DECLARE NOGIT BIT(1) INIT('0'B);
    DECLARE (FALSE) BIT(1) INIT ('0'B);

    DECLARE CMD CHAR(72) EXTERNAL,
            CTYP CHAR(8) VARYING EXTERNAL,
            RID CHAR(12) VARYING EXTERNAL,
            RN CHAR(8)   VARYING EXTERNAL,
            PID CHAR(12) VARYING EXTERNAL,
            MID CHAR(12) VARYING EXTERNAL;
    DECLARE TCARD CHAR(72) VARYING EXTERNAL,
            TSTRING CHAR(12) VARYING EXTERNAL;
    DECLARE
       1  PRODBUF_REC                    STATIC,
         3  P1                           CHAR(8) INIT(' '),
         3  P2                           CHAR(12) INIT(' '),
         3  P3                           CHAR(2) INIT(' '),

       1  MKTBUF_REC                     STATIC,
         3  M1                           CHAR(8) INIT(' '),
         3  M2                           CHAR(12) INIT(' '),
         3  M3                           CHAR(5) INIT(' '),
         3  M4                           CHAR(3) INIT(' '),
         3  M5                           CHAR(3) INIT(' '),
         3  M6                           CHAR(5) INIT(' '),
         3  M7                           CHAR(5) INIT(' '),
         3  M8                           CHAR(5) INIT(' '),

       1  MKTHISTBUF_REC                 STATIC,
         3  MH1                          CHAR(8) INIT(' '),
         3  MH2                          CHAR(23) INIT(' '),
         3  MH3                          CHAR(23) INIT(' '),
         3  MH4                          CHAR(23) INIT(' '),
```

```
        3  MH5                          CHAR(23)  INIT(' '),
        3  MH6                          CHAR(23)  INIT(' '),

      1  MKTESTBUF_REC                  STATIC,
        3  ME1                          CHAR(4)  INIT(' '),
        3  ME2                          CHAR(12)  INIT(' '),
        3  ME3                          CHAR(6)  INIT(' '),
        3  ME4                          CHAR(8)  INIT(' '),
        3  ME5                          CHAR(7)  INIT(' '),
        3  ME6                          CHAR(6)  INIT(' '),
        3  ME7                          CHAR(6)  INIT(' '),
        3  ME8                          CHAR(6)  INIT(' '),
        3  ME9                          CHAR(6)  INIT(' '),
        3  ME10                          CHAR(6)  INIT(' '),
        3  ME11                         CHAR(4)  INIT(' ');

   DECLARE
      PRODBUF CHAR(22) DEFINED PRODBUF_REC,
      MKTBUF CHAR(46) DEFINED MKTBUF_REC,
      MKTHISTBUF CHAR(123) DEFINED MKTHISTBUF_REC,
      MKTESTBUF CHAR(71) DEFINED MKTESTBUF_REC;

DECLARE (IDMS,ABORT) OPTIONS (INTER,ASM) ENTRY;
DECLARE ( NUPROD SUBSCHEMA, CHEMDEV SCHEMA VERSION 1)
        MODE (BATCH);
INCLUDE IDMS ( SUBSCHEMA_DESCRIPTION);

DECLARE
       PRODUCT_REC CHAR (24) DEFINED PRODUCT,
       MARKET_REC CHAR (48) DEFINED MARKET,
       MKTEST_REC CHAR (71) DEFINED MKTEST,
       MKTHIST_REC CHAR (123) DEFINED MKTHIST;

INCLUDE IDMS (SUBSCHEMA_BINDS);
SUBSCHEMA_CTRL.PROGRAM = 'DBQUERY' ;
BIND RUN_UNIT;
BIND RECORD (PRODUCT);
BIND RECORD (MARKET);
BIND RECORD (MKTEST);
BIND RECORD (MKTHIST);

INCLUDE IDMS(IDMS_STATUS);
READY EXCLUSIVE UPDATE;
CALL IDMS_STATUS;

/* BEGIN MAIN PROGRAM */

CALL DBUILD;
CALL DBMENU;
DO UNTIL (CMD = 'LEAVE   ');
   DISPLAY (' ');
   DISPLAY ('ENTER COMMAND')
      REPLY (CMD);
   DISPLAY (' ');
```

```
   CALL PROC_CMD;
END; /* DO UNTIL COM = 'LEAVE   '; */
FINISH;
CALL IDMS_STATUS;
END DBQUERY;  /* END MAIN PROGRAM */
```

TABLE 11

DBCHG_REC

```
DBCHG_REC : PROCEDURE;
  DECLARE RN CHAR(8) VARYING EXTERNAL;
   IF RN = 'PRODUCT' THEN
     DO;
        MODIFY RECORD (PRODUCT);
        CALL IDMS_STATUS;
     END;
   ELSE IF RN = 'MARKET' THEN
          DO;
             MODIFY RECORD (MARKET);
             CALL IDMS_STATUS;
          END;
     ELSE IF RN = 'MKTEST' THEN
            DO;
               MODIFY RECORD (MKTEST);
               CALL IDMS_STATUS;
            END;
       ELSE IF RN = 'MKTHIST' THEN
              DO;
                 MODIFY RECORD (MKTHIST);
                 CALL IDMS_STATUS;
              END;
END DBCHG_REC;
```

TABLE 12

DBCHG_VAL

```
DBCHG_VAL :      PROCEDURE;

   DECLARE DN CHAR(12) VARYING EXTERNAL,
           DVAL CHAR(24) VARYING EXTERNAL;

IF RN = 'PRODUCT' THEN
  DO;
     IF DN = 'PRODIDNO' THEN PRODIDNO  = DVAL;
     ELSE IF DN = 'PRODNAME' THEN PRODNAME = DVAL;
      ELSE IF DN = 'PRODSTATUS' THEN PRODSTATUS = DVAL;
         ELSE DO;
                 DISPLAY
                 (DN || '  IS NOT A CORRECT DATA ITEM NAME');
                 DISPLAY ('FOR THE PRODUCT RECORD');
                 CHG_OK = FALSE;
               END;
  END; /* RN = 'PRODUCT' */
ELSE IF RN = 'MARKET' THEN
  DO;
     IF DN = 'MKTIDNO' THEN MKTIDNO = DVAL;
     ELSE IF DN = 'MKTNAME' THEN MKTNAME = DVAL;
       ELSE IF DN = 'OVHDCOST' THEN OVHDCOST = DVAL;
         ELSE IF DN = 'OVHDRATE' THEN OVHDRATE = DVAL;
           ELSE IF DN = 'TAXRATE' THEN TAXRATE = DVAL;
             ELSE IF DN = 'SHARES' THEN SHARES = DVAL;
               ELSE IF DN = 'UCOST' THEN UCOST = DVAL;
                ELSE IF DN = 'PRICE' THEN PRICE = DVAL;
                   ELSE DO;
                           DISPLAY (DN || '  IS NOT A CORRECT');
                           DISPLAY ('DATA ITEM NAME');
                           DISPLAY ('FOR THE MARKET RECORD');
                           CHG_OK = FALSE;
                         END;
  END; /* RN = 'MARKET' */
  ELSE IF RN = 'MKTHIST' THEN
         DO;
       IF DN = 'PERIOD' THEN PERIOD = DVAL;
       ELSE IF DN = 'YEAR1' THEN YEAR1 = DVAL;
         ELSE IF DN = 'YEAR2' THEN YEAR2 = DVAL;
           ELSE IF DN = 'YEAR3' THEN YEAR3 = DVAL;
             ELSE IF DN = 'YEAR4' THEN YEAR4 = DVAL;
               ELSE IF DN = 'YEAR5' THEN YEAR5 = DVAL;
                   ELSE DO;
                           DISPLAY (DN || '  IS NOT A CORRECT');
                           DISPLAY ('DATA ITEM NAME');
                           DISPLAY ('FOR THE MKTHIST RECORD');
                           CHG_OK = FALSE;
                         END;
    END; /* RN = 'MKTHIST' */
```

```
    ELSE IF RN = 'MKTEST' THEN
 DO;
    IF DN = 'FDATE' THEN FDATE = DVAL;
    ELSE IF DN = 'MKTSHR' THEN MKTSHR = DVAL;
      ELSE IF DN = 'UNITS' THEN UNITS = DVAL;
        ELSE IF DN = 'SALES' THEN SALES = DVAL;
          ELSE IF DN = 'COS' THEN COS = DVAL;
            ELSE IF DN = 'GPROF' THEN GPROF = DVAL;
              ELSE IF DN = 'PBT' THEN PBT = DVAL;
                ELSE IF DN = 'PAT' THEN PAT = DVAL;
          ELSE IF DN = 'OVHDEXPS' THEN OVHDEXPS = DVAL;
                    ELSE IF DN = 'TAX' THEN TAX = DVAL;
                  ELSE IF DN = 'EARN' THEN EARN = DVAL;
                    ELSE DO;
                            DISPLAY (DN || ' IS NOT A CORRECT');
                            DISPLAY ('DATA ITEM NAME');
                            DISPLAY ('FOR THE MKTEST  RECORD');
                            CHG_OK = FALSE;
                          END;
      END; /* RN = 'MKTEST' */
END DBCHG_VAL; /* END OF PROCEDURE CHGVAL */
```

TABLE 13

DBPRINT

```
DBPRINT : PROCEDURE(PBUF,MBUF,MEBUF,MHBUF);

 DECLARE PBUF CHAR(22) ,
         MBUF CHAR(46) ,
         MHBUF CHAR(123) ,
         MEBUF CHAR(71) ;

  PUT EDIT (P1,M1,ME1,MH1)
           (COL(1),A(8),X(6),A(4),X(10),A(4),X(10),A(8));
  PUT EDIT (P2,M2,ME2,MH2)
           (COL(1),A(12),X(2),A(12),X(2),A(12),X(2),A(23));
  PUT EDIT (P3,M3,ME3,MH3)
           (COL(1),A(2),X(12),A(5),X(9),A(8),X(6),A(23));
  PUT EDIT (M4,ME4,MH4)  (COL(15),A(3),X(11),A(7),X(7),A(23));
  PUT EDIT (M5,ME5,MH5)  (COL(15),A(3),X(11),A(6),X(8),A(23));
  PUT EDIT (M6,ME6,MH6)  (COL(15),A(5),X(9),A(6),X(8),A(23));
  PUT EDIT (M7,ME7)  (COL(15),A(5),X(9),A(6));
  PUT EDIT (M8,ME8)  (COL(15),A(5),X(9),A(6));
  PUT EDIT (ME9)  (COL(29),A(6));
  PUT EDIT (ME10)  (COL(29),A(6));
  PUT EDIT (ME11)  (COL(29),A(4));
  PUT SKIP;

  PBUF = ' ';
  MBUF = ' ';
  MEBUF = ' ';
  MHBUF = ' ';

END DBPRINT;
```

TABLE 14

DBINIT

```
DBINIT : PROCEDURE(RN);
 DECLARE RN CHAR(8),
         TREC CHAR(72) INIT(' ');

  IF RN = 'PRODUCT' THEN
    DO;
       PRODUCT_REC = SUBSTR(INREC,8);
       STORE RECORD (PRODUCT);
       CALL IDMS_STATUS;
    END; /* RN = 'PRODUCT' */
  ELSE IF RN = 'MARKET' THEN
         DO;
            MARKET_REC = SUBSTR(INREC,8);
            STORE RECORD(MARKET);
            CALL IDMS_STATUS;
         END; /* RN = 'MARKET' */
    ELSE IF RN = 'MKTEST' THEN
           DO;
              MKTEST_REC = SUBSTR(INREC,8);
              STORE RECORD(MKTEST);
              CALL IDMS_STATUS;
           END; /* RN = 'MKTEST' */
      ELSE IF RN = 'MKTHIST' THEN
        DO;
      IF FTYP = 'PERIOD' THEN PERIOD = SUBSTR(INREC,8);
       ELSE IF FTYP = 'YEAR1 ' THEN YEAR1 = SUBSTR(INREC,8);
        ELSE IF FTYP = 'YEAR2 ' THEN YEAR2 = SUBSTR(INREC,8);
         ELSE IF FTYP = 'YEAR3 ' THEN YEAR3 = SUBSTR(INREC,8);
          ELSE IF FTYP = 'YEAR4 ' THEN YEAR4 = SUBSTR(INREC,8);
           ELSE IF FTYP = 'YEAR5 ' THEN
                       DO;
                          YEAR5 = SUBSTR(INREC,8);
                          STORE RECORD(MKTHIST);
                          CALL IDMS_STATUS;
                       END;
              END; /* RN = 'MKTHIST' */
     ELSE DISPLAY ( TYP || '  IS NOT A CORRECT RECORD NAME.');
 TREC = SUBSTR(INREC,8);
END DBINIT;
```

TABLE 15

DBDELETE

```
DBDELETE : PROCEDURE(RN);
 DECLARE RN CHAR(8);

    IF RN = 'PRODUCT' THEN
      DO;
         ERASE RECORD (PRODUCT) ALL;
         CALL IDMS_STATUS;
      END;
    ELSE IF RN = 'MARKET' THEN
           DO;
              ERASE RECORD (MARKET) ALL;
              CALL IDMS_STATUS;
           END;
      ELSE IF RN = 'MKTEST' THEN
             DO;
                ERASE RECORD (MKTEST) ALL;
                CALL IDMS_STATUS;
             END;
        ELSE IF RN = 'MKTHIST' THEN
               DO;
                  ERASE RECORD (MKTHIST) ALL;
                  CALL IDMS_STATUS;
               END;

END DBDELETE;
```

TABLE 16

DBGET

```
DBGET : PROCEDURE(RN);
  DECLARE RN CHAR(8);
  DISPLAY ('RECORD TYPE : ' || RN ) ;
  DISPLAY ('==============') ;
  DISPLAY (' ');
  PUT EDIT ('RECORD TYPE : ' || RN ) (COL(1),A);
  PUT EDIT ('==============') (COL(1),A);
  PUT SKIP;

  IF RN = 'PRODUCT' THEN
    DO;
       DISPLAY ('PRODID#       ' || PRODIDNO) ;
       DISPLAY ('PRODNAME      ' || PRODNAME) ;
       DISPLAY ('PRODSTATUS    ' || PRODSTATUS) ;
       DISPLAY (' ');
       PUT EDIT ('PRODID#       ' || PRODIDNO) (COL(1),A);
       PUT EDIT ('PRODNAME      ' || PRODNAME) (COL(1),A);
       PUT EDIT ('PRODSTATUS    ' || PRODSTATUS) (COL(1),A);
    END; /* RN = 'PRODUCT' */
  ELSE IF RN = 'MARKET' THEN
         DO;
            DISPLAY ('MKTID#      ' || MKTIDNO) ;
            DISPLAY ('MKTNAME     ' || MKTNAME) ;
            DISPLAY ('OVHDCOST    ' || OVHDCOST) ;
            DISPLAY ('OVHDRATE    ' || OVHDRATE) ;
            DISPLAY ('TAXRATE     ' || TAXRATE) ;
            DISPLAY ('SHARES      ' || SHARES) ;
            DISPLAY ('UCOST       ' || UCOST) ;
            DISPLAY ('PRICE       ' || PRICE) ;
            DISPLAY (' ');
            PUT EDIT ('MKTID#      ' || MKTIDNO) (COL(1),A);
            PUT EDIT ('MKTNAME     ' || MKTNAME) (COL(1),A);
            PUT EDIT ('OVHDCOST    ' || OVHDCOST) (COL(1),A);
            PUT EDIT ('OVHDRATE    ' || OVHDRATE) (COL(1),A);
            PUT EDIT ('TAXRATE     ' || TAXRATE) (COL(1),A);
            PUT EDIT ('SHARES      ' || SHARES) (COL(1),A);
            PUT EDIT ('UCOST       ' || UCOST) (COL(1),A);
            PUT EDIT ('PRICE       ' || PRICE) (COL(1),A);
         END; /* RN = 'MARKET' */
    ELSE IF RN = 'MKTEST' THEN
           DO;
              DISPLAY ('FDATE       ' || FDATE) ;
              DISPLAY ('MKTSHR      ' || MKTSHR) ;
              DISPLAY ('UNITS       ' || UNITS) ;
              DISPLAY ('SALES       ' || SALES) ;
              DISPLAY ('COS         ' || COS) ;
              DISPLAY ('GPROF       ' || GPROF) ;
              DISPLAY ('OVHDEXPS    ' || OVHDEXPS) ;
              DISPLAY ('PBT         ' || PBT) ;
```

```
            DISPLAY ('TAX          ' || TAX) ;
            DISPLAY ('PAT          ' || PAT) ;
            DISPLAY ('EARN         ' || EARN) ;
            DISPLAY (' ');
            PUT EDIT ('FDATE       ' || FDATE) (COL(1),A);
            PUT EDIT ('MKTSHR      ' || MKTSHR) (COL(1),A);
            PUT EDIT ('UNITS       ' || UNITS) (COL(1),A);
            PUT EDIT ('SALES       ' || SALES) (COL(1),A);
            PUT EDIT ('COS         ' || COS) (COL(1),A);
            PUT EDIT ('GPROF       ' || GPROF) (COL(1),A);
            PUT EDIT ('OVHDEXPS    ' || OVHDEXPS) (COL(1),A);
            PUT EDIT ('PBT         ' || PBT) (COL(1),A);
            PUT EDIT ('TAX         ' || TAX) (COL(1),A);
            PUT EDIT ('PAT         ' || PAT) (COL(1),A);
            PUT EDIT ('EARN        ' || EARN) (COL(1),A);
         END; /* RN = 'MKTEST' */
      ELSE IF RN = 'MKTHIST' THEN
           DO;
              DISPLAY ('PERIOD    ' || PERIOD) ;
              DISPLAY ('YEAR1     ' || YEAR1) ;
              DISPLAY ('YEAR2     ' || YEAR2) ;
              DISPLAY ('YEAR3     ' || YEAR3) ;
              DISPLAY ('YEAR4     ' || YEAR4) ;
              DISPLAY ('YEAR5     ' || YEAR5) ;
              DISPLAY (' ');
              PUT EDIT ('PERIOD    ' || PERIOD) (COL(1),A);
              PUT EDIT ('YEAR1     ' || YEAR1) (COL(1),A);
              PUT EDIT ('YEAR2     ' || YEAR2) (COL(1),A);
              PUT EDIT ('YEAR3     ' || YEAR3) (COL(1),A);
              PUT EDIT ('YEAR4     ' || YEAR4) (COL(1),A);
              PUT EDIT ('YEAR5     ' || YEAR5) (COL(1),A);
           END; /* RN = 'MKTHIST' */
   PUT SKIP;
END DBGET;
```

TABLE 17

DBCHANGE

```
DBCHANGE : PROCEDURE;

  DECLARE TCARD CHAR(72) VARYING EXTERNAL,
          TSTRING CHAR(12) VARYING EXTERNAL,
          DN       CHAR (12) VARYING EXTERNAL,
          DVAL     CHAR (24) VARYING EXTERNAL;
  DECLARE
      T_CHG CHAR(36) INIT(' '),
      I FIXED DECIMAL;
  DECLARE
      CHG_OK BIT(1) INIT('1'B),
      ANS     CHAR(3);

  DISPLAY
  ('DO YOU WANT TO CHANGE A VALUE IN ' || RN || ' RECORD?');
  DISPLAY('YES/NO')
      REPLY (ANS);
  DISPLAY(' ');
  IF (ANS ¬= 'NO') THEN IF (ANS ¬= 'YES') THEN
  DISPLAY ('YOUR ONLY VALID ANSWERS ARE YES OR NO.')
      REPLY (ANS);
  DISPLAY (' ');
  DO WHILE (ANS = 'YES');
      DISPLAY ('TYPE THE NAME AND THE NEW VALUE FOR THE ITEM');
      DISPLAY ('YOU WANT TO CHANGE, THEN PRESS ENTER')
          REPLY (T_CHG);
      DISPLAY (' ');
      TCARD = T_CHG;
      CALL GIT;
      DN = TSTRING;
      CALL GIT;
      DVAL = TSTRING;
      IF RN = 'MKTHIST' THEN IF (DN ¬= 'PERIOD') THEN
        DO I = 1 TO 3 ;
          CALL GIT;
          DVAL = DVAL || ' ' || TSTRING;
        END;
      CALL DBCHG_VAL;
      DN = ' '; DVAL = ' ';
      DISPLAY ('DO YOU WANT TO CHANGE ANY OTHER DATA ITEMS');
      DISPLAY ('IN THE ' || RN || ' RECORD? YES/NO')
          REPLY (ANS);
      DISPLAY (' ');
      IF (ANS ¬= 'NO') THEN IF (ANS ¬= 'YES') THEN
      DISPLAY ('YOUR ONLY VALID ANSWERS ARE YES OR NO.')
          REPLY (ANS);
      DISPLAY (' ');
  END; /* DO WHILE ANS = 'YES' */
  IF (CHG_OK = TRUE) THEN   CALL DBCHG_REC;
```

```
END DBCHANGE;
```

TABLE 18

DBREPORT

```
DBREPORT : PROCEDURE;

   DECLARE (ELAST,HLAST) BIT (1) INIT ('0'B),
           MORE BIT(1) INIT('1'B);
   PUT EDIT
('PRODUCT      MARKET       MKTEST            MKTHIST') (COL(1),A);
   PUT EDIT
('=======      ======       ======            =======') (COL(1),A);
   OBTAIN FIRST RECORD(PRODUCT) AREA(PRODUCT_REGION);
   CALL IDMS_STATUS;
   DO WHILE ( ERROR_STATUS = OK);
       PRODBUF = PRODUCT_REC;
       OBTAIN FIRST RECORD(MARKET) SET(PROD_MKT);
       IF ( ERROR_STATUS ¬= OK) THEN   CALL DBPRINT
           (PRODBUF,MKTBUF,MKTESTBUF,MKTHISTBUF);
       DO WHILE (ERROR_STATUS = OK);
           MKTBUF = MARKET_REC;
           OBTAIN FIRST RECORD(MKTEST) SET(EST);
           IF (ERROR_STATUS ¬= OK) THEN ELAST = TRUE;
           ELSE MKTESTBUF = MKTEST_REC;
           OBTAIN FIRST RECORD(MKTHIST) SET(HIST);
           IF (ERROR_STATUS ¬= OK) THEN HLAST = TRUE;
           ELSE MKTHISTBUF = MKTHIST_REC;
           IF (ELAST = TRUE) THEN IF (HLAST = TRUE) THEN
             DO;
              CALL DBPRINT(PRODBUF,MKTBUF,MKTESTBUF,MKTHISTBUF);
                MORE = FALSE;
             END;
           DO WHILE (MORE = TRUE);
             CALL DBPRINT(PRODBUF,MKTBUF,MKTESTBUF,MKTHISTBUF);
               OBTAIN NEXT RECORD(MKTEST) SET(EST);
               IF (ERROR_STATUS ¬= OK) THEN ELAST = TRUE;
               ELSE MKTESTBUF = MKTEST_REC;
               OBTAIN NEXT RECORD(MKTHIST) SET(HIST);
               IF (ERROR_STATUS ¬= OK) THEN HLAST = TRUE;
               ELSE MKTHISTBUF = MKTHIST_REC;
               IF (ELAST = TRUE) THEN IF (HLAST = TRUE) THEN
                                        MORE = FALSE;
           END; /* DO WHILE ¬(ELAST) OR ¬(HLAST); */
           OBTAIN NEXT RECORD(MARKET) SET(PROD_MKT);
       END; /* DO WHILE STILL MORE MARKET RECORDS IN THE SET */
       OBTAIN NEXT RECORD(PRODUCT) AREA(PRODUCT_REGION);
  END;/* DO WHILE STILL MORE PRODUCT RECORDS IN PRODUCT_REGION*/
    END DBREPORT;
```

TABLE 19

STATUS_CHECK

```
STATUS_CHECK : PROCEDURE(RN,RID);
 DECLARE RN CHAR(8),
         RID CHAR(12);
 DECLARE FOUND BIT(1) EXTERNAL;

    IF (ERROR_STATUS ¬= OK) THEN
      DO;
         DISPLAY ('THE ' || RID || ' ' || RN || ' RECORD');
         DISPLAY ('WAS NOT FOUND.');
         FOUND = FALSE;
      END;
    ELSE FOUND = TRUE;
END STATUS_CHECK;
```

TABLE 20

REC_FIND

```
REC_FIND :  PROCEDURE;
   DECLARE RN CHAR(8) VARYING EXTERNAL,
           RID CHAR(12) VARYING EXTERNAL,
           PID CHAR(12) VARYING EXTERNAL,
           MID CHAR(12) VARYING EXTERNAL;
   DECLARE FOUND BIT(1) EXTERNAL;

   IF RN = 'PRODUCT' THEN
     DO;
        PRODNAME = RID;
        OBTAIN CALC RECORD (PRODUCT);
        CALL STATUS_CHECK(RN,RID);
        IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
     END;
     ELSE IF RN = 'MARKET' THEN
            DO;
               PRODNAME = PID;
               FIND CALC RECORD(PRODUCT);
               CALL STATUS_CHECK(RN,RID);
               IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
               MKTNAME = RID;
               OBTAIN CALC RECORD(MARKET);
               CALL STATUS_CHECK(RN,RID);
               IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
            END;
       ELSE IF RN = 'MKTEST' THEN
              DO;
                 PRODNAME = PID;
                 FIND CALC RECORD(PRODUCT);
                 CALL STATUS_CHECK(RN,RID);
                 IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                 MKTNAME = MID;
                 FIND CALC RECORD(MARKET);
                 CALL STATUS_CHECK(RN,RID);
                 IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                 OBTAIN FIRST RECORD(MKTEST) SET(EST);
                 CALL STATUS_CHECK(RN,RID);
                 IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                 DO WHILE (FDATE ¬= RID);
                      OBTAIN NEXT RECORD(MKTEST) SET(EST);
                      CALL STATUS_CHECK(RN,RID);
                      IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                 END; /* DO WHILE (FDATE) ¬= RID; */
              END; /* IF RN = 'MKTEST' */
         ELSE IF RN = 'MKTHIST' THEN
                DO;
                   PRODNAME = PID;
                   FIND CALC RECORD(PRODUCT);
                   CALL STATUS_CHECK(RN,RID);
```

```
                    IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                    MKTNAME = MID;
                    FIND CALC RECORD(MARKET);
                    CALL STATUS_CHECK(RN,RID);
                    IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                    OBTAIN FIRST RECORD(MKTHIST) SET(HIST);
                    CALL STATUS_CHECK(RN,RID);
                    IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                    DO WHILE (PERIOD ¬= RID);
                       OBTAIN NEXT RECORD(MKTHIST) SET(HIST);
                       CALL STATUS_CHECK(RN,RID);
                       IF FOUND = FALSE THEN GOTO REC_NOT_FOUND;
                    END; /* DO WHILE (PERIOD) ¬= RID; */
                 END; /* RN = 'MKTHIST' */
              ELSE DISPLAY (RN || ' IS NOT A CORRECT RECORD TYPE');
REC_NOT_FOUND : ;

END REC_FIND;
```

TABLE 21

GIT

```
GIT : PROCEDURE;
     DECLARE TCARD CHAR(72) VARYING EXTERNAL,
             TSTRING CHAR(12) VARYING EXTERNAL;
     DECLARE (S,BL) FIXED DECIMAL (2,0);

     S = 1; BL = 0;
     IF (LENGTH(TCARD) >= 5 ) THEN
        DO;
           DO WHILE (SUBSTR(TCARD,1,1) = ' ');
             TCARD = SUBSTR(TCARD,2);
             IF (LENGTH(TCARD) < 5 ) THEN
                  DO;
                     NOGIT = TRUE;
                     GOTO CMD_NOT_FOUND;
                  END;
             ELSE NOGIT = FALSE;
           END; /* WHILE (SUBSTR(TCARD,1,1) = ' ') */

           BL = INDEX(TCARD,' ');
           IF (BL = 0) THEN
             DO;
                TSTRING = TCARD;
                NOGIT = FALSE;
             END;
           ELSE DO;
                   TSTRING = SUBSTR(TCARD,1,BL - 1);
                   TCARD = SUBSTR(TCARD,BL + 1);
                   NOGIT = FALSE;
                END;
        END;

     ELSE DO;
             DISPLAY (TCARD);
             DISPLAY ('CAN NOT BE PARSED.');
          END;
CMD_NOT_FOUND :;
END GIT;
```

TABLE 22

DBPARSE_CMD

```
DBPARSE_CMD : PROCEDURE;
  DECLARE TCARD CHAR(72) VARYING EXTERNAL,
          TSTRING CHAR(12) VARYING EXTERNAL;

  CALL GIT;
  IF (NOGIT = TRUE) THEN DISPLAY ('CTYP NOT FOUND');
  ELSE CTYP = TSTRING;
  IF CTYP = 'DBMENU' THEN;
   ELSE IF CTYP = 'DBREPORT' THEN;
    ELSE IF CTYP = 'LEAVE' THEN;
     ELSE DO;
             CALL GIT;
             CALL GIT;
             IF (NOGIT = TRUE) THEN DISPLAY('RID NOT FOUND');
             ELSE RID = TSTRING;
             CALL GIT;
             IF (NOGIT = TRUE) THEN DISPLAY ('RN NOT FOUND');
             ELSE RN = TSTRING;
             IF RN = 'PRODUCT' THEN;
             ELSE IF RN = 'MARKET' THEN
                   DO;
                      CALL GIT;
                      CALL GIT;
                      IF (NOGIT = TRUE) THEN DISPLAY
                                          ('PID NOT FOUND ');
                      ELSE PID = TSTRING;
                   END;
              ELSE IF RN = 'MKTEST' THEN
                    DO;
                       CALL GIT;
                       CALL GIT;
                       IF (NOGIT = TRUE) THEN DISPLAY
                                           ('PID NOT FOUND');
                       ELSE PID = TSTRING;
                       CALL GIT;
                       CALL GIT;
                       CALL GIT;
                       IF (NOGIT = TRUE) THEN DISPLAY
                                          ('MID NOT FOUND');
                       ELSE MID = TSTRING;
                    END;

               ELSE IF RN = 'MKTHIST' THEN
                    DO;
                       CALL GIT;
                       CALL GIT;
                       IF (NOGIT = TRUE) THEN DISPLAY
                                          ('PID NOT FOUND');
                       ELSE PID = TSTRING;
```

```
                        CALL GIT;
                        CALL GIT;
                        CALL GIT;
                        IF (NOGIT = TRUE) THEN DISPLAY
                                          ('MID NOT FOUND');
                        ELSE MID = TSTRING;
                      END;
          END;

END DBPARSE_CMD;
```

TABLE 23

DBADD_REC

```
DBADD_REC : PROCEDURE;
 DECLARE RN CHAR(8) VARYING EXTERNAL;

  IF RN = 'PRODUCT' THEN
    DO;
       STORE RECORD(PRODUCT);
       CALL IDMS_STATUS;
    END; /* RN = 'PRODUCT' */
  ELSE IF RN = 'MARKET' THEN
         DO;
            STORE RECORD(MARKET);
            CALL IDMS_STATUS;
         END; /* RN = 'MARKET' */
    ELSE IF RN = 'MKTEST' THEN
           DO;
              STORE RECORD(MKTEST);
              CALL IDMS_STATUS;
           END; /* RN = 'MKTEST' */
      ELSE IF RN = 'MKTHIST' THEN
        DO;
           STORE RECORD(MKTHIST);
           CALL IDMS_STATUS;
        END; /* RN = 'MKTHIST' */
END DBADD_REC;
```

TABLE 24

DBADD

```
DBADD : PROCEDURE;
  DECLARE TCARD CHAR(72) VARYING EXTERNAL,
          TSTRING CHAR(12) VARYING EXTERNAL,
          DN       CHAR (12) VARYING EXTERNAL,
          DVAL     CHAR (24) VARYING EXTERNAL;
  DECLARE
      T_CHG CHAR(36) INIT(' '),
      I FIXED DECIMAL;
  DECLARE
      CHG_OK BIT(1) INIT('1'B),
      ANS     CHAR(3);
  DISPLAY('DO YOU WANT TO ADD A NEW ' || RN || ' RECORD?');
  DISPLAY('YES/NO')
      REPLY (ANS);
  DISPLAY (' ');
  IF (ANS ¬= 'NO') THEN IF (ANS ¬= 'YES') THEN
  DISPLAY ('YOUR ONLY VALID ANSWER IS YES OR NO.')
  REPLY (ANS);
  DISPLAY (' ');
  DO WHILE (ANS = 'YES');
      DISPLAY ('TYPE THE NAME AND THE NEW VALUE FOR AN ITEM');
      DISPLAY ('IN THE NEW RECORD, THEN PRESS ENTER')
          REPLY (T_CHG);
      DISPLAY (' ');
      TCARD = T_CHG;
      CALL GIT;
      DN = TSTRING;
      CALL GIT;
      DVAL = TSTRING;
      IF RN = 'MKTHIST' THEN IF (DN ¬= 'PERIOD') THEN
        DO I = 1 TO 3 ;
          CALL GIT;
          DVAL = DVAL || ' ' || TSTRING;
        END;
      CALL DBCHG_VAL;
      DN = ' '; DVAL = ' ';
      DISPLAY ('DO YOU WANT TO ADD ANY OTHER DATA ITEMS');
      DISPLAY ('IN THE ' || RN || ' RECORD? YES/NO')
          REPLY (ANS);
      DISPLAY (' ');
      IF (ANS ¬= 'NO') THEN IF (ANS ¬= 'YES') THEN
      DISPLAY ('YOUR ONLY VALID ANSWER IS YES OR NO.')
          REPLY (ANS);
      DISPLAY (' ');
  END; /* DO WHILE ANS = 'YES' */
  IF (CHG_OK = TRUE) THEN  CALL DBADD_REC;
END DBADD;
```

TABLE 25

PROC_CMD

```
PROC_CMD : PROCEDURE;
 DECLARE CMD CHAR(72)EXTERNAL;
 DECLARE TCARD CHAR(72) VARYING EXTERNAL;
 DECLARE FOUND BIT(1) EXTERNAL;
 TCARD = CMD;
  CALL DBPARSE_CMD;
  IF CTYP = 'DBMENU' THEN CALL DBMENU;
   ELSE IF CTYP = 'DBREPORT' THEN
          DO;
             CALL DBREPORT;
             DISPLAY
             ('THE REPORT HAS BEEN SENT TO THE OUTPUT FILE');
          END;
    ELSE IF CTYP = 'LEAVE' THEN;
      ELSE DO;
  CALL REC_FIND;
  IF (FOUND = TRUE) THEN
    DO;
       IF CTYP = 'DBGET' THEN
         DO;
            CALL DBGET(RN);
            DISPLAY ('THE ' || RID || ' ' || RN || ' RECORD');
            DISPLAY ('HAS BEEN SENT TO THE OUTPUT FILE');
         END;
       ELSE IF CTYP = 'DBDELETE' THEN
              DO;
                 CALL DBDELETE(RN);
                 DISPLAY
                 ('THE ' || RID || ' ' || RN || ' RECORD');
                 DISPLAY ('HAS BEEN DELETED.');
              END;
         ELSE IF CTYP = 'DBCHANGE' THEN
                DO;
                   CALL DBCHANGE;
                   DISPLAY
                   ('THE ' || RID || ' ' || RN || ' RECORD');
                   DISPLAY ('HAS BEEN CHANGED');
                END;
           ELSE IF CTYP = 'DBADD' THEN
                  DO;
                    IF RN = 'PRODUCT' THEN RN ='MARKET';
                       ELSE IF RN = 'MARKET' THEN
                              DO;
                                 DISPLAY('DO YOU WANT TO ADD');
                                 DISPLAY
                                 ('A MKTHIST OR A MKTEST');
                                 DISPLAY('RECORD?MKTHIST/MKTEST')
                                 REPLY (RN);
                                 DISPLAY (' ');
```

```
                                    IF (RN ¬= 'MKTHIST') THEN IF
                                       (RN ¬= 'MKTEST') THEN
                                    DISPLAY
      ('YOUR ONLY VALID ANSWER IS MKTHIST OR MKTEST.')
                                    REPLY (RN);
                                    DISPLAY (' ');
                                 END;
                                 ELSE IF RN = 'NEW_PRODUCT' THEN
                                    RN = 'PRODUCT';
                     CALL DBADD;
                     DISPLAY('A NEW RECORD OCCURANCE HAS BEEN');
                     DISPLAY
                     ('INSERTED FOR THE ' || RN || ' RECORD');
                     DISPLAY ('TYPE.');
                  END;
            ELSE DO;
                     DISPLAY (CTYP || ' IS NOT A CORRECT');
                     DISPLAY ('COMMAND TYPE.');
                  END;
   END;
END;

CTYP = ' '; RID = ' '; RN = ' '; PID = ' '; MID = ' ';

END PROC_CMD;
```

TABLE 26

DBUILD

```
DBUILD : PROCEDURE;

/* IN CASE NO 'EOF' INCLUDED */

ON ENDFILE(SYSIN) INREC = 'EOF';
GET EDIT (INREC) (COL(1),A(80));
DO WHILE (SUBSTR(INREC,1,3) ¬= 'EOF');
   RN = TYP;
   PUT EDIT (INREC) (COL(1),A);
   CALL DBINIT(RN);
   GET EDIT (INREC) (COL(1),A(80));
END;  /* DO WHILE NOT EOF */

END DBUILD;
```

TABLE 27

DBMENU

```
   DBMENU : PROCEDURE;
    DECLARE NOTHING CHAR(8) INIT(' ');
      DISPLAY ('IF YOU ARE USING A COURIER TERMINAL PLEASE');
      DISPLAY ('PRESS PA 2 THEN ENTER. OTHERWISE PRESS RETURN')
      REPLY(NOTHING);


      DISPLAY
('YOU ARE NOW RUNNING UNDER IDMS. BELOW YOU WILL SEE DISPLAYED A');
      DISPLAY
 ('MENU OF THE AVAILABLE DB MANIPULATION COMMANDS. BEFORE USING
ANY');
      DISPLAY
 ('OF THE COMMANDS IN THE MENU IT IS RECOMMENDED THAT YOU READ A');
      DISPLAY
  ('DESCRIPTION OF EACH. THIS INFORMATION IS AVAILABLE FROM YOUR');
      DISPLAY
 ('INSTRUCTOR. IF YOU WANT TO LEAVE IDMS TYPE "LEAVE" THEN PRESS');
      DISPLAY
  ('ENTER FOLLOWING ANY PROMPT FOR A COMMAND.');
      DISPLAY (' ');
      DISPLAY
  ('    DBGET THE (RID) (RN) FOR (PID) IN THE (MID)');
      DISPLAY (' ');
      DISPLAY
  ('    DBDELETE THE (RID) (RN) FOR (PID) IN THE (MID)');
      DISPLAY (' ');
      DISPLAY
  ('    DBCHANGE THE (RID) (RN) FOR (PID) IN THE (MID)');
      DISPLAY (' ');
      DISPLAY
  ('    DBADD AFTER (RID) (RN) FOR (PID) IN THE (MID)');
      DISPLAY (' ');
      DISPLAY
  ('    DBMENU ;          DBREPORT ;      LEAVE ;      ');
      DISPLAY (' ');
      DISPLAY
  ('TO EXECUTE A COMMAND TYPE THE COMMAND WITH THE ');
      DISPLAY
  ('APPROPRIATE ARGUMENTS THEN PRESS RETURN.');
   END DBMENU;
```

TABLE 28

IDMS Session Showing Use of the User Front End

```
R;
runprog dbquery
2. EXEC LINKIDMS
'29B' REPLACES ' I (29B) '
I (29B) R/O
3. EXEC XDMSINIT DBASE
ENTER DMCL NAME:
devdmcl
Enter number of IDMS files in your DMCL:
1
Enter size of your IDMS files in pages:
30
IDMSINIT Completed with RETURN CODE = 0
4. &TYPE RUNNING SAMPLE APPLICATION
RUNNING SAMPLE APPLICATION
5. &STACK FILEDEF SYSPRINT DISK DBOUT LISTING
6. &STACK FILEDEF PLIDUMP DUMMY
7. &STACK FILEDEF SYS010 DISK DATABASE FILE1 A (XTENT 30 BLOCK 496
8. &STACK FILEDEF SYSIN DISK DBIN DECK * (BLOCK 80 RECFM F LRECL 80
9. &STACK
10. EXEC XDMSRUN DBQUERY
ENTER FILEDEFS FOR APPLICATION PROGRAM
EXECUTION BEGINS...
THE FOLLOWING NAMES ARE UNDEFINED:
 IDMSCLCX IDMSTRAC IDMSJLRX
THE FOLLOWING NAMES ARE UNDEFINED:
 IDMSJNL2 IDMSIOXT IDMSJLRX
IF YOU ARE USING A COURIER TERMINAL PLEASE
PRESS PA 2 THEN ENTER. OTHERWISE PRESS RETURN

YOU ARE NOW RUNNING  UNDER IDMS. BELOW  YOU WILL SEE DISPLAYED A
MENU OF THE AVAILABLE DB MANIPULATION COMMANDS. BEFORE USING ANY
OF THE  COMMANDS IN  THE MENU IT  IS RECOMMENDED THAT YOU READ A
DESCRIPTION OF  EACH. THIS  INFORMATION IS  AVAILABLE  FROM YOUR
INSTRUCTOR.  IF YOU  WANT TO LEAVE  IDMS TYPE "LEAVE" THEN PRESS
ENTER FOLLOWING ANY PROMPT FOR A COMMAND.

    DBGET THE (RID) (RN) FOR (PID) IN THE (MID)

    DBDELETE THE (RID) (RN) FOR (PID) IN THE (MID)

    DBCHANGE THE (RID) (RN) FOR (PID) IN THE (MID)

    DBADD AFTER (RID) (RN) FOR (PID) IN THE (MID)

    DBMENU ;          DBREPORT ;       LEAVE ;

TO EXECUTE A COMMAND TYPE THE COMMAND WITH THE
APPROPRIATE ARGUMENTS THEN PRESS RETURN.
```

```
ENTER COMMAND
dbreport

THE REPORT HAS BEEN SENT TO THE OUTPUT FILE

ENTER COMMAND
dbget the 1984 mktest for newlub in the midwest

RECORD TYPE : MKTEST
============

FDATE       1984
MKTSHR      015025.50.75
UNITS       20273
SALES       202730
COS         $202730
GPROF       127720
OVHDEXPS    75010
PBT         28149
TAX         46862
PAT         22494
EARN        2.50

THE 1984 MKTEST RECORD
HAS BEEN SENT TO THE OUTPUT FILE

ENTER COMMAND
dbchange the 1984 mktest for newlub in the midwest

DO YOU WANT TO CHANGE A VALUE IN MKTEST RECORD?
YES/NO
yes


TYPE THE NAME AND THE NEW VALUE FOR THE ITEM
YOU WANT TO CHANGE, THEN PRESS ENTER
earn 2.30

DO YOU WANT TO CHANGE ANY OTHER DATA ITEMS
IN THE MKTEST RECORD? YES/NO
no


THE 1984 MKTEST RECORD
HAS BEEN CHANGED

ENTER COMMAND
dbget the 1984 mktest for newlub in the midwest

RECORD TYPE : MKTEST
============

FDATE       1984
```

```
MKTSHR      015025.50.75
UNITS       20273
SALES       202730
COS         $202730
GPROF       127720
OVHDEXPS    75010
PBT         28149
TAX         46862
PAT         22494
EARN        2.30

THE 1984 MKTEST RECORD
HAS BEEN SENT TO THE OUTPUT FILE

ENTER COMMAND
dbadd after midwest market for newlub

DO YOU WANT TO ADD
A MKTHIST OR A MKTEST
RECORD?MKTHIST/MKTEST
mkthist


DO YOU WANT TO ADD A NEW MKTHIST RECORD?
YES/NO
yes


TYPE THE NAME AND THE NEW VALUE FOR AN ITEM
IN THE NEW RECORD, THEN PRESS ENTER
period 72thru77

DO YOU WANT TO ADD ANY OTHER DATA ITEMS
IN THE MKTHIST RECORD? YES/NO
yes


TYPE THE NAME AND THE NEW VALUE FOR AN ITEM
IN THE NEW RECORD, THEN PRESS ENTER
year1 11111 11111 11111 11111

DO YOU WANT TO ADD ANY OTHER DATA ITEMS
IN THE MKTHIST RECORD? YES/NO
yes


TYPE THE NAME AND THE NEW VALUE FOR AN ITEM
IN THE NEW RECORD, THEN PRESS ENTER
year2 22222 22222 22222 22222

DO YOU WANT TO ADD ANY OTHER DATA ITEMS
IN THE MKTHIST RECORD? YES/NO
yes
```

```
TYPE THE NAME AND THE NEW VALUE FOR AN ITEM
IN THE NEW RECORD, THEN PRESS ENTER
year3 33333 33333 33333 33333

DO YOU WANT TO ADD ANY OTHER DATA ITEMS
IN THE MKTHIST RECORD? YES/NO
yes


TYPE THE NAME AND THE NEW VALUE FOR AN ITEM
IN THE NEW RECORD, THEN PRESS ENTER
year4 44444 44444 44444 44444

DO YOU WANT TO ADD ANY OTHER DATA ITEMS
IN THE MKTHIST RECORD? YES/NO
yes


TYPE THE NAME AND THE NEW VALUE FOR AN ITEM
IN THE NEW RECORD, THEN PRESS ENTER
year5 55555 55555 55555 55555

DO YOU WANT TO ADD ANY OTHER DATA ITEMS
IN THE MKTHIST RECORD? YES/NO
no


A NEW RECORD OCCURANCE HAS BEEN
INSERTED FOR THE MKTHIST RECORD
TYPE.

ENTER COMMAND
dbget the 72thru77 mkthist for newlub in the midwest

RECORD TYPE : MKTHIST
=============

PERIOD   72THRU77
YEAR1    11111 11111 11111 11111
YEAR2    22222 22222 22222 22222
YEAR3    33333 33333 33333 33333
YEAR4    44444 44444 44444 44444
YEAR5    55555 55555 55555 55555

THE 72THRU77 MKTHIST RECORD
HAS BEEN SENT TO THE OUTPUT FILE

ENTER COMMAND
dbreport

THE REPORT HAS BEEN SENT TO THE OUTPUT FILE

ENTER COMMAND
```

```
dbdelete the 78thru83 mkthist for newlub in the midwest

THE 78THRU83 MKTHIST RECORD
HAS BEEN DELETED.

ENTER COMMAND
dbreport

THE REPORT HAS BEEN SENT TO THE OUTPUT FILE

ENTER COMMAND
dbmenu

IF YOU ARE USING A COURIER TERMINAL PLEASE
PRESS PA 2 THEN ENTER. OTHERWISE PRESS RETURN

YOU ARE NOW RUNNING UNDER IDMS. BELOW YOU WILL SEE DISPLAYED A
MENU OF THE AVAILABLE DB MANIPULATION COMMANDS. BEFORE USING ANY
OF THE COMMANDS IN THE MENU IT IS RECOMMENDED THAT YOU READ A
DESCRIPTION OF EACH. THIS INFORMATION IS AVAILABLE FROM YOUR
INSTRUCTOR. IF YOU WANT TO LEAVE IDMS TYPE "LEAVE" THEN PRESS
ENTER FOLLOWING ANY PROMPT FOR A COMMAND.

    DBGET THE (RID) (RN) FOR (PID) IN THE (MID)

    DBDELETE THE (RID) (RN) FOR (PID) IN THE (MID)

    DBCHANGE THE (RID) (RN) FOR (PID) IN THE (MID)

    DBADD AFTER (RID) (RN) FOR (PID) IN THE (MID)

    DBMENU ;          DBREPORT ;       LEAVE ;

TO EXECUTE A COMMAND TYPE THE COMMAND WITH THE
APPROPRIATE ARGUMENTS THEN PRESS RETURN.

ENTER COMMAND
leave

$
R;
```

Appendix C

THE EMPIRE FILES FOR THE DSS PROBLEM ENVIRONMENTS

The purpose of this appendix is to document the software support tools needed to complete the DSS problem environments. It contains a copy of the EMPIRE models, EMPIRE report, EMPIRE data, and EMPIRE control files. It also contains a copy of the session the student will experience in completing DSS problem environment two.

C.1 CHANGES TO THE CMS VIRTUAL MACHINE FOR EMPIRE

There are some minor changes to the CMS virtual machine that are necessary to enable one to use EMPIRE in the CMS environment. The user storage must be at least 700k. This can be set at logon by using the command 'RESTOR 700k' or it could be palced in the PROFILE EXEC. A second chamge that is necessary is to provide the user with 4 loader tables. The is accomplished by using the command 'SET LDRTBLS 4' at logon and not later. The 'SET LDRTBLS' command defines the number of pages of storage provided for the loader tables. EMPIRE has an interactive DEBUG facility that may be useful

if students are required to write an EMPIRE model program. In order to use this facility the virtual machine must be provided with 5 pages of storage for the loader tables. This is accomplished by issuing the command 'SET LDRTBLS 5' immediately after logon.

In the model for the forecast of potential earnings which will be used for DSS problem environments two and three included as comments are two statements which will be excluded for problem environment two but included for problem environment three. These statements are the 'OPTION SECTION' and 'CONTROL SAMPCON' statements.

TABLE 29

EMPIRE model for Potential Earnings Forecast

```
;OPTION SECTION
;CONTROL PROFORMA
  COLUMN SECTION
  CREATE QTR 1:20 "HISTORY"
  MAR     "PERIOD/ENDING/3-31"
  JUN     "PERIOD/ENDING/6-30"
  SEP     "PERIOD/ENDING/9-30"
  DEC     "PERIOD/ENDING/12-31"
  TOTAL   "TOTAL/FOR/YEAR"
  ROW SECTION
    MKTSHR INPUT "MARKET SHARE"
    TUNITS     "TOTAL MARKET"
    UNITS      "UNITS SOLD"
    SALES      "SALES REVENUE"
    COS        "COST OF SALES"
    GPROF      "GROSS PROFIT"
    OVHD       "OVERHEAD EXPENSE" 5000*
    PBT        "PROFIT BEFORE TAX"
  SCALAR SECTION
    PRICE INPUT "UNIT COST"
    UCOST INPUT "UNIT COST"
    TAXRAT "TAX RATE" .48
    TAX        "INCOME TAX"
    PAT        "PROFIT AFTER TAX"
    EARN       "EARNINGS PER SHARE"
  RULES SECTION
    FOR COL=MAR TO DEC DO
        UNITS=ROUND(MKTSHR(COL)*TUNITS(COL),0)
        SALES=UNITS*PRICE
        COS=UNITS*UCOST
        GPROF=SALES-COS
        IF COL NE MAR THEN DO
            IF SALES GT SALES(COL-1) THEN &
                OVHD=OVHD(COL-1)+.05*(SALES-SALES(COL-1))
            ELSE OVHD=OVHD(COL-1)
        END
        PBT=GPROF-OVHD
    END
    TOTAL=MAR+JUN+SEP+DEC
    TAX=TAXRAT*PBT(TOTAL)
    PAT=PBT(TOTAL)-TAX
    EARN=PAT/12000
```

TABLE 30

EMPIRE Report Control for DSS problem env. 2

```
SELECT MAR:TOTAL
COLUMNWIDTH 10
TITLE 1 CENTER "CHEMICAL DIVISION"
TITLE 2 CENTER "INCOME STATEMENT"
TITLE 3 CENTER /"PRODUCT 4792"//
POSITION 5
PRINT UNITS
SKIP
PREFIX "$"
PRINT SALES
PREFIX
PRINT COS
LINE
PRINT /,GPROF,OVHD
LINE
PRINT /,PBT,TAX@5,/,PAT@5,/
DECIMAL 2
PRINT EARN@5
```

TABLE 31

EMPIRE Control for DSS problem env. 3

```
DATA
OPEN HIST2
READ TUNITS AS CHEM46
CLOSE
MKTSHR(MAR) .025,.075,.15,.25
UCOST 6.3
PRICE 10
END
FIT
SIMPLE REGRESSION
TUNITS
COL
QTR1,QTR20
BEST
END
4
YES
TUNITS
MAR
END
RUN MYSAMPLE
```

TABLE 32

EMPIRE Data File for problem env. 2 & 3

```
CHEM46 32450 33270 32990 33440 &
33560 35250 35360 35500 &
35170 36660 36540 36770 &
37280 36820 37410 38580 &
38160 38220 38830 39430
```

TABLE 33

EMPIRE model file for problem env. 1

```
;*********************************************************
;THIS MODEL ANALYZES THE CONSOLIDATED TEN YEAR FINANCIAL AND
;OPERATING SUMMARIES OF JUSTIN ALLEN PETROLEUM CORPORATION
;FROM 1970 TO  1979
;*********************************************************
OPTION SECTION ;SECTION REDEFINES REPORT DEFAULTS AND TRANSFERS
               ;CONTROL TO FSUM
CONTROL FSUM
ROWTITLE 100
COLUMN 14
WIDTH 132
COLUMN SECTION ;*******************************************************
;HISTORICAL DATA
A1970 "1970"
A1971 "1971"
A1972 "1972"
A1973 "1973"
A1974 "1974"
A1975 "1975"
A1976 "1976"
A1977 "1977"
A1978 "1978"
A1979 "1978"
;SUMMARY DATA
S7074 "SUBTOTAL FOR/1970-1974"
S7579 "SUBTOTAL FOR/1975-1979"
T7079 "TOTAL FOR/1970-1979"
ROW SECTION ;*********************************************************
;EARNINGS STATISTICS
NSALES "NET SALES......................................"
ECOSTS "  EMPLOYMENT COSTS..............................."
MATSER "   MATERIALS AND SERVICE........................."
DEP " DEPRECIATION....................................."
TAXES "   INCOME TAXES................................."
TCOSTS "          "
OPINC "OPERATING INCOME................................."
INTINC " INTEREST, DIVIDENDS AND OTHER INCOME..........."
INTEXP " INTEREST AND OTHER DEBT CHARGES................"
CLCOST " ESTIMATED CLOSEDOWN COST......................."
FLDEXP " FLOOD EXPENSE................................."
INCBTX "INCOME (LOSS) BEFORE INCOME TAXES..............."
INCTAX "INCOME TAXES..................................."
NETINC "  AMOUNT......................................."
RULES SECTION ;*******************************************************
;COMPUTE INCOME STATEMENT RELATIONSHIPS
FOR COL=A1970 TO A1979 DO
     TCOSTS=COLSUM(ECOSTS,TAXES,COL)
     OPINC=NSALES-TCOSTS
     INCBTX=OPINC+COLSUM(INTINC,FLDEXP,COL)
```

```
     NETINC=INCBTX-INCTAX
END
;SUMMARY OF PERFORMANCE
FOR ROW=NSALES TO NETINC DO
     A(S7074,ROW)=SUM(A(1,ROW),A1970,A1974)
     A(S7579,ROW)=SUM(A(1,ROW),A1975,A1979)
     A(T7079,ROW)=SUM(A(1,ROW),S7074,S7579)
END
```

TABLE 34

EMPIRE report control file for problem env. 1

```
PAGE 60,6
WIDTH 132
COLUMN 14
ROWTITLE 60
HEADING CENTER
EXCHANGE 0 "    -    "
MARKS OFF
SELECT S7074:T7079
TITLE 1 LEFT "JUSTIN ALLEN PETROLEUM CORP AND SUBSIDIARY COMPANIES"
TITLE 2 LEFT "TEN YEAR FINANCIAL SUMMARY"//
SUBTITLE "(DOLLARS IN MILLIONS)"
SUBTITLE "EARNINGS STATISTICS"
SUFFIX
DECIMAL 1
PRINT NSALES
SUBTITLE "COSTS AND EXPENSES:"
SUBTITLE "  OPERATING CHARGES:"
PRINT ECOSTS
PREFIX
PRINT MATSER:DEP
SUBTITLE "  TAXES, OTHER THAN EMPLOYMENT AND"
PRINT TAXES=
PREFIX "$"
PRINT TCOSTS=OPINC
PREFIX
SUBTITLE "OTHER INCOME (EXPENSE)"
PRINT INTINC:FLDEXP=
PREFIX "$"
PRINT INCBTX
PREFIX
PRINT INCTAX=
SUBTITLE "NET INCOME (LOSS)"
PREFIX "$"
PRINT NETINC
```

TABLE 35

EMPIRE Control file for problem env. 1

```
DATA
OPEN FSUMDATA
READ NSALES
READ ECOSTS
READ MATSER
READ DEP
READ TAXES
READ INTINC
READ INTEXP
READ CLCOST
READ FLDEXP
READ INCTAX
CLOSE
END
RUN FINANSUM
```

TABLE 36

EMPIRE data file for problem env. 1

```
NSALES 2935.4,2969.1,3113.6,4137.6,5381.0,4977.2,5248.0,5370.0 &
6184.9,7137.2
ECOSTS 1361.7,1323.6,1448.6,1759.3,2072.0,2139.2,2313.6,2368.5 &
2550.0,2939.1
MATSER 1214.4,1198.5,1208.4,1765.1,2442.7,2240.4,2373.0,2707.0 &
2889.8,3367.7
DEP 166.0,159.3,180.8,196.1,210.9,234.2,275.6,300.1,321.9,351.3
TAXES 54.6,60.7,60.5,58.8,63.1,68.1,70.8,72.3,74.4,86.1
INTINC 28.8,32.5,26.6,41.3,67.7,51.1,56.7,40.2,47.7,81.7
INTEXP -33.4,-38.3,-40.3,-43.0,-44.0,-63.4,-77.7,-82.5,-86.4 &
-80.0
CLCOST 7*0,-750
FLDEXP(8) -41.0
INCTAX 44.0,82.0,67.0,150.0,274.0,41.0,26.0,-463.0,85.0,119.0
```

TABLE 37

EMPIRE session for problem env. 2

```
 empire

  -----------
  E M P I R E
  TRANSLATOR
    VER 3B
  (C) 1982 ADR
  -----------



   TRANSLATOR>
  execute mysample

   ***TRANSLATION COMPLETED***

  R;

  ***CREATING MYSAMPLE, PLEASE WAIT...

  ***LOADING MYSAMPLE, PLEASE WAIT...

   -----------
   E M P I R E
    EXECUTIVE
      VER 3B
   (C) 1982 ADR
   -----------



   EXEC>
  data
   DATA>
  open hist2
   DATA>
  read tunits as chem46
   DATA>
  close
   DATA>
  mktshr(mar) .025,.075,.15,.25
   DATA>
  ucost 6.3
   DATA>
  price 10
```

```
 DATA>
end
 <CURRENT DATA "SAVED" INTO BACKUP WORKSPACE>
 EXEC>
fit

 ANALYSIS OPTION?
simple regression
 DEPENDENT ITEM NAME?
tunits
 INDEPENDENT ITEM NAME, OPTIONAL OFFSET?
col
 ENTER FIRST & LAST COLUMNS TO BE USED:
qtr1,qtr20

 CURVE TYPE?
best
CURVE TYPE                  R**2  A-COEFF     B-COEFF
 1  TUNITS=A+B*COL          .958 32410.207    349.932
 2  TUNITS=A*EXP (B*COL)    .953 32516.371       .010
 3  TUNITS=A*(COL**B)       .888 31157.781       .069
 4  TUNITS=A+B/COL          .533 37327.836  -6911.773
 5  TUNITS=1/(A+B*COL)
 EXEC>
run mysample

 EXEC>
print from myrep
 ADJUST PAPER, THEN ENTER A CARRIAGE RETURN
TO PROCEED=>
```

MODEL: MYSAMP DATE: 07/14/83 13:44 PAGE: 1

CHEMICAL DIVISION
INCOME STATEMENT

PRODUCT 4792

PERIOD ENDING 3-31	PERIOD ENDING 6-30	PERIOD ENDING 9-30	PERIOD ENDING 12-31		TOTAL FOR YEAR
994	3,008	6,069	10,202	UNITS SOLD	20,273
$ 9,940	$ 30,080	$ 60,690	$ 102,020	SALES REVENUE	$ 202,730
6,262	18,950	38,235	64,273	COST OF SALES	127,720
3,678	11,130	22,455	37,747	GROSS PROFIT	75,010
5,000	6,007	7,538	9,604	OVERHEAD EXPENSE	28,149
(1,322)	5,123	14,918	28,143	PROFIT BEFORE TAX	46,862
				INCOME TAX	22,494
				PROFIT AFTER TAX	24,368
				EARNINGS PER SHARE	2.03

EXEC>
exit

E M P I R E - END OF SESSION

R;

BIBLIOGRAPHY

Applied Data Research. EMPIRE Decision Support System, Beginner's Guide. January,1980 Draft.

Applied Data Research,Inc. EMPIRE Modeling, Analysis, and Reporting System : An Introduction. September, 1978.

Applied Data Research, Inc. EMPIRE Decision Support System User Reference Manual. September, 1980.

Applied Data Research, Inc. EMPIRE Decision Support System VM/370 User Guide. P12A-VM-01.

Awad, Elias M. Business Data Processing. Englewood Cliffs, New Jersey : Prentice Hall Inc, 1980.

Conroe, Kenneth. The CMS Cookbook. Manhattan, Kansas : Computing Center, Kansas State University. December, 1982.

Cullinane Corporation. IDMS Data Definition Languages, Utilities, and GCI Reference Guide. Release 3.1, revision 1. April, 1975.

Cullinane Corporation. IDMS Database Design and Definition Guide. Release 4.0, revision 4. February, 1977.

Cullinane Corporation. IDMS Programmers Reference Guide. Release 4.5. August, 1977.

International Business Machines. OS and DOS PL/I Language Reference Manual. 1st ed., September, 1981.

International Business Systems. OS PL/I Optimizing Compiler : Programmers Guide. Release 4.0, 1981.

International Business Machines. IBM Virtual Machine/System Product : CMS Command and Macro Reference. 1st ed., September 1980.

International Business Machines. IBM Virtual Machine/System Product : System Programmer's Guide. 1st ed.,September 1980.

Shea, William. IDMS Query Language. Masters Report, 1977. Kansas State University, 1977.

Strunk, Neal. Support Tools for an Undergraduate Management Information System Course. Masters Report, 1982. Kansas State University, 1982.

DESIGN AND IMPLEMENTATION OF PROBLEM ENVIRONMENTS AND SOFTWARE SUPPORT TOOLS FOR A MANAGEMENT INFORMATION SYSTEMS COURSE

by

ROBERT A. BIRCHARD

B.A., Emporia State University, 1977

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree of

MASTER of SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1983

This report describes the design and implementation of problem environments and software support tools for a management information systems (MIS) course to be offered for information systems majors by the Computer Science department at Kansas State University. Three areas were chosen from the student behavioral learning objectives (SBLO) to develop problem environments for classroom use. These areas are Office Automation (OA), Data Base Management Systems (DBMS), and Decision Support Systems (DSS).

A data base of product information and a user 'front end' were designed and implemented for the DBMS problem environments. The information in the data base relates to the DSS problem environments and ties the two areas together.

The EMPIRE DSS was used for the DSS problem environments and the IDMS DBMS for the DBMS problem environments. The OA problem environments were implemented without development of new software support tools on the Interdata 8/32.